DRAM Fault Analysis and Test Generation

DRAM Fault Analysis and Test Generation

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen

op maandag 27 juni 2005 om 15.30 uur

door

Zaid AL-ARS

elektrotechnisch ingenieur
geboren te Bagdad, Irak

Dit proefschrift is goedgekeurd door de promotor:

Prof.dr.ir. A.J. van de Goor

Samenstelling promotiecommissie:

Rector Magnificus, chair	Delft University of Technology, Delft, The Netherlands
Prof.dr.ir. A.J. van de Goor	Delft University of Technology, Delft, The Netherlands
Prof.dr.ing. E.J. Aas	Norwegian Univ. of Sci. and Tech., Trondheim, Norway
Prof.dr. C.I. Beenakker	Delft University of Technology, Delft, The Netherlands
Prof.dr. H. Corporaal	Tech. Univ. of Eindhoven, Eindhoven, The Netherlands
Dr. S. Mijalkovic	Delft University of Technology, Delft, The Netherlands
Dr. G. Müller	Infineon Technologies, Munich, Germany
Prof.dr. L. Nanver	Delft University of Technology, Delft, The Netherlands
Prof.dr. J.R. Long	Delft University of Technology, Delft, The Netherlands

This work has been supported by Infineon Technologies, Munich, Germany, with a grant to the Delft University of Technology.

Graphical cover illustration by ATOC Designs (http://www.atocdesigns.com/)

ISBN-13: 978-90-9019612-1
ISBN-10: 90-9019612-9
NUR 959

Copyright © 2005 Zaid Al-Ars
All rights reserved. No part of this publication, either text or image may be used for any purpose other than personal use. Reproduction, modification, storage in a retrieval system or retransmission, in any form or by any means, electronic, mechanical, or otherwise, for reasons other than personal use, is strictly prohibited without prior written permission from the author.

To my father for teaching me how to have a dream,
and to my mother for showing me how to make it a reality.

DRAM Fault Analysis and Test Generation

Abstract

The dynamic random access memory (DRAM) is the most widely used type of memory in the market today, due to its important application as the main memory of the personal computer (PC). These memories are elaborately tested by their manufacturers to ensure a high quality product for the consumer. However, this testing process is responsible for a large portion of the cost of these memories, standing now at 40% and gradually rising with each new generation. Companies usually develop the required memory tests in an ad hoc manner, relying heavily on an expensive combination of experience and statistics to construct the best test approach, the price of which is ultimately paid by the end consumer.

In this thesis, we propose a new alternative approach to the development of industrial memory testing that is more systematic and less expensive than the currently prevalent test approaches. The new approach is based on the introduction of a number of fault analysis algorithms that enable the application of electrical Spice simulations to develop effective memory tests in a short amount of time. The new approach makes it possible to enhance memory tests in many different manufacturing stages, starting from the initial test application stage where silicon is manufactured, through the memory ramp-up stage where products are shipped to the customer, and ending with the test adaptation stage, based on memory failures in the field. The new test development approach has been implemented and evaluated at Infineon Technologies, a leading DRAM manufacturer, to develop tests for their DRAM products.

This thesis describes the details of the proposed test development algorithms, along with the way they are practically implemented for DRAM devices. Two different aspects necessary for DRAM tests are identified: test patterns and test stresses, then methods to generate and optimize both of them are proposed. In addition, the thesis discusses the results of applying the proposed approach to a large number of DRAM defects, modeled from known DRAM failures on the layout level. Finally, the thesis ends with an elaborate case study to generate and optimize tests for a specific DRAM problem for which the new test development approach has proven its effectiveness.

Fout Analyseren en Test Genereren in DRAMs

Samenvatting (Abstract in Dutch)

Het dynamic random access memory (DRAM) is heden het meest gebruikte type geheugen, wegens zijn belangrijke toepassing als het werkgeheugen van de personal computer (PC). Geheugen chips worden door geheugen fabrikanten uitvoerig getest om de hoge kwaliteit van hun producten te garanderen. Dit testproces is verantwoordelijk voor een groot gedeelte van de kosten van een geheugen, die nu rond 40% is, en geleidelijk met elke nieuwe geheugen generatie toeneemt. In het algemeen ontwikkelen bedrijven de vereiste geheugentests op een ad hoc manier, gebaseerd op een dure combinatie van ervaring en statistieken, om de beste tests te construeren. De prijs daarvan wordt uiteindelijk door de consument betaald.

In dit proefschrift, stellen wij een nieuwe en alternatieve benadering van industrieel test ontwikkeling voor, die systematischer en goedkoper is dan de huidige testmethoden. De nieuwe benadering is gebaseerd op de introductie van een aantal algoritmen om geheugenfouten door middel van elektronische Spice simulaties te analyseren. Dit maakt het mogelijk efficiënte geheugentests op korte termijn te ontwikkelen en om de tests in vele verschillende productiestadia te verbeteren. De nieuwe testontwikkeling methoden zijn bij Infineon Technologies, een belangrijke DRAM fabrikant, uitgevoerd en gevalueerd om tests voor hun DRAM producten te ontwikkelen.

Het proefschrift beschrijft de details van de voorgestelde testontwikkeling algoritmen, samen met de manier waarop zij praktisch voor DRAMs worden uitgevoerd. Twee belangrijke DRAM testaspecten worden geïdentificeerd (testpatronen, en teststress), en methoden om beide testaspecten te produceren en te optimaliseren worden voorgesteld. Bovendien bespreekt het proefschrift de toepassings resultaten van de voorgestelde testmethoden op een groot aantal bekende DRAM defecten, die belangrijk zijn op het layout-niveau. Tot slot eindigt het profschrift met een gedetailleerde casestudy, om tests voor een specifiek DRAM probleem te produceren en te optimaliseren. Hierin wordt het nut van de nieuwe benadering van testontwikkeling bewezen.

Preface

From the bulky drum memory used in the early computer systems, through the relatively expensive core memory, to the semiconductor memory used extensively in most of today's integrated circuits (ICs), memory has played a vital, albeit quiet role in the history of computing. However, this important role is gradually becoming ever clearer, due to the exponentially increasing amount of memory used in ICs today, and because a constantly growing percentage of these ICs is being dedicated to implement memory devices. According to the International Technology Roadmap for Semiconductors (ITRS), a leading authority in the field of semiconductors, memory occupied 20% of the area of an IC in 1999, 52% in 2002, and is forecast to occupy up to 90% of the area by the year 2011. Due to this considerable usage of memory in many ICs, improvements in the design and fabrication process of memory devices have a considerable impact on the overall IC characteristics. Therefore, reducing the energy consumption, increasing the reliability or, most importantly, reducing the price of memories has a similar impact on the overall systems that contain them.

Furthermore, a large portion of the price of a memory today is incurred by the high costs of memory testing, which has to satisfy very high quality constraints ranging from 50 failing parts per million (ppm) for computer systems to less than 10 ppm for mission-critical applications (such as those in the automotive industry). Memory testing is expensive because of the high cost of the test equipment (a production memory tester costs more than $500,000), a cost that has to be distributed over all of the produced chips, good and bad. As an example, a well-designed test of a typical 64 Mb DRAM comprises 39% of its total cost. The test cost of a 256 Mb DRAM is estimated to be more than 50% of the total cost, making the field of memory testing a rather significant and influential part of the whole memory production process.

Despite the huge current and future growth potential of the filed of memory testing, not much research has been dedicated to this important aspect of the IC industry. This lack of research is mainly due to the fact that memory testing is a relatively new and developing field, and to the unorthodox mathematical techniques it employs for fault modeling and test generation. Moreover, only a limited number of research projects exist in this field in the form of a collaboration between the industry and the academic world, due to the company-confidential nature of the memory test information.

A word on the project

The work in this thesis describes the results of a joint research project titled "Fault Analysis of DRAMs Using Defect Injection and Simulation" between Infineon Technologies, Munich, Germany, and the Delft University of Technology, Delft, The Netherlands, to analyze the faulty behavior of DRAMs produced by Infineon. It represents an example of a small number of projects between the memory industry and academia, meant to study the faulty behavior of memory devices and to suggest new, theoretically justified memory tests. The current project is a continuation of a previous joint project between Siemens Semiconductors (presently Infineon Technologies) and the Delft University of Technology, which supplied the material for the masters thesis "Analysis of the Space of Functional Fault Models and Its Application to Embedded DRAMs", by Zaid Al-Ars.

When we started out working on the project, the objective was to come up with a set of memory tests able to effectively detect the faults commonly encountered in DRAM chips. But soon it became clear that designing such a set of tests is not possible, because of the changing nature of the faulty behavior that takes place with every new technology or new design that comes into the market. As a result, the objective of the project rapidly shifted from generating a set of generic tests for all DRAMs to devising new test approaches and test methods that enable systematic and cost-effective generation and optimization of memory tests on a memory-specific basis.

As a result, the project includes research aspects that have not been investigated before by the memory testing research community. One example of the new aspects treated in this project is the theoretical analysis of memory test stresses as an integral part of the test process. This has commonly been considered a practical issue in memory test development, and has not been understood nor justified from a theoretical point of view.

Due to the extensive industrial input to the project, many practical aspects have been incorporated into the research, while the theoretical roots of the project ensured the rigorous nature of the performed analysis. In such an environment, where different aspects of theory and practice need to be taken into consideration, it is rather difficult to fulfill the requirements of both. However, we hope that in this research project we were able to strike a good tradeoff between the two, without compromising any important aspects on either side.

A word on the thesis

The information included in this thesis is fully self-contained and does not assume any prior knowledge of specific aspects of memory devices on the part of the reader. A reader with minimum understanding of memories should be able to read through the whole thesis with ease, and gradually build up the information necessary to understand the described concepts.

The thesis is organized as follows. Chapter 1 gives a general introduction to the history and status of memory technology. It also provides an introduction to the main concepts and challenges of memory testing. Chapter 2 describes the external behavior and the internal architecture of modern DRAM devices in some detail. Chapter 3 builds on the information in Chapter 2 to give a detailed description of the buildup and behavior of internal electrical circuits, in addition to the possible layout implementations of DRAMs. Chapter 4 presents the theoretical bases used to model the faulty behavior of DRAMs. Chapter 5 presents the new simulation-based fault analysis methods, used to generate the memory test patterns and to optimize the needed memory test stresses. Chapter 6 discusses the concepts of coupling and crosstalk between internal memory lines, and the way these influence the faulty behavior of the memory. In Chapter 7, the faulty behavior of a large number of memory defects is evaluated, based on an electrical level simulation of the behavior. Chapter 8 shows how to use the theoretical analysis of DRAM faults performed in Chapter 4 to derive the required DRAM tests. Chapter 9 discusses an elaborate case study performed on a specific well-known DRAM defect, clearly showing the full test generation and optimization potential of the simulation-based fault analysis methods. Finally, Chapter 10 summarizes the thesis and ends with the conclusions and recommendations.

This thesis has a web site on the Internet at http://ce.et.tudelft.nl/~pazaid/books/phd.html. This site contains the latest information regarding this thesis, updated errata and links to related work. The site also gives information about the performed fault analysis work that has been left out from this thesis. In addition, links are given to the raw fault analysis data generated in the course of the simulation-based analysis described in the thesis. Any questions, comments, suggestions or typos can be submitted using the e-mail link on the site, and will be greatly appreciated.

This thesis has been typeset using the LaTeX 2ε package on a Linux machine. Most of the figures have been constructed using Xfig, the freeware vector graphics tool included with most Linux distributions. Simulations have been performed using Titan, the Siemens/Infineon in-house Spice-based circuit simulator, while figures of simulation output and figures of function plots have been produced using gnuplot, the freeware function plotting utility. The graphical cover illustration was designed by the media design company ATOC Designs (http://www.atocdesigns.com).

Acknowledgments

This thesis work has lasted for a period of more than four years, and in that period it received the kind support and encouragement of countless individuals, without whom this thesis would not have been realized. First and foremost, I wish to thank Prof. Ad van de Goor, my professor, from the Delft University of Technology, for his active involvement and continued support that helped shape many parts of the project. This thesis was only made possible with his guidance, meticulous reviews, and brilliant advice. My gratitude also goes to Detlev Richter, my manager at the

product engineering group in Infineon Technologies, for his vision of merging the two worlds of theory and practice, and for the confidence he had in our work. I would further like to thank Jens Braun, my supervisor at Infineon for his valuable comments and for the countless hours he dedicated to discuss every step of the project. At its onset, the project was also supported by Michael Kund and Bernd Gauch from Infineon, who merit my gratitude as well.

Special thanks go to Georg Müller and Carsten Ohlhoff from Infineon for ensuring continued support to the project after the departure of Detlev Richter and Jens Braun. I wish to give my personal gratitude to Martin Herzog from the Technical University of Munich for creating some of the fault analysis programs and automation scripts used in this thesis, and for applying them to the defective memory models. The thesis also benefited greatly from the input and suggestions to improve the quality of the work contributed by Jörg Vollrath and Martin Versen from Infineon.

I also wish to thank the members of the product engineering department, my department at Infineon, for keeping up a great working atmosphere and making my stay among them cheerful and fun. From those, I would especially like to mention Martin Meier, Henno Harder and Christian Jaekel. Many thanks also go to the computer-aided design department, particularly Stefan Scholl, for their round-the-clock support of our work. Gratitude is also due to the members of the design department, most notably Eckehard Plaettner for setting up the memory design environment, and Udo Weber for performing kerf measurements of special wafer structures needed for defect modeling.

I would also like to thank Mirjam Nieman, the editor of technical English at the Delft University of Technology, for her review and commentary on the thesis. I further want to thank the members of the Laboratory of Computer Engineering in the Delft University of Technology, my group, for their moral support and help to finalize this thesis, many of whom I consider personal friends and colleagues. Special mention is due to Prof. Stamatis Vassiliadis for his support of the project, to Said Hamdioui for his continuous encouragement, to Bert Meijs for keeping up an excellent and flexible working environment, and finally to Lidwina Tromp, our group assistant, for her help and understanding during the long periods I spent away from the university.

And last but not least, my many thanks go to my family, and my parents in particular, for their love, unlimited support and patience along the years.

Contents

Abstract **vii**

Samenvatting (Abstract in Dutch) **ix**

Preface **xi**

1 Introduction **1**
- 1.1 Semiconductor memories . . . 2
  - 1.1.1 Definition and physical form . . . 2
  - 1.1.2 Types of semiconductor memories . . . 3
- 1.2 Memory testing . . . 8
  - 1.2.1 Definition of memory testing . . . 8
  - 1.2.2 Manufacturing test flow . . . 10
  - 1.2.3 Frontend and backend testing . . . 11
- 1.3 Contribution of the project . . . 16
  - 1.3.1 Framework of the contribution . . . 16
  - 1.3.2 Details of the contribution . . . 20
- 1.4 Outline of the thesis . . . 21
- Summary . . . 23

2 DRAM behavior and architecture **25**
- 2.1 Modeling of DRAMs . . . 26
- 2.2 Behavioral DRAM model . . . 27
  - 2.2.1 DRAM block diagram . . . 28
  - 2.2.2 DRAM operations and timing diagrams . . . 30
- 2.3 Functional DRAM model . . . 34
  - 2.3.1 Functional block diagram . . . 34
  - 2.3.2 Cell array organization . . . 36
  - 2.3.3 Internal DRAM behavior . . . 39
- Summary . . . 42

3 DRAM design and implementation **43**
- 3.1 Basics of MOS transistors . . . 43
  - 3.1.1 Physical transistor structure . . . 44

3.1.2 MOS transistor behavior 45
3.1.3 Transistor as a switch 48
3.2 Electrical DRAM model 49
3.2.1 Electrical memory circuits 49
3.2.2 Memory cell 50
3.2.3 Sense amplifier 54
3.2.4 Other DRAM circuits 57
3.3 DRAM layout model 60
3.3.1 DRAM trench capacitor 61
3.3.2 DRAM stacked capacitor 62
3.3.3 Cell array layout 63
Summary 65

4 Modeling faulty memory behavior 67
4.1 Definition of fault models 67
4.1.1 Representing operation sequences 68
4.1.2 Fault primitives and fault models 69
4.2 Generic space of faults 69
4.2.1 Classification of fault primitives 69
4.2.2 Static fault models 71
4.2.3 Complexity of fault primitives 76
4.2.4 Fault primitive detection and march tests 77
4.3 DRAM-specific faults 78
4.3.1 Voltage dependent faults 79
4.3.2 Time dependent faults 82
4.3.3 Space of DRAM faults 86
4.4 Industrial test practices 89
4.4.1 Stress combinations 89
4.4.2 Device-specific fault primitives 91
4.4.3 Examples of new notation 94
Summary 96

5 Fault analysis approximation methods 97
5.1 Conventional analysis 97
5.1.1 Complexity of simulation 98
5.1.2 Example of conventional analysis 99
5.1.3 Fault analysis time 101
5.2 Approximate simulation 101
5.2.1 Example of 1D analysis 102
5.2.2 Fault analysis time 106
5.3 Two dimensional analysis 107
5.4 DRAM-specific simulation 111
5.4.1 Approximation with DRAM commands 112
5.4.2 Stress optimization methodology 113

Summary . . . 116

6 Effects of bit line coupling 117
6.1 Concept of BL coupling . . . 117
  6.1.1 Modeling BL coupling . . . 118
  6.1.2 Effects of coupling . . . 120
6.2 Simulation of BL coupling . . . 122
  6.2.1 Fault analysis method . . . 122
  6.2.2 Simulation results . . . 124
6.3 Impact of BL twisting . . . 126
  6.3.1 Theoretical evaluation of twisting . . . 126
  6.3.2 Simulation and analysis results . . . 128
Summary . . . 132

7 Application of the approximation method 133
7.1 Memory simulation model . . . 134
  7.1.1 Simulated DRAM circuits . . . 134
  7.1.2 Model reduction techniques . . . 135
7.2 Classification of defects . . . 138
  7.2.1 Relations between injected defects . . . 139
  7.2.2 Definition of defects . . . 140
7.3 Application of 1D analysis . . . 146
7.4 Application of 2D analysis . . . 153
Summary . . . 160

8 Space of DRAM tests 161
8.1 Taxonomy of DRAM faults . . . 161
  8.1.1 Theoretical fault space . . . 161
  8.1.2 Realistic fault space . . . 165
8.2 DRAM-specific tests . . . 166
  8.2.1 Detecting hard faults . . . 167
  8.2.2 Detecting transient faults . . . 171
  8.2.3 Detecting soft faults . . . 176
8.3 Customizing march tests . . . 181
  8.3.1 Memory design considerations . . . 181
  8.3.2 Memory layout implications . . . 188
Summary . . . 191

9 Case study: the strap problem 193
9.1 Definition of strap problem . . . 194
  9.1.1 The defect . . . 194
  9.1.2 Simulation model . . . 195
  9.1.3 Process variations . . . 196
9.2 Analysis methodology . . . 198

9.2.1 Simulated sequences . . . 199
9.2.2 Background 00 . . . 200
9.2.3 Backgrounds 10, 11 and 01 . . . 203
9.3 Results and tests . . . 205
9.3.1 Summary of results . . . 205
9.3.2 Test generation . . . 207
9.4 Optimizing test stresses . . . 210
9.4.1 Optimizing t_{cyc} . . . 211
9.4.2 Optimizing temperature . . . 215
Summary . . . 219

10 Conclusions and recommendations 221
10.1 Conclusions . . . 221
10.2 Contributions . . . 226
10.3 Recommendations . . . 228
Summary . . . 230

A List of symbols 231

B List of abbreviations 235

Bibliography 239

Index 247

List of publications 255

Curriculum vitae 259

Statements 261

Stellingen (Statements in Dutch) 263

Contents of this chapter

1.1 Semiconductor memories
1.2 Memory testing
1.3 Contribution of the project
1.4 Outline of the thesis

1

Introduction

Since the conception of the first electronic systems, memory components have always fulfilled the vital task of storing and retrieving system and user-specific data. Depending on the available technology and design requirements, memories have taken many different forms, ranging from the slow and bulky punch card storage prevalent in the early days of the computer industry to the fast and highly integrated semiconductor memories commonly found in today's leading-edge computer systems.

The fast development of memory devices and the strong market competition have increased the standards of these produced devices. The increased demand on reliability has, in turn, stressed the importance of failure analysis and memory testing techniques. More and more effort is being dedicated to the study of the faulty behavior of memory devices than ever before. This thesis describes one such study, made as a joint project between Infineon Technologies and the Delft University of Technology to study the faulty behavior of a number of memory designs, by utilizing electrical (or Spice) simulation models prepared during the design process to evaluate the faulty behavior of the memory.

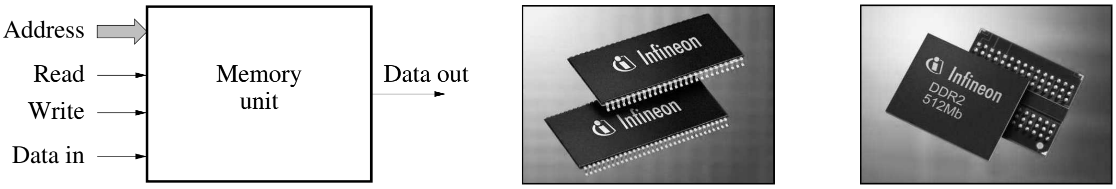
This chapter gives an introduction to the topic of memory devices, the way they are tested, and to the Ph.D. project described in this thesis. Section 1.1 starts with an overview of semiconductor memory devices, presenting the different types of these devices along with the advantages and the disadvantages of each. Then, Section 1.2 discusses the test strategies used to test memories in the different stages of their production. Section 1.3 presents the contributions of this Ph.D. project to scientific knowledge and to industrial practices. Finally, an overview of the different chapters of this thesis is given in Section 1.4.

1.1 Semiconductor memories

In their relatively short 30-year history, semiconductor memories have come from being an unknown and somewhat unproven technology to one of the most important memory technologies around. They provide an optimum combination of performance, price and density that gives them a central position in the computer industry. This section gives an overview of semiconductor memories, their different types and characteristics.

1.1.1 Definition and physical form

In very simple terms, a memory is a device used to store and retrieve information. Figure 1.1(a) shows a generic block diagram of a memory with the most basic *input/output* (*I/O*) interface possible, consisting of four inputs and one output. The address input identifies the memory cell of interest, on which memory operations are to be performed. The read input line signals a read operation performed to forward the data in the addressed cell to the data-out line. The write input line signals a write operation performed to store the data present on the data-in line into the addressed cell. Note that address input is represented in the figure as a wide arrow, which represents a so-called *address bus*, where a number of address lines are collected together into one single arrow. The terms “memory”, “memory component”, “memory device”, and “memory part” all have the same meaning and are usually used interchangeably in the literature.



(a) Block diagram (b) TSOP package (c) TBGA package

Figure 1.1. (a) Memory block diagram, (b) Infineon DRAM chip in TSOP, and (c) in TBGA package (source: Infineon Technologies).

One of the most well-known memory components in the industry is the **dynamic random access memory** (**DRAM**), primarily used as the main memory in *personal computers* (*PCs*), workstations and mainframes. The amount of DRAM found in a PC has a significant impact on its overall performance, and is considered an important parameter in PC specifications. DRAMs are usually produced in the form of an *integrated circuit* (*IC*) chip, which might be packaged in many different ways, depending on the specific application and the performance of the DRAM. Figure 1.1(b) shows an example of an Infineon memory packaged in a *TSOP* (*thin small outline package*), which is characterized by small size and low cost. This is a very common DRAM package suitable for memories of almost any size and density. It has an elongated rectangular shape with pins located on two sides of the package.

Figure 1.1(c) shows a 512 Mb (mega bit) *DDR2* (*double data rate 2*) Infineon DRAM chip, packaged in a special high performance *TBGA* (*tape ball grid array*) package. It has a square shape with pins arranged in the form of a ball grid, located on the bottom surface of the chip. TBGA is a more expensive package to manufacture and to incorporate into a system, but it enables better performance and higher clock rates than TSOP by reducing internal die-to-pin path lengths and pin impedance. It also enables a much higher pin count, since it uses the whole bottom surface area of the package instead of only two sides of it.

PC consumers are much more familiar with a *DRAM module* (also called DRAM card or DRAM stick) than with a single DRAM chip. A module basically consists of a *printed circuit board* (*PCB*) where a number of DRAM chips are connected together and made ready to be plugged into the system bus of the PC. Figure 1.2 gives an example of a 512 MB (mega byte) DDR2 Infineon DRAM module, where 8 TSOP 512 Mb chips of the type shown in Figure 1.1(b) are placed side by side and connected together on the PCB. There are two main types of DRAM modules: *single in-line memory modules* (*SIMMs*), and *dual in-line memory modules* (*DIMMs*), both of which have the same form shown in Figure 1.2. The main difference between the two lies in the fact that the data bus on a DIMM is twice as wide as that on a SIMM. The data bus width for a SIMM can range from 32 bits in small PCs up to 200 bits in a workstation, while twice this bus width applies for a DIMM [Prince99].

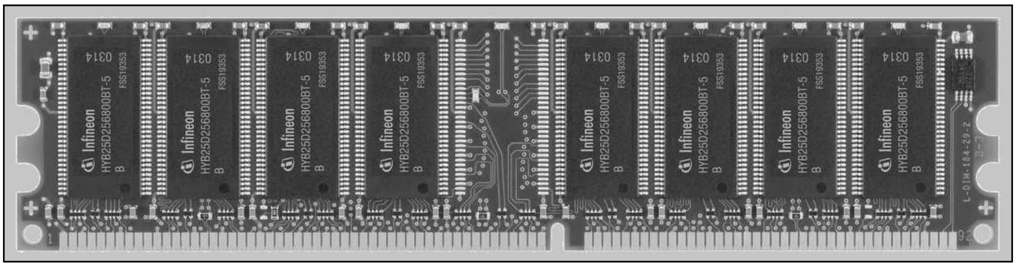


Figure 1.2. Infineon 512 MB DRAM module with chips in TSOP packages (source: Infineon Technologies).

1.1.2 Types of semiconductor memories

Figure 1.3 shows a common classification of the most important types of semiconductor memories, where two main classes are identified: **random access memories** (**RAMs**), and **read-only memories** (**ROMs**). The name RAM stands for a memory device in which cells may be accessed at random to perform a read or a write operation. Depending on the internal architecture and the actual memory cell structure, RAMs may be further divided into:

- dynamic RAMs (DRAMs), and

- static RAMs (SRAMs).

The name ROM, on the other hand, stands for a memory that can only be written a limited number of times but can be read indefinitely. The most important types of ROM are:

- programmable ROM (PROM),
- erasable PROM (EPROM),
- electrically erasable PROM (EEPROM), and
- flash memory.

As the manufacturing processes of ROMs improve, and device performance and number of possible write operations increase, the sharp boundary between RAM and ROM is gradually eroding [Waser03]. In the following, we will provide a general description of these architectures along with their advantages and disadvantages.

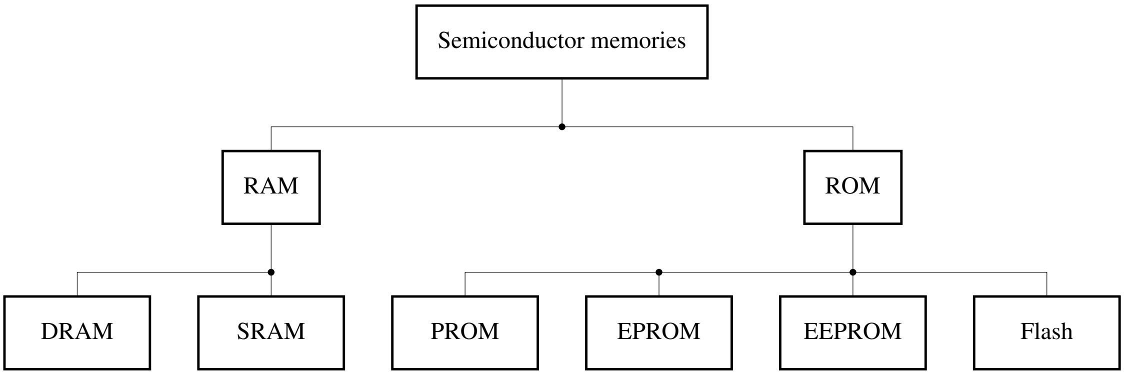


Figure 1.3. Classification of semiconductor memories.

RAM devices

A simple block diagram of a RAM is given in Figure 1.4(a). Three main inputs are shown: a read/write ($\mathrm{R}/\overline{\mathrm{W}}$) switch to signal the type of operation performed, an address input which identifies the cell to be accessed, and a data input line that supplies the data to be written in case of a write operation. A RAM also has a data output line to be used on a read operation to forward data from the addressed cell to the outside world. In principle, both DRAMs and SRAMs share this same general interface, but the specific implementation is different and mainly depends on the targeted application of the memory.

The electrical structure of an SRAM cell is shown in Figure 1.5(a). The cell is constructed using six transistors, four of which are of one transistor type (NMOS) while the other two are of another type (PMOS). The *word line* (*WL*) in the figure

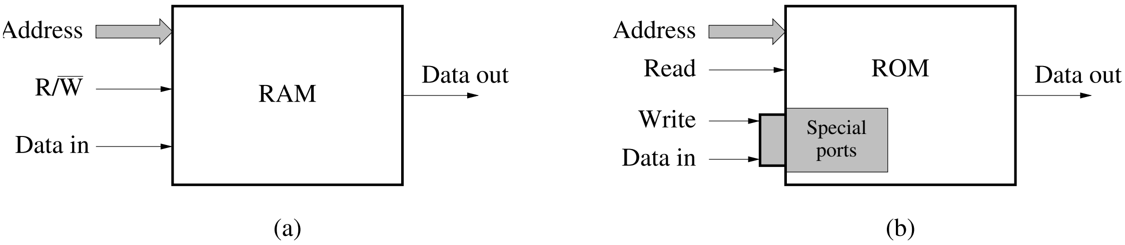


Figure 1.4. Block diagrams of (a) RAMs and (b) ROMs.

performs the address selection function for the cell. The *true bit line* (BT) and *complement bit line* (BC) serve both as the data input line and the data output line for the cell at the same time. The selection between performing a read or a write operation is fulfilled by other memory parts external to the cell. The operation of the cell in the figure is based on the fact that SRAM cells are bistable electrical elements (i.e., circuits that have two stable states). Each state is used to represent a given logical level. Once a cell is forced into one of the two states, it will stay in it as long as the memory is connected to the power supply; the name “static RAM” refers to this property.

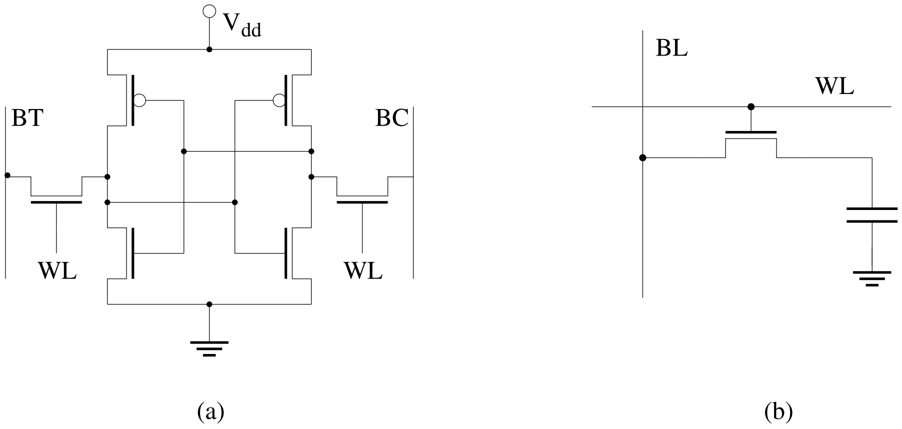


Figure 1.5. Electrical structure of (a) SRAM and (b) DRAM core cells.

The electrical structure of a DRAM cell is shown in Figure 1.5(b). The cell is constructed using one transistor and one capacitor. The WL performs the address selection, while the *bit lines* (BLs) are used as both the data input and data output lines. The selection between read and write operations is performed by other parts of the memory. As seen in the figure, DRAMs are constructed of simple capacitive elements that store electrical charges to represent a given logical level. Inherently, DRAM cells suffer from gradual charge loss, as a result of a phenomenon known as *transistor leakage currents*, which cause a cell to lose its charge gradually. In order to help cells keep their state, it is necessary for DRAMs to rewrite, or *refresh*, the already stored data bits from time to time before the cells lose their charge completely. The name "dynamic RAM" refers to the fact that the data stored in the DRAM cell may change spontaneously after a given period of time.

Both DRAMs and SRAMs are called **volatile memories** because they can only keep their data content if they stay connected to the power supply. A closer look at the two RAM structures reveals that SRAMs store their data actively by pulling their nodes to high or low voltage levels, while DRAMs store their data in capacitive elements that take time to charge up and discharge. Therefore, SRAMs have a much higher performance than DRAMs, and this is the reason they are used as the first level memory (or **cache memory**) directly supporting the *central processing unit* (*CPU*) in a microprocessor. Among other things, the main advantage that DRAMs have over SRAMs is in their density. Figure 1.5 clearly shows that DRAM cells are simple, compact elements that achieve much higher chip densities than their SRAM counterparts, which also makes them much cheaper. This cost difference is so important that it outweighs all other aspects in most applications [Al-Ars99].

ROM devices

The other architectural variation of semiconductor memories is the ROM, which is shown in Figure 1.4(b). ROMs are preprogrammed memory devices that permanently store information needed by the microprocessor for basic operation, information such as routines to interact with discs, keyboards and the display [Prince91]. The figure indicates that the write and data input lines are applied by special means, in accordance with the read-only functionality of ROMs. The most basic type of ROM, called **masked ROM** or simply ROM, can only be written once at the time of manufacturing, after which stored data cannot be changed. ROMs are non-volatile and, therefore, keep their stored data even when power is turned off.

PROM One variant of the ROM is the *programmable ROM*, or *PROM*, which is delivered unprogrammed to the consumer, who is then able to program it by writing an application-specific set of data only once. The stored information cannot be erased afterwards, a property that puts a limitation on the reusability of the PROM if the application for which the memory is used changes.

EPROM To tackle the limited reusability problem of PROMs, the *erasable PROM* (*EPROM*) was introduced. Once programmed, the EPROM acts like a normal ROM. However, if the need arises, the user is capable of erasing the contents of the EPROM and can reprogram it again. The disadvantage of the EPROM is that it cannot be reprogrammed while residing in the system; it must be removed from the system and erased with ultraviolet (UV) light first, and then reprogrammed using special equipment. Other disadvantages of the EPROM include a lower performance than ROMs, sensitivity to light, and expensive packaging with small quartz windows.

EEPROM The disadvantages of the EPROM have led to the introduction of the *electrically erasable programmable ROM* (*EEPROM*). The EEPROM can be electrically reprogrammed in the system, which eliminates the need of removing it, erasing it with UV light and reprogramming it using special equipment. EEPROMs have cheap packaging and are not sensitive to light. The main disadvantage here is the greater cell complexity as compared to the EPROM, and consequently the higher price. EEPROMs are only economically justified when non-volatility and in-system re-programmability are required.

Flash memory One variant of the EEPROM is the so-called flash memory, which combines the best features of the memory devices described thus far. Flash memories have high density, low cost, are non-volatile, fast (to read, but not to write), and electrically reprogrammable. These advantages are clearly overwhelming and, as a result, the market share of flash memory has increased dramatically in the past few years, especially in embedded systems and for mobile applications. From a software point of view, flash and EEPROM technologies are very similar. The major difference between them is that flash devices can only be erased in chunks, one sector at a time, and not on a byte-by-byte basis. Typical erasable sector sizes are in the range of 256 bytes to 16 KB. Despite this disadvantage, the flash memory combines an unbeatable set of advantages and is, therefore, becoming much more popular than the EEPROM. Flash is rapidly displacing other ROM devices as well, and in an increasing range of applications.

To summarize the discussion above, Table 1.1 lists the features of each type of semiconductor memory, and tries to quantify some aspects of memory operation. It is clear from the table that each memory type has its own strengths and weaknesses, which means that no one type can replace all others, but that each memory suits its own specific application [IBM02, Infineon04, Itoh01].

Table 1.1. Summary of the characteristics of the different memory architectures.

Criterion	SRAM	DRAM	ROM	EPROM	EEPROM	Flash
Relative cell size	4–6	1.5	1	1.5	3–4	1.5
Volatility	yes	yes	no	no	no	no
Data retention	∞	64 ms	∞	10 years	10 years	10 years
In-system re-programmability	yes	yes	no	no	yes	yes
In-system read speed	2.5 ns	55 ns	90 ns	90 ns	200 ns	60 ns
In-system write speed	2.5 ns	55 ns	—	—	2.5 s	6.4 μs
Number of writes	∞	∞	1	100	10^4–10^5	10^4–10^5

1.2 Memory testing

The exponential increase in the integration density of memory components and the associated increase in the complexity of memory faulty behavior have made fault analysis and memory testing a significantly important, yet a difficult task to perform. This section serves as a global introduction to the topic of memory testing and the way it is usually performed in the industry.

1.2.1 Definition of memory testing

Memory testing means different things to different people. There are three major parties involved in the supply chain of memory devices, and each needs to test the memory in a different way:

- the memory chip manufacturer,
- the system integrator, and
- the end user.

Figure 1.6 graphically represents these three parties involved in the supply chain and the way they interact. A *memory chip manufacturer* is the party involved in defining the specifications of memory devices and subsequently designing and manufacturing raw memory devices of the form shown in Figure 1.1(b) and (c). The *system integrator* is the party that buys memory devices from a memory manufacturer and implements them into a system intended to solve a specific customer need, such as PCs, workstations or networking equipment. The *end user* is the party that acquires the equipment provided by the system integrator in order to deploy it for solving a specific problem.

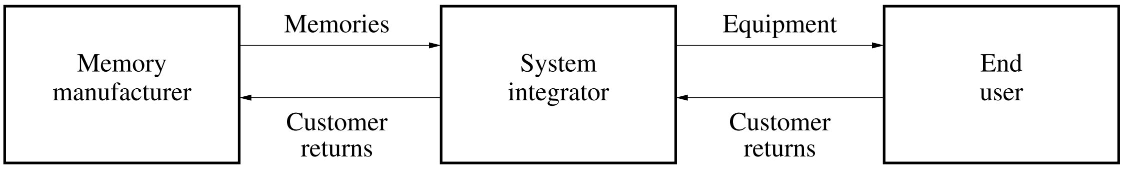


Figure 1.6. Supply chain of memory devices.

The big burden of extensive testing and qualification of memory devices rests squarely on the shoulders of the memory chip manufacturer, the first party in the memory supply chain. The system integrator may perform simple tests on purchased memory devices to screen out defective parts before incorporating them into bigger systems. But this practice is gradually disappearing, as many companies expect the devices to be delivered just in time for use, and with very high quality levels. System integrators *are* expected, however, to test *their* systems extensively before delivery to the end user. During this testing process, if memory devices are systematically found to cause equipment failure, these defective memories are sent back to the manufacturer in the form of **customer returns**. The manufacturer is then expected to investigate these returns and to try to screen them out before they are sold to the system integrator. Finally, the end user is not expected to perform any specific testing on acquired systems other than setting them up for regular operation. Here too, the end user sends back defective systems to the integrator in the form of customer returns for reparation or replacement.

The fact that memory testing should mainly, and almost exclusively, be performed by memory manufacturers is due to the exponential increase in the cost of detecting a defective component after it gets incorporated into increasingly more complex systems, from chip to module to system. A well-known industrial rule of thumb (sometimes referred to as the "rule of tens") on the relative cost of system testing states that *at each successive stage in production, it cost ten times as much to identify and repair a defective component than it would have cost at a previous production stage* [Jha03]. Figure 1.7 gives a graphical representation of this rule.

$$\text{Cost}_{\text{stage}(i+1)} = 10 \cdot \text{Cost}_{\text{stage}(i)} \tag{1.1}$$

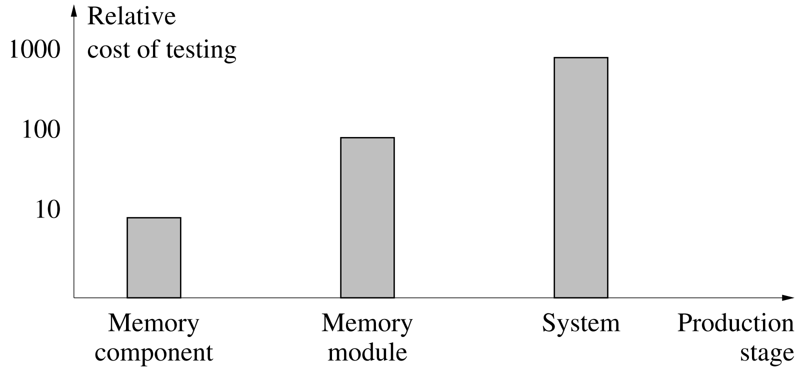


Figure 1.7. Relative cost of component testing at each stage in production.

This rule means that memory manufacturers have to elaborately test their memory chips in order to ensure the very high quality requirements expected from ICs in general and from memories in particular. Currently, manufacturers of high quality memories, such as those in the automotive industry, should supply chips with failure rates as low as 10 *ppm* (*parts per million*) to their customers [Majhi05].

Since memory testing is mainly performed by memory manufacturers, this thesis is only concerned with manufacturing tests, as it is the field where the biggest investment in testing is incurred, and where the highest possibility of payback on research is expected.

1.2.2 Manufacturing test flow

A **test flow** is a description of the stages needed to test a specific device, along with the activities needed to set up and adapt these stages so that the device can be tested successfully. Different memory components require different test flows that fit the specific needs of each component. Despite the differences, all memories have lots of similarities and share the same general test flow structure.

Figure 1.8 shows a simplified block diagram of a typical manufacturing test flow performed by a memory manufacturer [compare with Figure 1.6]. The figure also shows a three-stage representation of the design flow, starting with the specification of a new memory technology, followed by the design process and culminating with the actual chip manufacturing process [Nakamae03, Antonin91]. The design flow is included here for completeness, and to underscore the important interaction between the design and test flow. Within the test flow, the actual testing process takes place in only two blocks in the figure

- frontend (or wafer level) testing, and
- backend (or component level) testing,

drawn as two large rectangles. Frontend testing is performed before chip packaging so that only functional chips get to the packaging process, in order to reduce the costs of packaging. Backend testing ensures that packaged chips function properly before delivery to the customer. More details about the activities in these two test stages are given in the next section.

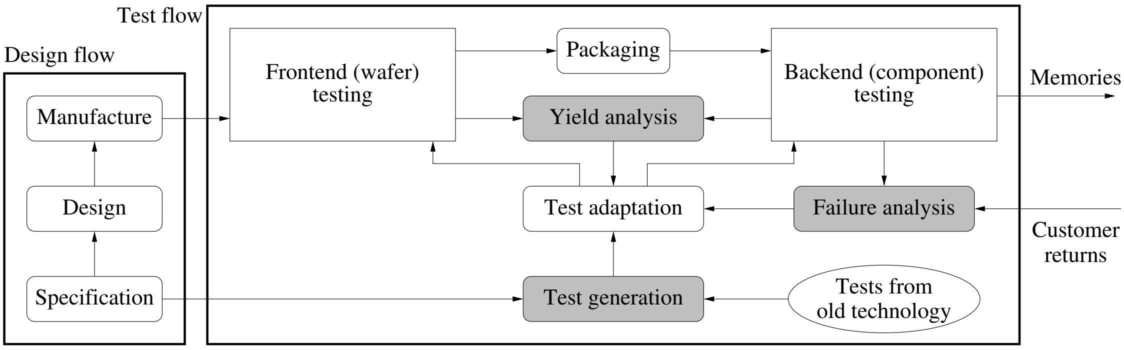


Figure 1.8. Block diagram of a manufacturing test flow for a memory manufacturer.

Figure 1.8 shows that in order to set up and maintain a proper testing process, three main activities need to be carried out (identified as three shaded blocks in the figure):

- test generation,
- yield analysis, and
- failure analysis.

Test generation For every new memory technology, an initial set of tests is needed to test the first manufactured components. This initial set is generated by adapting an existing set of tests used for the previous memory technology, so that the tests would fit the specifications of the new memory technology. Generating this initial set is the first step in setting up the memory testing process, and it starts even before the design is complete and before *first silicon* (a term used in the industry for the first wafers produced for the new technology) is manufactured.

Yield analysis *Yield* (Y) is defined as the fraction of functional memories relative to the total number of produced memories by a manufacturing process.

$$Y = \frac{\text{number of functional memories}}{\text{total number of memories}} \tag{1.2}$$

A successful manufacturing process is expected to result in high yield values. However, the actual manufacturing yield depends heavily on the quality of the testing process. A very stringent test process results in eliminating a large number of chips and allows only memories with the highest quality to pass to the customer, while a less restrictive test process allows a large number of high and low quality components to pass to the customer. Depending on the required memory quality and the effectiveness of the manufacturing process, a target yield is identified and tests should be modified to match it. For example, memories produced for mission-critical applications (such as life-support systems) demand very stringent test requirements, for which a lower yield (and a corresponding increase in price) is tolerated.

Failure analysis The ultimate arbitrator of a product's success is customer satisfaction. As long as the customer buys produced components and does not complain about equipment failures as a result of purchased memories, then the total manufacturing and testing process is considered to be functioning properly. But once the customer expresses dissatisfaction, and customer returns start coming back to the memory manufacturer, then failure analysis must be performed immediately on these customer returns to identify the defect causing the failure. Subsequently, the test process must be modified as soon as possible to prevent this type of defective devices from being shipped to the customer.

The next section discusses in more detail the actual manufacturing test process, which is performed within the frontend and backend stages of the test flow.

1.2.3 Frontend and backend testing

The frontend and backend test stages are the cornerstones of any memory test process. They are the test flow stages where the tests are actually performed on the memory, and the correctness of the memory behavior is inspected [Falter00].

Although the frontend and backend test stages are part of the test flow of most memory devices, the exact sequence of test steps (also called **test insertions**) that take place in each of these two stages vary significantly depending of the specific type of memory being tested. Still, the test process needed for DRAMs and SRAMs have a number of shared characteristics, and can be divided into four main test insertions [see Figure 1.9]:

- prefuse testing,
- postfuse testing,
- burn-in testing, and
- package testing.

In the following, these four test insertions are discussed, after which some of the differences in the test process of DRAMs and SRAMs are identified.

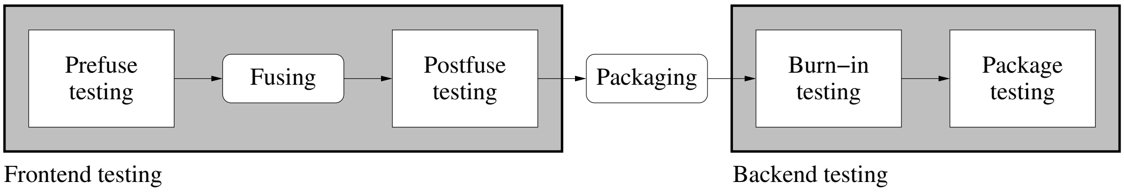


Figure 1.9. Typical test process of DRAM and SRAM devices.

Prefuse testing This is the first test insertion of frontend testing in the memory test process, with the main objective of identifying the failing cells and repairing them using redundant memory elements. In addition, the tests used here should be able to indicate the type of failure mechanism at play in case a failure does take place. This information is important as feedback to the manufacturing facility (**fab**) to modify the fabrication process, and to eliminate the failure in order to increase yield. One important characteristic of prefuse testing is that it is performed on-wafer, using tiny needles (or probes) that are unable to conduct large amounts of current, thereby limiting the speed at which tests are performed in this test insertion. In other words, the high-speed operation of the memory cannot be inspected by prefuse tests. Such speed tests can only be performed after memory packaging, in the package testing stage, the final test insertion in the test flow [see package testing below].

Postfuse testing This is the second test insertion of frontend testing in the memory test process, with the main objective of checking that redundancy fusing and repair has been done in a proper way. Tests performed in this stage are rather simple and short sequences of write and read operations (so-called *scan tests*) used to check the basic functionality of repaired memory elements.

Burn-in testing This is the first test insertion of backend testing in the memory test process, and it takes place after wafer dicing and chip packaging. Burn-in is a well-known method to check the reliability of manufactured components by applying highly stressful operational conditions, called *test stresses* (such as high voltages, possibly combined with high temperature), which accelerate the aging process of the memory [vdGoor98]. Two different types of tests are done in this stage: those used merely to accelerate the aging process itself, and others used to check for proper functionality under test stresses. Tests performed to stimulate the aging process are simple scan tests or *hammer tests* (also called *disturb tests*, where a specific operation is repeated a multiple number of times on the same cell). The pass/fail information provided by these tests are generally neglected since they are not meant to test the actual functionality of the memory operation. The second type of tests is meant to ensure proper memory operation at high stresses. Most notably here are the *data retention tests*, which inspect the ability of memory cells to keep their data.

Package testing This is the second test insertion of backend testing and the last test insertion in the memory test process, with the main objective of validating the operation of the memory according to the specifications. In this test insertion, a set of memory tests is performed at high as well as at low temperature to ensure proper component functionality. Some tests performed in prefuse testing are repeated here, with a slightly lower level of stress on the memory compared to that applied during prefuse testing (using a so-called stress *guard band*). If such a test results in a relatively high fail count during package testing, then the corresponding prefuse test is not effective enough, and should be stressed further to identify failing cells during prefuse testing and to try to repair them during fusing.

The test process description above identifies the three necessary components of DRAM and SRAM tests, and the three different requirements a modern memory test should satisfy [see Figure 1.10].

- **Test components**

  1. *Operation sequence*: a test should specify a memory operation sequence of writes and reads to be performed in a specific order on memory cells.
  2. *Data pattern* (or *data background*): a test should also specify a pattern of 0s and 1s to be written into and read from accessed memory cells.
  3. *Stresses*: a memory test should include a specification of different operational conditions or stresses (such as timing, temperature and voltage) that the test is supposed to be performed at. This component is important to test for proper functionality within the parameter range defined by the specifications. Stresses are also important (sometimes necessary) to increase the coverage of a test without increasing its length.

- **Test requirements**

  1. *Fault detection*: a test that fulfills this requirement should result in a fail when applied on a memory that contains the fault. This is a basic requirement of any memory test designed to test for a specific type of faulty behavior.
  2. *Fault localization*: a test that fulfills this requirement should be able to identify the specific memory cell (or group of cells) where the fault takes place. This requirement is associated with the need to identify and repair failing cells by fusing in order to increase yield.
  3. *Fault diagnosis*: a test that fulfills this requirement should be able to indicate the physical root cause behind the observed faulty behavior. This requirement is associated with the need to give instant feedback to the fabrication process regarding possible fabrication causes of observed faults.

Figure 1.10 gives a summary of the components that DRAM and SRAM tests should have and the requirements these tests should satisfy. The three white blocks in the figure (data background, fault detection and fault localization) indicate those items that DRAM and SRAM tests have in common, while the three shaded blocks (operation sequence, stresses and fault diagnosis) indicate those items where DRAM and SRAM tests are different. In the following, the differences are discussed and the reasons for these differences are given [see summary in Table 1.2].

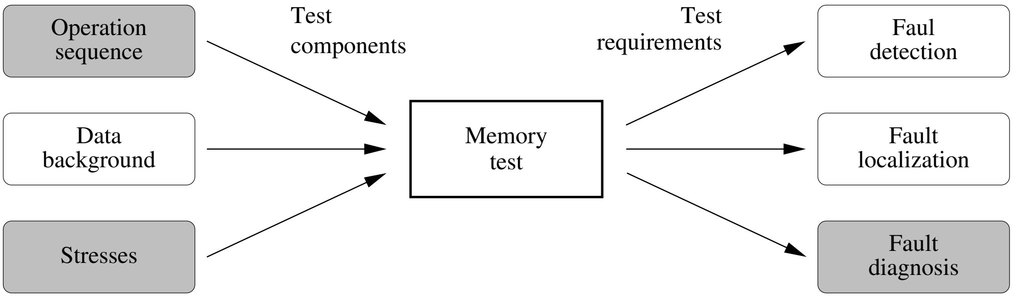


Figure 1.10. Components and requirements a memory test should satisfy.

Differences in the operation sequence SRAMs and DRAMs commonly suffer from different types of memory faults. The reason for this difference is the digital nature of SRAM cells (stored data are either logic 0 or logic 1) and the analog nature of DRAM cells (stored data can take any value within a range of voltages between 0 and 1). This aspect of DRAM operation is discussed in more detail in Chapter 4. The most common types of memory faults observed in SRAMs are the so-called *static faults*, where a fault is sensitized by performing at most one operation on the memory [Hamdioui00]. Therefore, the vast majority of SRAM tests are designed to target these static faults. Since the number of static faults is limited to 12 [see Chapter 4], each of which requires at most one operation for fault sensitization, this means that SRAM tests are rather short, limited in number, which in turn makes it feasible to perform a single test that is theoretically proven to detect all 12 static faults. In contrast, a relatively large portion of DRAM faults are the so-called *dynamic faults*, which are faults sensitized by performing two or more memory operations on the memory [see Chapter 7]. As a result, a large portion of DRAM tests are designed to target these dynamic faults. Since there is an unlimited number of dynamic faults, each of which requires a potentially large number of sensitizing operations, DRAM tests are rather complex and time consuming. Therefore, it is not possible to exhaustively analyze dynamic faults and derive general corresponding tests. The only way to derive DRAM tests is by analyzing the behavior of each DRAM design separately and deriving *device-specific* tests for each design.

Differences in stresses Almost all memory tests designed for any type of memory include some kind of stress specification to perform the test at. For SRAM devices, stresses are most commonly used to inspect for proper functionality of the memory within the parameter range defined by the specifications. As a result, the specifications give a very good indication of the types of stresses and the values to be used for each stress in SRAM tests, even before the chips are manufactured. Small stress modifications are subsequently needed to ensure a specific quality level of the tests, which is not too relaxed nor too stressful. For DRAMs, on the other hand, stresses are needed not only to ensure proper functionality within the specifications, but also to identify the strength of the stored cell voltage and the amount of leakage current in the cell. DRAM cells suffer from leakage currents which gradually degrade any stored cell voltage, and eventually result in the total loss of stored values after a specific amount of time. To test for the ability of the cell to keep its voltage for a long enough amount of time, stresses are used to limit the maximum stored voltage level and to increase the amount of leakage current, so that the tests can identify failing cells in a short amount of time. Unlike SRAM stresses, DRAM stresses have no clear a priori indicators for the types and values to be used in the test. Therefore, an educated guess is attempted at first and, depending on yield feedback, these values are modified iteratively in a slow and costly process until acceptable final values are obtained. This thesis proposes an alternative, more cost-effective approach to optimize stresses, where it is possible to use Spice simulation to get a fairly good initial prediction for the required stresses. The issue of the importance of stresses for DRAMs is discussed in more detail in Chapter 4, while the Spice-based optimization method is discussed in Chapter 5.

Differences in fault diagnosis A faulty fabrication process that systematically results in defective chips is very costly, and therefore feedback from the test process with regard to potential problems in the fabrication process is of prime importance. Therefore, it is common for many memory tests to have some kind of diagnostic capability. Tests designed with diagnostic capabilities are complex, since they should efficiently detect faults caused by a specific failure mechanism, while having minimal coverage of faults caused by other, unrelated failure mechanisms. As a result, the more complex and versatile the total faulty behavior of the memory is, the more complex each diagnostic test for a specific failure mechanism becomes, since a test must also avoid detecting faults resulting from other failure mechanisms. Usually, DRAMs lead the semiconductor industry by using the latest cutting-edge fabrication processes with the newest developments, where newly introduced failure mechanisms should be identified and dealt with. In addition, DRAMs need special, unconventional on-chip structures (such as trench or stack capacitors) that are difficult to manufacture and result in special, unconventional types of faults. SRAMs, on the other hand, are usually manufactured using an established manufacturing process, where the most common failure mechanisms are already known, and they mainly require conventional on-chip structures (transistors), similar to those commonly used for logic devices. As a consequence, DRAM test development is a gradual and slow process, where tests should be developed based on a close observation of the fabrication process, and where each test has a special relation to the fabrication process, to other tests in the test flow, and to the specific memory design being tested. This is a far cry from the world of SRAMs, where devices can be designed as macros and sold in the form of *intellectual property* (*IP*) structures, without having to care about the specifics of the fabrication process to be used.

1.3 Contribution of the project

The research performed in the course of this project has contributed in a number of ways to the development of a new methodology in DRAM testing on an industrial scale [Al-Ars05]. We start this section by identifying the global industrial framework of the contribution, followed by a detailed discussion of each of its aspects.

1.3.1 Framework of the contribution

The high cost and complexity of the test process of DRAM devices calls for introducing new innovative ways to tackle the test development problem. As discussed in Section 1.2 and shown in Figure 1.11, the tests performed in the frontend and backend stages of the test flow are traditionally generated in two steps: 1. based on information from the specification stage of the design flow and, 2. based on feedback information from test application after the manufacture stage of the design flow. These two stages, shown shaded in Figure 1.11, are described next.

1. **Specification stage**—Every test flow for a new memory technology starts

Table 1.2. Differences in the attributes of memory tests used for DRAMs and SRAMs.

Attribute	Difference	Cause	Consequences
Operation sequence	**DRAMs:** long (possibly infinite) sequences **SRAMs:** short sequences (one operation)	**DRAMs:** analog cell operation **SRAMs:** mostly digital cell operation	**DRAMs:** custom-made tests for each design **SRAMs:** theoretically derived tests for all designs
Stresses	**DRAMs:** necessary to test functionality within specifications, *and* to identify strength of voltage level and amount of leakage current **SRAMs:** mainly needed to test functionality within specifications	**DRAMs:** gradual degradation of stored voltage level due to inherent leakage current **SRAMs:** stored voltage level is fixed and leakage current is compensated for	**DRAMs:** needed types and values of stresses are identified using a time-consuming iterative process **SRAMs:** needed types and values of stresses are close to specifications and are easier to identify
Fault diagnosis	**DRAMs:** necessary for immediate feedback to fabrication process **SRAMs:** not needed for a robust fabrication process, minimal feedback will suffice	**DRAMs:** new, unknown technology usually used and difficult structures fabricated **SRAMs:** existing, tested technology usually used and typical transistors fabricated	**DRAMs:** more complex tests with detection, localization *and* diagnosis abilities **SRAMs:** less complex tests with mainly detection and localization abilities

with an analysis of the specifications of the new memory and an analysis of the tests used for the previous memory technology [see the block "Tests from old technology" in Figure 1.11]. The specifications are used to generate a new set of tests by adapting the old tests to comply with specifications of the new memory [see the block "Test generation"].

2. **Manufacture stage**—After manufacturing the chips, the specifications-based tests generated by the previous step are applied to the memory in the front-end and backend stages of the test flow. For every new memory technology, the yield analysis stage [see block "Yield analysis"] identifies new, previously unknown, failure mechanisms for which additional (or modified) tests are needed. Furthermore, it is commonplace to have subtle fails that can only be identified after chips fail with the customer, who sends them back to the manufacturer as *customer returns*, where they get analyzed in the failure analysis stage of the test flow [see block "Failure analysis"]. The feedback from the yield analysis and the failure analysis stages is used to adapt the tests in the test adaptation stage [see block "Test adaptation"], so that a set of tests can

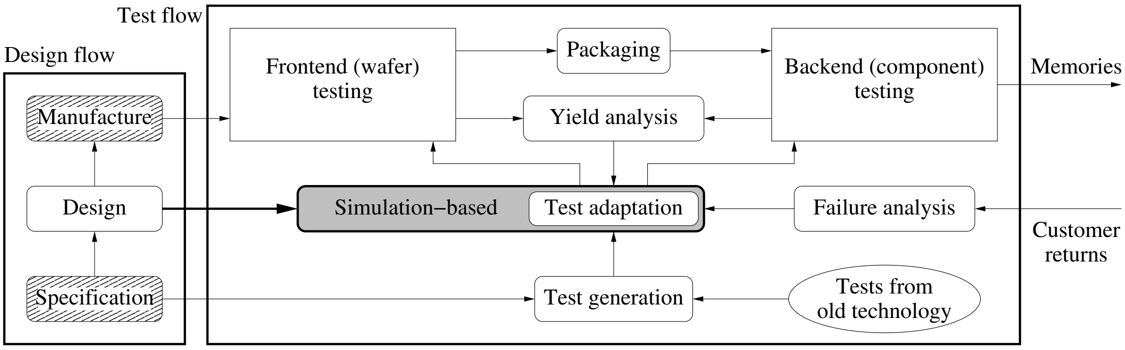


Figure 1.11. Manufacturing test flow, where the contribution of this thesis is shaded.

be attained that is more capable of detecting the previously unknown types of faulty behavior.

Therefore, a test set is developed in two steps: first using information from the specification stage and then adapting it based on information from the manufacture stage of the design flow. Each one of these two steps in test generation has advantages and disadvantages in terms of quality and cost, as shown in Figure 1.12.

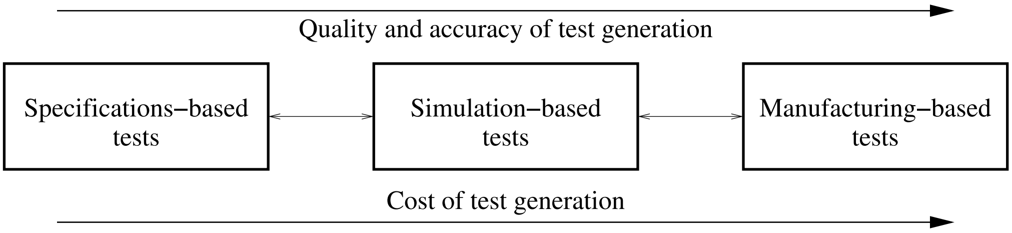

Quality and accuracy of test generation
Specifications–based tests
Simulation–based tests
Manufacturing–based tests
Cost of test generation

Figure 1.12. Cost-quality tradeoff of test generation.

Specifications-based test development Tests generated based on the specifications of the memory have a relatively low cost since they can be derived analytically by adapting already known tests of previous memory designs. This style of test generation assumes that the faulty behavior of the memory does not change with changes in memory design and technology. This assumption might be valid for generic types of failure mechanisms, but for every new memory technology there are always some types of failure mechanisms that are specific to the memory under analysis. Therefore, specifications-based test generation, though relatively cheap, is not memory-specific enough to derive accurate tests for the memory under analysis.

Manufacturing-based test development Tests generated based on feedback from the manufacturing process are very accurate in describing the faulty behavior of the memory under analysis, since the tests are generated by statistically analyzing feedback data from the manufacturing facility (fab) as tests are performed on real memory chips. This style of test generation assumes that there is a large volume of failing memory parts to enable a meaningful statistical analysis such that reliable tests can be derived. This assumption requires a rather expensive and time-consuming *test adaptation loop* of test application, yield analysis and test adaptation until a stable and a reliable set of tests can be generated. In Figure 1.11, two test adaptation loops are shown: 1. Frontend → Yield analysis → Test adaptation, and 2. Backend → Yield analysis → Test adaptation. If such expensive test adaptation loops are not properly implemented, the memory manufacturer runs the risk of delivering defective parts to the customer, which will most probably be sent back to the manufacturer in the form of defective customer returns, a situation that is bound to reduce customer satisfaction levels with the product. In conclusion, manufacture-based testing, though accurate and memory specific, has a high price tag associated with it, making it an expensive alternative for test generation.

Thesis contribution: simulation-based test development This thesis proposes a *simulation-based test generation approach*, which strikes a tradeoff between specifications and manufacturing-based test generation, and provides an alternative that is both moderately cheap as well as device specific. As shown in the shaded "Simulation-based" block of Figure 1.11, simulation-based test generation uses information from the design stage of the design flow, where an electrical Spice model of the memory is generated to evaluate the expected memory behavior. The Spice model of the memory represents its internal design and behavior, in addition to an electrical description of the fabrication process to be used to manufacture the memory. This provides a fairly accurate representation of the specific behavior of the memory under analysis. At the same time, simulations can be directly related to a specific failure mechanism and, therefore, they can greatly accelerate the expensive and time-consuming feedback loop needed by manufacturing-based test generation. Furthermore, simulation-based test generation provides the following added advantages:

- It increases our understanding of the internal faulty behavior of the memory.
- It supports test adaptation activities after yield analysis of fab tests.
- It supports failure analysis activities for customer returns.
- It enables early identification of *design* bugs before tape out, which is the design stage where the design data is stored on tape and transported to the fab.

1.3.2 Details of the contribution

In order to materialize the vision of simulation-based test development for DRAMs, a number of theoretical (scientific) as well as practical (industrial) hurdles have been tackled and solved in this thesis. It is worth noting here that this vision has been *exclusively and completely* carried out within the framework of this project, since there is no previously published work on employing electrical simulation for DRAM test development. First, the scientific contributions are listed, followed by the industrial contributions.

Scientific contributions

- The definition of a general space of memory faults in combination with a taxonomy that describes any possible faulty behavior exhibited by the memory [Al-Ars99, Al-Ars00, Al-Ars01a, Al-Ars03a, vdGoor00]. This general space is treated in Chapter 4.

- The identification of the specific fault classes needed to describe the faulty behavior of DRAMs, such that the rather complex general space of memory faults is reduced to a smaller, manageable size [Al-Ars01b, Al-Ars02a, Al-Ars04a]. These DRAM-specific fault classes are treated in Chapter 4.

- The inclusion of stresses (voltage, timing and temperature) as a theoretically fundamental part of memory testing, and devising a language to model it [Al-Ars01c, Al-Ars01d].

- The analysis of interactions between different memory faults in a way that may limit the ability of memory tests to detect them [Al-Ars04c, Hamdioui03b, Hamdioui04b].

- The theoretical derivation of a number of memory tests to effectively test special types of faulty behavior [Hamdioui02, Hamdioui03a, Hamdioui04c, vdGoor04a, vdGoor04b].

Industrial contributions

- Solving the problem of the long simulation time needed to simulate the faulty behavior of the memory by introducing the concept of a *reduced memory model* [Al-Ars01e]. This issue is treated in Chapter 7.

- Introducing simulation-based fault analysis methods to properly interpret the faulty behavior in the simulation results and to correctly map them into memory faults [Al-Ars02b, Al-Ars02c, Al-Ars03d, Al-Ars05]. This issue is treated in Chapter 5.

- Proposing a simulation-based stress optimization method to use circuit simulation to identify the most optimal stresses for a given defect [Al-Ars03c, Al-Ars03b].

- Dealing with simulation model inaccuracy and the issue of variations in the manufacturing process to ensure a high-quality fault analysis approach [Al-Ars02d, Al-Ars02e, Al-Ars03e].

- Evaluating the influence of parasitic capacitances in the simulation model, and analyzing the effect of bit line coupling on the simulated faulty behavior for a given defect [Al-Ars04b, Al-Ars04c].

1.4 Outline of the thesis

This thesis is organized in 10 chapters of continuous material that gradually builds on the information presented as the thesis progresses. Unless one has previous knowledge of the presented topics, progression from chapter to chapter is important to gain the necessary understanding of the whole thesis.

Chapter 1 introduces the field of semiconductor memory devices, where a short description is given of a number of different types of semiconductor memories. The issue of manufacturing test development is discussed with some detail, in order to identify the position of this project in the big picture of industrial test development.

Chapter 2 gives a detailed discussion of the external behavior of DRAMs, represented by timing diagrams, as well as a description of the internal functional units a typical DRAM contains.

Chapter 3 builds the needed fundamental knowledge in transistor operation and Spice simulation to understand the heavily simulation-based fault analysis described in the thesis. This chapter also simulates the functionality of important DRAM circuits, and gives an idea of the way DRAMs are manufactured on silicon.

Chapter 4 defines the theoretical basis of fault modeling, and the concepts of fault primitives and functional fault models. In addition, new fault models are introduced that take timing into consideration (which is of particular importance to DRAMs).

Chapter 5 presents the approximate simulation-based fault analysis methods, called the one dimensional (1D) and two dimensional (2D) analysis, able to evaluate the faulty behavior of defective memory devices having one or two floating (i.e., disconnected) nodes.

Chapter 6 discusses the concepts of coupling and crosstalk between internal memory lines, and the way these influence the faulty behavior of the memory. The chapter also identifies the effects of line twisting in eliminating internal crosstalk and in modifying the resulting faulty behavior.

Chapter 7 evaluates the results of a simulation-based analysis of a number of DRAM defects causing one and two floating nodes. The simulation analysis used in this chapter is based on the approximate simulation methods presented in the previous chapter.

Chapter 8 uses the theoretical analysis of DRAM faults performed in Chapter 4 to derive the full space of DRAM tests. These tests can be used to detect all the different types of DRAM-specific faults proposed in this thesis.

The thesis culminates in Chapter 9, which discusses an industrial case study performed on a specific defect called the "open strap." The study shows how simulation-based fault analysis can be used in an industrial environment to solve real engineering problems starting from defect modeling, through test generation, and finally ending with stress optimization.

The thesis ends with Chapter 10, in which the conclusions and contributions of the thesis are listed, followed by a number of recommendations intended to strengthen the vision of the simulation-based fault analysis in the industry.

Summary

This chapter served as a simple introduction to the concepts associated with semiconductor memories and memory testing in general. The main topics discussed are as follows:

- Definition of a memory, its interface, and physical form. A number of different semiconductor memory devices have been discussed along with the most commonly used memory of all, the DRAM. The advantages and disadvantages of each memory were presented, and compared with those of other memories.
- Identification of the three basic types of memory testing: testing by the manufacturer, testing by the system integrator, and testing by the end user. It has been shown that the task of exhaustive test application and defective memory elimination rests squarely on the shoulders of the memory chip manufacturer.
- Detailed discussion of the different phases of the industrial test flow, starting from the identification of new tests for a new memory technology to the final adaptation of the tests used to detect subtle memory-specific fails.
- Description of the major differences between the DRAM and SRAM testing process. The differences are classified into three categories: those related to the operation sequence, those related to the stresses, and those related to the fault diagnosis abilities.
- Discussion of the contribution of the project and the way it fits into the big scheme of industrial memory test generation. The contribution represents a new simulation-based approach to industrial test generation and adaptation, which strikes a tradeoff between the inexpensive and inaccurate specifications-based test generation and the rather expensive and accurate manufacturing-based test generation.

Contents of this chapter

2.1 Modeling of DRAMs
2.2 Behavioral DRAM model
2.3 Functional DRAM model

2

DRAM behavior and architecture

The first DRAM was manufactured in 1970 by the then newly formed Intel company, and sold as a product named the 1103 (1 Kbit PMOS dynamic RAM IC), which by 1972 became the best selling semiconductor memory chip in the world, defeating the magnetic core memory. Since then, the structure of DRAMs has changed gradually to meet the ever increasing performance demands of the microprocessor. The DRAM is today's optimum memory for inexpensive, high density applications, and it is expected to maintain this position well into the foreseeable future. This chapter discusses in some detail the external behavior of a DRAM, represented by timing diagrams, as well as its internal structure, represented by the functional units it contains.

The chapter employs a hierarchical modeling approach to describe DRAMs in a top-down system fashion, starting from an external behavioral model down to the physical buildup of the DRAM components. Section 2.1 begins the chapter by defining the modeling approach used to represent the different modeling levels of the memory. Section 2.2 is concerned with the behavioral memory model, which treats the memory as a single black box, and defines its input/output characteristics using the concept of timing diagrams. Section 2.3 presents the functional model of the memory, which defines the memory as a collection of interconnected black boxes, each with its own set of behavioral specifications. It describes the functionality of different memory blocks and identifies the way external memory operations are performed internally. The section also defines the concept of internal DRAM commands and the way they relate to the external behavior.

2.1 Modeling of DRAMs

This chapter describes the behavior and the buildup of DRAMs using a layered modeling approach commonly used to describe modern, overly complex VLSI IC systems. The different levels of this modeling approach are shown in Figure 2.1. The lowest level shown, referred to as the layout model and represented by the largest block in the figure, is the one most closely related to the actual physical system. As we move from the layout model (lowest) toward the behavioral model (highest) in the figure, the models become less representative of the physical system and more related to the way the system behaves, or in other words, less physical and more abstract. Therefore, each modeling level in the figure is called a *level of abstraction*. In the figure, a model represented by a larger block has a lower level of abstraction than a model represented by a smaller block. A higher level of abstraction contains more explicit information about the way a system is expected to function and less about its buildup. It is possible to have a model that contains components from different levels of abstraction, an approach referred to as *mixed-level modeling*. With mixed-level modeling, one may focus on low-level details only in the area of interest in the system, while maintaining high-level models for the rest of the system. In the following, the modeling levels shown in Figure 2.1 are explained in more detail [vdGoor98].

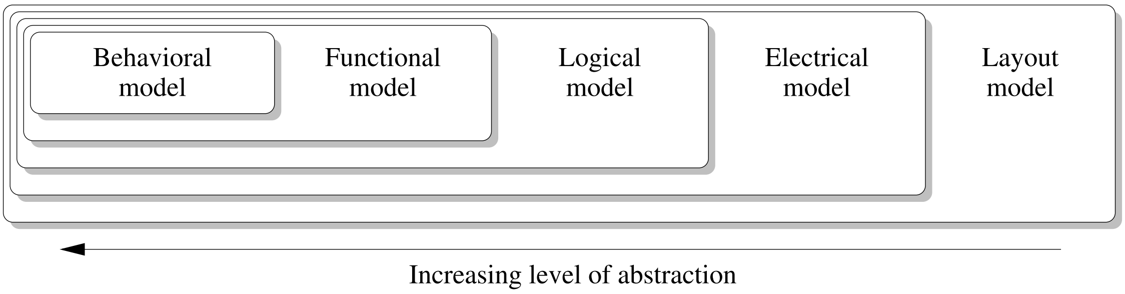


Figure 2.1. Different modeling levels usually used to represent ICs.

The behavioral model This is the highest modeling level in the figure and it is based on the specifications of the system. At this level, there is practically no information given about the internal structure of the system or possible implementations of the performed functions. The only information given is the relation between input and output signals while treating the system as a black box. A model at this level usually makes use of timing diagrams to convey information about the system behavior. In this chapter, the behavioral DRAM model is presented in Section 2.2.

The functional model This model distinguishes functions the system needs to fulfill in order to operate properly. At this level, the system is divided into several interacting subsystems each with a specific function. Each subsystem is basically a black box called a *functional block* with its own behavioral model. The collective operation of the functional blocks result in the proper operation of the system as a whole. In this chapter, the functional DRAM model is given in Section 2.3.

The logical model This model is based on the logic gate representation of the system. At this level, simple boolean relations are used to establish the desired system functionality. It is not very common to model memories exclusively using logic gates, although logic gates are often present in models of a higher or lower level of abstraction to serve special purposes. Therefore, no exclusive DRAM logical model is given in this chapter.

The electrical model This model is based on the basic electrical components that make up the system. In semiconductor memories, the components are mostly transistors, resistors and capacitors. At this level, we are not only concerned with the logical interpretation of an electrical signal but also the actual electrical value of it (its voltage level, for example). Since this thesis is primarily concerned with the electrical level simulation of DRAM circuits, this memory model is treated in depth in Section 3.2.

The layout model This is the lowest modeling level shown in the figure and the one with the most details about system structure. It is directly related to the actual physical implementation of the system. At this level, all aspects of the system are taken into consideration, even the geometrical configuration plays a role, such as the length and thickness of signal lines. For this reason, this model is also called the *geometrical model*. Section 3.3 discusses the layout representation of DRAM cells.

Taking a closer look at the behavioral and the functional models reveals that there is a strong correspondence between the two. In fact, the behavioral model can be treated as a special case of the functional model, with the condition that only one function is presented, namely the function of the system as a whole. Therefore, some authors prefer to classify both modeling schemes as special cases of a more general model called the *structural model* [Abramovici90]. The structural model describes a system as a number of interconnected functional blocks. According to this definition, a behavioral model is a structural model with only one function, while a functional model is a structural model with more than one interconnected function.

2.2 Behavioral DRAM model

The behavioral DRAM model [see Figure 2.1] is an abstract description of the external behavior of the memory, where attention is only given to the relations between input and output signals of the system, coupled with their associated timing. At this level of abstraction, nothing is supposed to be known about the internal structure or specific implementation of the system. The behavioral model contains the least amount of information any memory *datasheet* (i.e., the documentation given with the memory) must provide to describe the operation of the memory. Usually, datasheets use *timing diagrams* for this purpose, along with the minimum or maximum timing requirements for every system action.

2.2.1 DRAM block diagram

Figure 2.2 gives a general block diagram of a DRAM, where the input and output lines are shown [compare with Figure 1.1]. The shown memory has a clock signal as input, which means that memory operations are synchronized with the functionality of the system using a single system clock. These types of memories are referred to as *synchronous DRAMs* (or *SDRAMs*), which make up the vast majority of DRAMs manufactured today.

To access a given memory cell, the DRAM has a number of address lines, collected together in the figure and represented as a single wide arrow called the **address bus**. The memory also has a $R/\overline{W}$ signal to identify the type of operation being performed. Furthermore, the memory has a **data bus** to exchange data, which means that data transfer is not necessarily accomplished by exchanging a single bit of information, but multiple cells may be connected in parallel and given a single address to increase the *data transfer rate* of the memory. The data transferred on the data bus or, in other words, the block of data the memory is able to exchange, is called the memory *data word* or simply *word*. If the data bus has only one line, then the data word is called a *bit* and is abbreviated as “b”, while if the data bus has 8 lines, the data word is called a *byte* and is abbreviated as “B”.

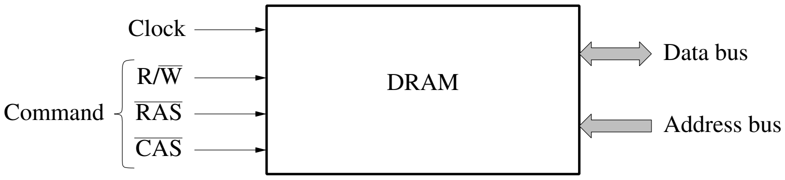


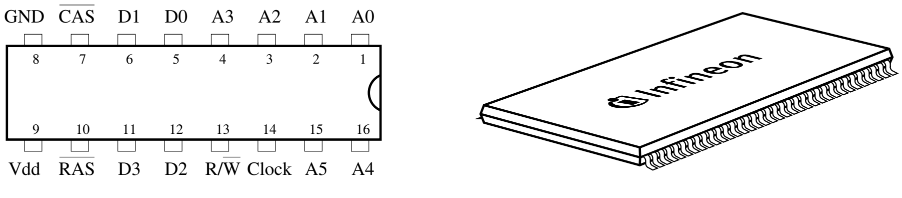
Figure 2.2. Block diagram of a DRAM.

The double-sided arrow of the data bus in the figure indicates that data-in and data-out signals share the same signal lines using a strategy called **multiplexing**. Multiplexing is a technique commonly used in DRAMs to halve the number of pins needed externally, thereby reducing the package cost. There are two main types of multiplexing commonly used in DRAMs: *data bus multiplexing* and *address bus multiplexing.*

Data bus multiplexing is used in most DRAMs nowadays due to the high number of external lines needed in modern high capacity memories. Data bus multiplexing results in a performance penalty, though, since a read operation on the memory should wait for the data to propagate to the output before a succeeding write operation can be performed.

Another, though more subtle, form of multiplexing usually found in DRAMs is address bus multiplexing. The address bus is *time multiplexed*, which means that the address should be split into two parts, called the *row address* and the *column address*, and set up one after the other on the address pins. To control this process, two input signals are used: the *row address strobe* ($\overline{RAS}$), to indicate that the row address is ready on the address lines, and the *column address strobe* ($\overline{CAS}$), to indicate that the column address is ready. These two signals are externally provided by the user as control signals. The reason for the names row and column addresses will be clear later on in this chapter. Since the three input signals $\text{R}/\overline{\text{W}}$, $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ provide the control for the memory, they are often considered together to make up a new input bus called the **command bus**.

Figure 2.3(a) shows a generic pin configuration of a DRAM package having a clock input (Clock), 6 address lines (A0 ... A5), 4 data lines (D0 ... D3), a $\text{R}/\overline{\text{W}}$ input, $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ control signals, and the power supply pins V_{dd} and GND. The memory has a multiplexed address bus (since $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signal are present), which means that the address has a maximum width of $6 \times 2 = 12$, resulting in as many as $2^{12} = 4096$ different addresses. And since the memory has a word length of 4 bits, the size of the memory is $2^{12} \times 4 = 16$ Kb. Figure 2.3(b) shows an example package of a 512 Mb DDR DRAM from Infineon, which has a total of 66 pins, 13 of which are address pins ($2^{13\times2} = 64$ M address) each accessing one byte ($2^{26} \times 8 = 512$ Mb).



(a) (b)

Figure 2.3. (a) Pin configuration, and (b) a packaging example (source: Infineon Technologies).

One way to quantify the performance of a memory is to calculate its data transfer rate, also called *data rate* or **bandwidth** (BW), which is defined as the maximum number of bytes the memory can transfer across its data bus per second during a full memory operation.

$$\text{BW} = \frac{\text{number of bytes transferred}}{\text{operation time}} \tag{2.1}$$

According to this relation, a higher memory performance could be attained either by decreasing the operation time (resulting in a faster memory with a higher clock frequency) or by increasing the width of the data bus (resulting in more bytes transferred per operation). Faster memories have a reduced data access time, which results in a higher memory bandwidth and improved performance. Wider bus memories allow more bytes to be transferred in a given amount of time which also increases the performance. The higher bandwidth requirement is driving the move toward higher speeds and wider data buses in new memory designs.

2.2.2 DRAM operations and timing diagrams

Timing diagrams are waveform representations of the signals on the input and output lines of the memory while performing a specific memory operation. These diagrams identify the timing interdependencies between different memory signals and define the timing conditions required for proper operation. In the following, the timing diagrams for read and write operations are shown and discussed in detail. One general timing parameter shown in all timing diagrams is the *clock cycle time* or the *clock period* (t_{CK}) which gives the specs for the clock signal. The specs for this timing parameter and others can be found in Table 2.1, as discussed later in this section.

Read operations

Read operations are commonly referred to as $r0$ and $r1$, where the 0 and 1 are the values expected on the output by the read operation. Figure 2.4 shows a typical timing diagram of a read operation performed on a 512 Mb, 400 MHz Infineon DDR DRAM chip, using a rather relaxed cycle time of $t_{CK} = 10$ ns [Infineon04]. The commands on top of the figure (No operation, Activate, etc.) are the internal DRAM commands during each clock cycle of memory operation, as discussed in Section 2.3. The figure shows that a read operation starts by setting up the first half of the address (the row address) on the address bus a minimum time period of t_{IS} (input setup time) before the rising edge of the clock signal, in order for the address to stabilize on the inputs. The address should be held on the inputs for t_{IH} (input hold time) after the clock so that the address can be read properly. The user informs the memory that the address is present on the inputs by pulling the $\overline{\text{RAS}}$ down, in combination with pulling the R/$\overline{\text{W}}$ up. After providing (or strobing) the row address, a minimum period of t_{RCD} (row-column delay time) should pass before the column address can be provided to the memory, which is initiated by pulling the $\overline{\text{CAS}}$ down. At the same rising edge of the clock, the read command needs to be provided by pulling the R/$\overline{\text{W}}$ up to declare the current operation as a read operation. The data stored in the addressed cell appears on the data bus a period of CL ($\overline{\text{CAS}}$ latency) after setting up the address and issuing the read command to the memory. In the case shown in the figure for example, CL has the value of $3 \times t_{CK}$. To declare the end of the read operation, $\overline{\text{RAS}}$ must be pulled down, in combination with pulling R/$\overline{\text{W}}$ down (as opposed to pulling it up at the *beginning* of a read). From start to finish, the length of the read operation is governed by the parameter t_{RAS} (row address strobe time) which defines the maximum and minimum time between two $\overline{\text{RAS}}$ pulses of a single memory operation. After the end of the read operation, a minimum time period of t_{RP} (row precharge time) should pass before a new operation may start. In total, a read operation should last at least a period of t_{RC} (row cycle time) before a subsequent operation is issued.

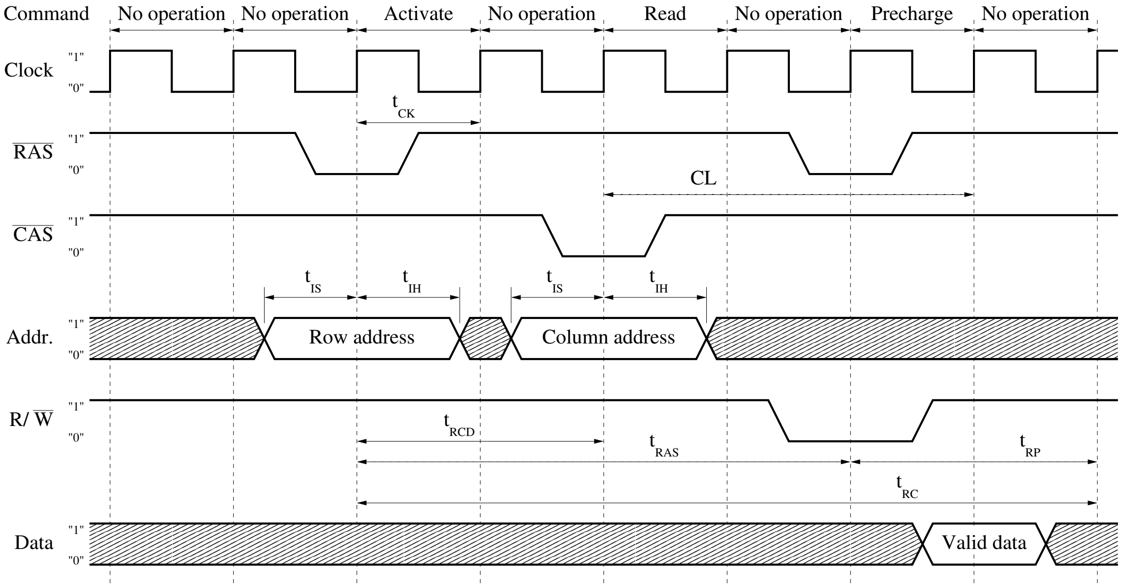


Figure 2.4. Timing diagram of a read on a 512 Mb, 400 MHz DRAM, using t_{CK} = 10 ns.

Write operations

Write operations are commonly referred to as $w0$ and $w1$, where the 0 and 1 are the values to be written by the write operation into the cell. Figure 2.5 shows a typical timing diagram of a write operation. The figure shows that setting up the row and column address for the write operation is identical to that for the read operation, and has the same timing parameters. As the column address is provided to the memory, the R/$\overline{\text{W}}$ signal should be pulled down to declare the current operation as a write operation. One clock cycle later, the write data should be provided on the data bus and left to stabilize for t_{DS} (data setup time) before the rising edge of the clock, and for t_{DH} (data hold time) afterwards. A minimum period of t_{WR} (write recovery time) should pass before ending the write operation (by pulling $\overline{\text{RAS}}$ down) to give the memory enough time for writing the required voltage into the cell. t_{WR}, commonly referred to as the **write back window**, is an important time parameter in testing which is used to stress the ability of a memory to write a specific voltage into the cell. This parameter is extensively investigated as a stress condition in Chapter 9 to test for the elevated strap resistance problem.

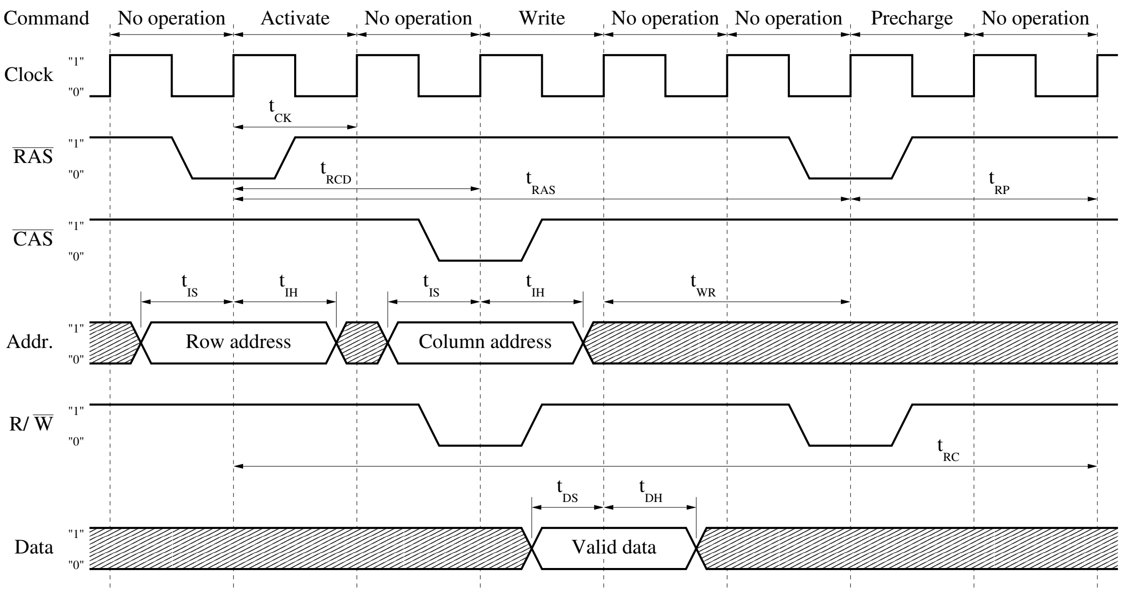


Figure 2.5. Timing diagram of a write on a 512 Mb, 400 MHz DRAM, using $t_{CK} = 10$ ns.

Timing parameters

Each DRAM has its own set of timing parameters for read and write operations, depending on the specific memory design and behavior. Table 2.1 gives an overview of the read and write timing parameters (given in ns) of a 512 Mb DDR SDRAM chip (running with CL = $3 \times t_{CK}$, @ 400 MHz) manufactured by Infineon[04]. The parameters listed here correspond to those used in the read and write timing diagrams above. Each parameter is given a short explanation and is provided with minimum and maximum bounds. In the list, one of the most important parameters for SDRAMs is t_{CK} (clock cycle time) since it gives the bus frequency at which the memory operates, which (in combination with CL) indicates the time needed till data can be retrieved from the memory. SDRAMs are usually characterized by the minimum bound of this parameter, such that a memory with $t_{CK,min} = 5$ ns is said to be a -5 SDRAM. This number is also referred to sometimes as the **speed sorting** (or *sorting*) of the memory, which indicates the speed the memory is able to achieve for a given technology. It is interesting to note that t_{CK} has an upper bound in the specs, which means that the memory frequency may not go below a given lower bound. This is only true for DDR SDRAMs (not for regular SDRAMs) since they employ an internal *delay-locked loop* (*DLL*) circuit to align the rising and falling edges of data signals with respect to the clock. Optimum operation of a DLL requires a clock signal that operates within a given max-min frequency range.

The timing parameters listed in Table 2.1 are not independent from each other, but should satisfy some timing relationships among each other. For example, the input and data, setup and hold times of the memory should be less than half the minimum time of the clock cycle, since it should be possible to setup a different

Table 2.1. Timing parameters of a 512 Mb DDR DRAM chip from Infineon (given in ns).

Parameter	Description	Min.	Max.
t_{CK}	Clock cycle time	5	12
t_{DH}	Data hold time	0.4	—
t_{DS}	Data setup time	0.4	—
t_{IH}	Input hold time	0.6	—
t_{IS}	Input setup time	0.6	—
t_{RAS}	Row address strobe time	40	$70,000$
t_{RC}	Row cycle time	55	—
t_{RCD}	Row-column delay time	15	—
t_{RP}	Row precharge time	15	—
t_{WR}	Write recovery time	15	—

input to the memory at each clock cycle.

$$t_{IS}, t_{IH}, t_{DS}, t_{DH} \leq \frac{1}{2} t_{CK,min} \tag{2.2}$$

The following examples show that the period of any timing parameter must be larger than the period of the timing parameters it covers.

$$\begin{aligned} t_{RAS} &\geq t_{RCD} + t_{WR} \quad \text{[see Figure 2.4]} && (2.3) \\ t_{RC} &\geq t_{RAS} + t_{RP} \quad \text{[see Figure 2.4]} && (2.4) \\ t_{RC} &\geq t_{RCD} + t_{WR} + t_{RP} \quad \text{[see Figure 2.5]} && (2.5) \end{aligned}$$

Other operations

Besides the read and write operations, DRAMs nowadays have a variety of operational modes that enable them to provide more functional flexibility or higher performance for a given technology.

One such operation is the *refresh operation*, where all cells are rewritten using the same values they contain. This prevents losing the data contained in memory cells due to naturally occurring leakage currents. The refresh operation is issued to the memory by providing a special sequence of values on the command bus that the DRAM interprets as a request to refresh.

The *fast page mode* is another mode of operation meant to increase the performance of DRAMs. This mode of operation starts by issuing a row address to the memory, which opens a full row of memory cells (also called a *memory page*). Any cell from this page can subsequently be read or written by providing only the column address of the specific cell to be accessed. Figure 2.6 shows the timing diagram of two write operations performed in the fast page mode, the first of which is performed on a cell with column address "Col. add. 1", followed by another write performed on a different cell with column address "Col. add. 2". This mode of operation can dramatically increase the bandwidth of the memory by reducing the access time, since only column address part need to be provided on the inputs. A deeper treatment of these and other modes of operation can be found in the literature [Prince99].

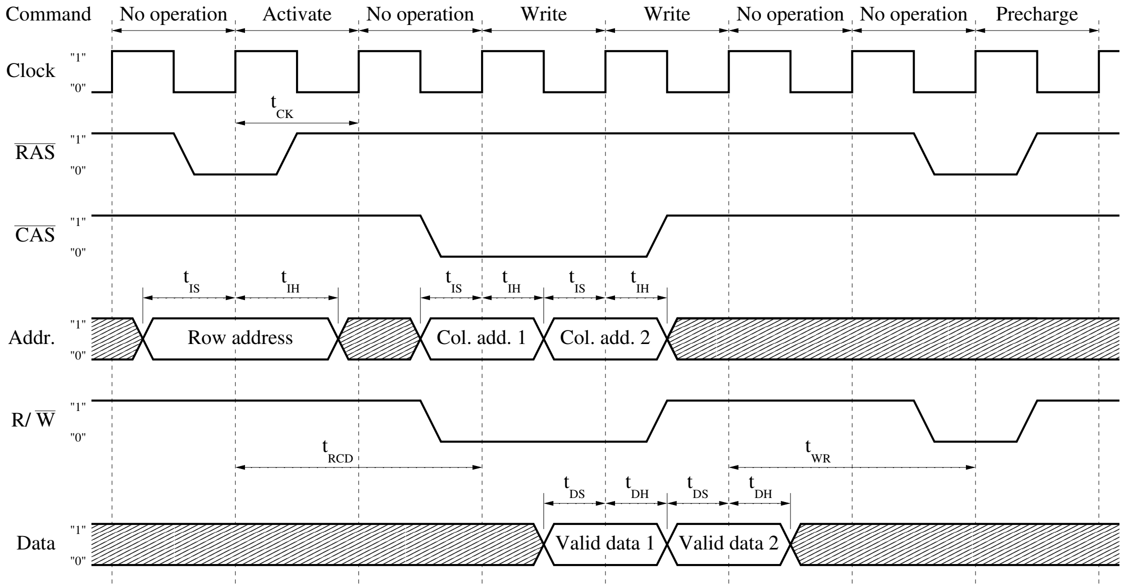


Figure 2.6. Timing diagram corresponding to a DRAM fast page mode of operation.

2.3 Functional DRAM model

A memory can be subdivided into a number of interacting functional units, each with its own function, that contribute together to achieve the desired external memory behavior. This section investigates these different memory functions and presents a functional DRAM model that describes them, together with the way they interact. At this level of abstraction, we are dealing with the internal structure of the memory, that represents the DRAM as a collection of interconnected functional blocks, each performing its own distinct function.

2.3.1 Functional block diagram

Figure 2.7 shows a simplified functional block diagram for DRAMs. The figure distinguishes several functional blocks needed for the DRAM to operate properly, such as the memory cell array, the control logic, and the address decoders. These blocks are discussed below in more detail.

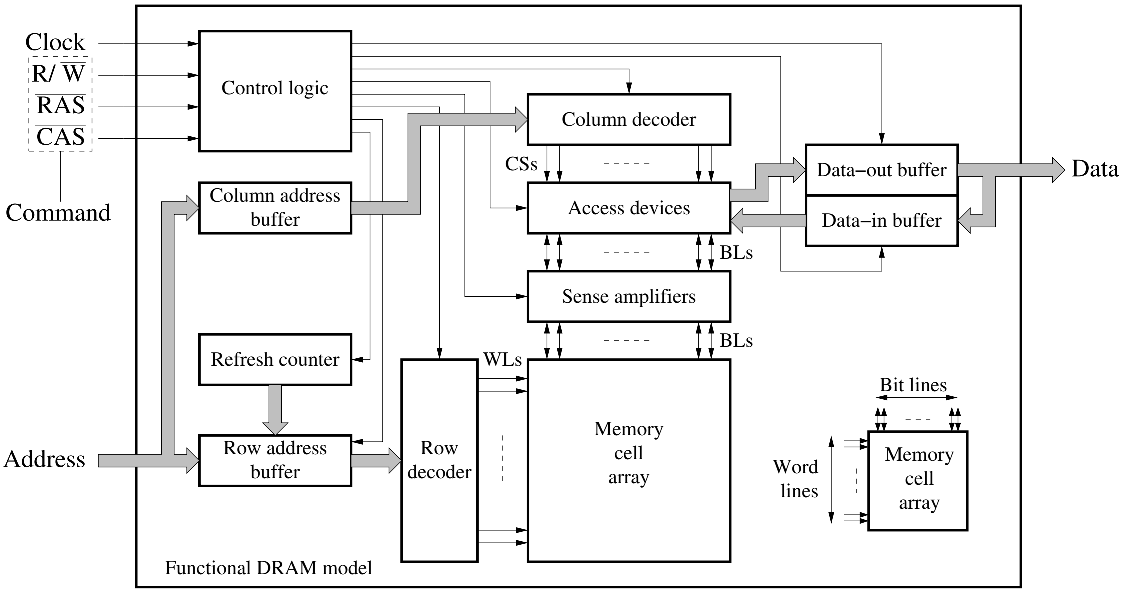


Figure 2.7. Simplified functional block diagram of a DRAM.

Memory cell array The memory cell array block is the most significant part of the DRAM since it occupies up to 60% of the chip area. This block contains the memory cells arranged next to each other in the form of an array, with rows and columns, such that the memory chip may have a rectangular shape. For example, a 1 Mb memory with 1 M cells can practically be organized as an array with 512 rows and 2048 columns, 1024 rows and 1024 columns, or 2048 rows and 512 columns of cells. The external DRAM behavior partly reflects this internal organization by requiring to split the address into the row address (used to select one row of cells) and the column address (used to select a column). Due to the relatively large size of the cell array, it has become the focus of attention for fault analysis and test generation activities.

Control logic The memory uses the control logic (also called the *timing generator*) to activate and deactivate the desired functional blocks at the right moments. The control logic takes the clock and the command bus (R/$\overline{\text{W}}$, $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$) as its inputs and uses them to generate internal control signals for almost all other functional blocks in the memory. As an example, the row and column address buffers are used to hold the row and column addresses, respectively. The control logic instructs the buffers to sample the addresses when they appear on the inputs, based on the values present on the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines.

Address decoders In order to address a cell in the memory cell array, row and column addresses need to be decoded. This takes place in the row and column decoders, respectively. The inputs of the address decoders are cell addresses, while the outputs are called *word lines* (*WLs*) in the case of the row decoder, and *column select* (*CS*) lines in the case of the column decoder. Each row and column in the memory has a specific line that selects it, and the combination of selecting a row and a column results in selecting a single word in the array.

Other functional blocks The *data-in buffers* and *address buffers* are used to latch input data and addresses at the input, while the *data-out buffer* stores read output data and keeps it for the user on the data bus. The *sense amplifier* is the part of the memory used to identify the data stored within the memory cells in the cell array. This block is needed because data bits within the cells are stored with low energies and in weak leaky capacitors, such that data in the memory cells need to be amplified before they can drive other circuits in the memory. The access devices are used as an interface between the data buffers and the sense amplifiers. Depending on the column address, only a limited number of columns is connected to the data buffers, and depending on the performed operation, either the write or the read buffer is connected to the sense amplifiers. The last functional block shown in the figure is the *refresh counter*, which is responsible for counting through the addresses of the memory so that data stored in all memory cells can be refreshed. The control logic is the part responsible of regulating the functionality of all these functional blocks.

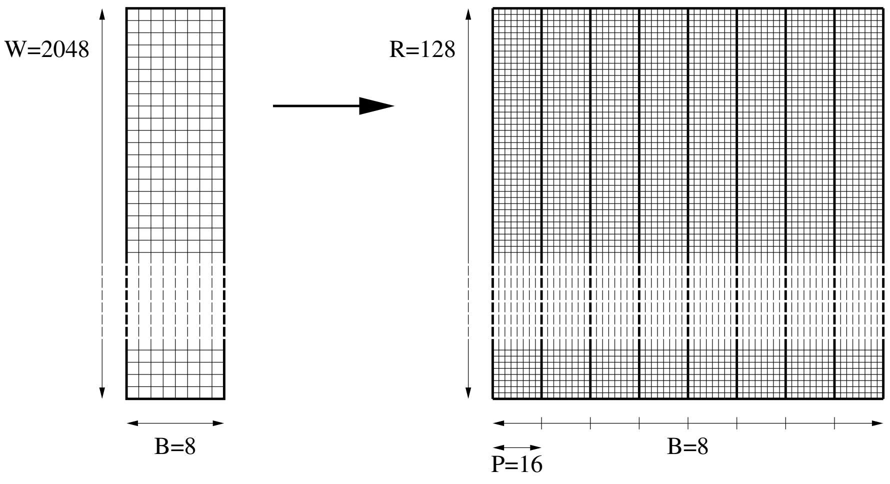
2.3.2 Cell array organization

The memory cell array is the place where memory cells are organized next to each other in the form of an array. In the following, cell array organization is described in terms of cell placement (which refers to the way the cells are placed next to each other) and cell connection (which refers to the way cells are connected to one another).

Cell placement

Each memory cell is able to store one bit of data. If we assume that the number of bits in each data word in the memory is B, and that the number of data words is W, then the total number of bits (and consequently memory cells) in the memory is equal to $W \times B$. During a read or a write operation, all bits within a single word are accessed simultaneously by the WL. Consequently, a natural organization of the cell array would be to stack all words on top of one another and connecting each word to its own WL, as shown in Figure 2.8(a).

Such an organization would result in an unrealistically elongated cell array that is W bits long and B bits wide. A more reasonable way to organize the cell array would be to give it a roughly square shape, by making the length and width of the cell array roughly equal, a technique referred to as *array folding*. This is done by dividing the single stack of words into P equal parts and placing them side by side, such that all adjacent bits in all P words are accessed by a single WL



(a) Stacked data words

(b) Multiple columns

Figure 2.8. Cell array organization using (a) stacked data words, and (b) multiple columns.

simultaneously [see Figure 2.8(b)]. Subsequently, only one word of the P accessed words is selected using a multiplexer. This reduces the length of the cell array from W bits down to R bits (the number of **cell array rows** in each part), and increases the width of the cell array from B bits up to C, where C is the number of **cell array columns** needed to store all bits of a word. These cell array parameters discussed above are related to each other by the following relationships.

$$C = B \times P \tag{2.6}$$

$$R = \frac{W}{P} \tag{2.7}$$

B and W are already given, but we may choose the value of P arbitrarily in such a way to get the desired *form factor* for the cell array. The form factor can be defined as the width of the array divided by its length, or more precisely $\frac{R}{C}$. For an exactly square cell array with a form factor of 1, the following relationships should be satisfied.

$$P = \sqrt{W/B} \tag{2.8}$$

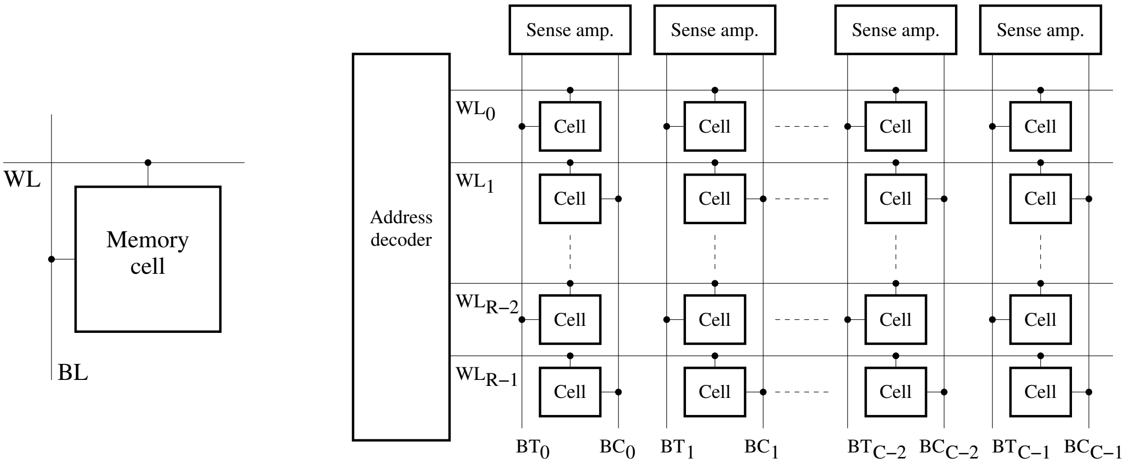
$$R = C \tag{2.9}$$

$$= \sqrt{W \times B} \tag{2.10}$$

Figure 2.8 shows an example of array partitioning, where a 2 K × 8 array is partitioned into an array with 16 parts ($P = \sqrt{2^{11}/2^3} = 2^4$) and with 128 rows and columns ($R = C = \sqrt{2^{11} \times 2^3} = 2^7$). This simple example represents a special case, where it is possible to generate an exactly square partitioned array, but when this is not possible, a different form factor other than but close to 1 is chosen.

Cell connection

Each cell in the cell array is connected to a word line (WL) and to a bit line (BL), as shown in Figure 2.9(a). The WL carries the signal that controls access to the cell, while the BL carries the write data into the cell during a write operation, and the read data out of the cell during a read operation. Figure 2.9(b) shows how multiple cells are connected to each other in the cell array, and to different parts of the memory around the array.



(a) Single cell

(b) Multiple cells

Figure 2.9. The way (a) a single cell, and (b) multiple cells are connected in the cell array.

Every WL in the cell array defines a single cell array row, which means that when a single cell in a given row is accessed, *all* other cells in that row are accessed too. The WLs originate from the row address decoder, which selects a given WL, depending on the specific row address provided on the input. In contrast to array rows, cell array columns are defined by a **BL pair**, one of which is called the *true bit line* (*BT*) while the other is called the *complement bit line* (*BC*). The true BL is connected to half of the cells of a cell array column, while the complement BL is connected to the other half of the cells. Both BT and BC are connected to the sense amplifier, which is used to sense the voltage level in the accessed cell. This BL organization is called **bit line folding**, which distinguishes these *folded bit lines* from *open bit lines*, where all cells in a column are connected to the sense amplifier through a single BL. The names true and complement refer to the complementary logical interpretation of voltage levels present in the cells on these BLs. In other words, a voltage high (H) in a cell on BT is interpreted by the sense amplifier as logic 1, while the same voltage H in a cell on BC is interpreted as logic 0. In a similar way, a voltage low (L) in a cell on BT is interpreted by the sense amplifier as logic 0, while the same voltage L in a cell on BC is interpreted as logic 1. This kind of complementary interpretation of the voltages stored in different cells is referred to as *data scrambling*, while the actual voltage levels stored in the cell are referred to as *physical data*, as opposed to the term *logical data* used for the way stored voltages are interpreted.

2.3.3 Internal DRAM behavior

As discussed in Section 2.2, DRAMs today have a number of different operations (besides simple reads and writes) that aim primarily at increasing the performance by reducing the time needed to access stored information (which is the case with the fast page mode), or aim at efficiently automating necessary DRAM functionality (which is the case with the refresh operation). These external operations, and many others, are implemented internally using only five primitive **DRAM commands**. These commands are represented in Figure 2.10 and are described next:

1. **Act**: This is the *activate command*. When this command is issued, a word line (WL) in the cell array is selected, which accesses a row of memory cells. Furthermore, an *internal* read operation is performed by moving the data from the row of memory cells to the sense amplifiers.

2. **Rd**: This is the *read command*. When this command is issued, the data in one of the sense amplifiers is moved to the data buffers and on to the data bus. This resembles an *external* read operation.

3. **Wr**: This is the *write command*. When this command is issued, the data in the data buffers is moved to both the sense amplifiers and the cell array as well. This resembles an external as well as an internal write operation.

4. **Pre**: This is the *precharge command*. When this command is issued, any selected WL is deselected and internal voltages are precharged to their predefined voltages.

5. **Nop**: This is the *no operation command*, which does not change the state of the memory, but simply extends the time span of any previously issued command. Interestingly, this means that the impact of the Nop command on the memory depends on the type of the previously issued command, and not on the Nop command itself.

It is important to note that these commands are not independent from one another, but need to be issued in a specific predefined manner for the memory to function properly. This is unlike memory operations such as writes and reads, which can be performed at any instant irrespective of the operations performed before or after the current operation. In principle, any command should be issued according to the following sequence

$$\text{Act } \text{C}d_1 \text{ ... } \text{C}d_i \text{ ... } \text{C}d_n \text{ Pre Nop ... Nop}, \ \text{C}d_i \in \{\text{Wr0}, \text{Wr1}, \text{Rd0}, \text{Rd1}, \text{Nop}\} \quad (2.11)$$

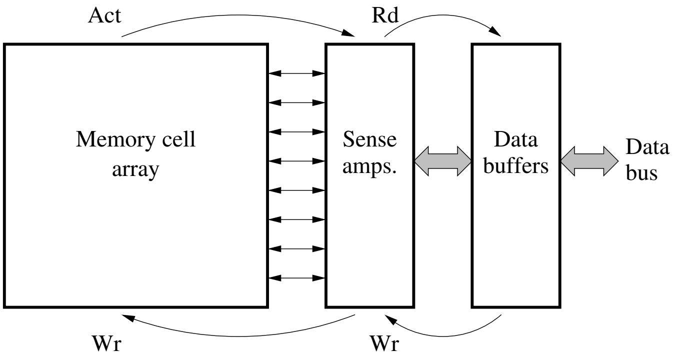


Figure 2.10. Internal behavior of a DRAM.

Usually, a number of Nop commands should be present in between other Cd_i commands issued between Act and Pre, as well as after the Pre command. The number of Nops that should be used depends on the timing parameters given in the specifications of the memory (such as those given in Table 2.1), such that the timing requirements are fulfilled. According to the timing diagram in Figure 2.4, for example, Act is followed by one Nop and Rd is followed by one Nop as well, before subsequent commands are issued.

Each of the DRAM commands is issued internally by decoding the signals provided by the user on the command bus. Table 2.2 gives the encoding used on the command bus in the form of a functional truth table with inputs $R/\overline{W}$, $\overline{RAS}$ and $\overline{CAS}$.

Table 2.2. Encoding of primitive DRAM commands in the command bus.

Command	Symbol	$\overline{RAS}$	$\overline{CAS}$	$R/\overline{W}$
Activate	Act	0	1	1
Read	Rd	1	0	1
Write	Wr	1	0	0
Precharge	Pre	0	1	0
No operation	Nop	1	1	1

Using the above five primitive commands, any DRAM operation can be described, as shown in the following examples.

Write operation The write operation is traditionally denoted as wd_c, where d is the data to be written into cell c. Using the five primitive commands, the write operation is performed as Act_{c_r} Nop ... Nop $\text{Wr}d_{c_c}$ Nop ... Nop Pre, where c_r is the row address of c and c_c is the column address of c [see Figure 2.5].

Read operation The read operation is traditionally denoted as rd_c, where d is the expected data to be read from cell c. Using the five primitive operations, the read operation is performed as Act_{c_r} Nop ... Nop $\text{Rd}d_{c_c}$ Nop ... Nop Pre [see Figure 2.4].

Refresh operation The refresh operation is used to restore data into the memory cells to prevent losing stored data by leakage. This operation cannot be represented using the traditional r and w operations. Using the five primitives, the refresh operation is performed as Act_{c_r} Nop ... Nop Pre for all rows c_r in the memory.

Read modify write operation This operation performs a Rd followed by a Wr command on the same cell, without the need to provide a new address or precharge the memory in between. This operation cannot be represented using the traditional operations. Using the five primitives, this operation is performed as Act_{c_r} Nop ... Nop $\text{Rd}x_{c_c}$ $\text{Wr}y_{c_c}$ Nop ... Nop Pre.

Fast page mode In this mode, Rd and Wr commands are performed on any cell on a given activated WL (page) before precharging. This mode of operation can greatly increase the performance of the memory. Using the five primitive commands, the fast page mode is performed as Act_{c_r} $\text{C}d1_{cc1}$ ... $\text{C}dn_{ccn}$ Pre, where C is either Rd, Wr or Nop [see Figure 2.6].

Summary

This chapter discussed the most important aspects of DRAM behavior and architecture. The discussion primarily focused on the synchronous DRAM, as it is the most widely sold type of DRAM in the memory market today. The main issues presented in this chapter are the following.

- Definition of a top-down modeling approach typically used to design and analyze complex electronic systems. This approach consists of five layers, starting with the behavioral model on top, followed by the functional model, then the logical model, the electrical model, and finally the layout model at the bottom.

- Discussion of the external behavior of the DRAM and the way it is defined using timing diagrams in memory datasheets. For a memory, the external behavior is characterized by the different types of possible memory operations. This chapter detailed the behavior of three DRAM operations: write, read and fast page mode of operation.

- Introduction of the functional model of the DRAM, where the internal blocks of the memory are presented and discussed. This model distinguishes a number of different functional blocks, starting with the memory cell array, the control logic, the address decoder, the sense amplifiers, the data buffers, and the refresh counter.

- Description of the cell array organization. Two different aspects are discussed here, cell placement and cell connection. Cell placement is related to the way memory cells are collected together and placed next to each other within the cell array. Cell connection is related to the way cells are electrically connected to each other by bit lines and word lines within the cell array.

- Definition of internal DRAM commands that are more general and more flexible than simple external memory operations. There are five different DRAM commands: activate (Act), write (Wr), read (Rd), precharge (Pre) and no operation (Nop). DRAM commands can be combined together (under certain conditions) to construct all possible memory operations.

Contents of this chapter

3.1 Basics of MOS transistors
3.2 Electrical DRAM model
3.3 DRAM layout model

3

DRAM design and implementation

At the electrical level, DRAMs today are constructed using semiconductor components to achieve the high device integration possible using cutting-edge semiconductor fabrication processes. The most important building blocks used in semiconductor memory design are transistors, resistors and capacitors. These components are heavily integrated on silicon and used to construct a large variety of electrical circuits, each of which with its own specific functional contribution to the memory. In this chapter, we take a closer look at some of these circuits and discuss important aspects of their behavior. The chapter also presents important information about Spice simulation, needed to understand the simulation-based analysis in the thesis.

Section 3.1 discusses the details of transistor operation, and presents the mathematical relationships needed to analytically evaluate its behavior. Section 3.2 analyzes the electrical design and behavior of the most important DRAM circuits. Then, Section 3.3 presents the layout organization of the memory, and the different silicon implementation possibilities of DRAM cells.

3.1 Basics of MOS transistors

Metal oxide semiconductor (*MOS*) transistors are most predominantly used in electrical circuits to implement the functionality of a simple *electronic switch* that is turned on or off according to the voltage on a special control signal. Despite this seemingly simple application, MOS transistors have a rather complex behavior that requires special attention to characterize during fabrication, to ensure that its behavior is in line with the expected one.

3.1.1 Physical transistor structure

There are two types of MOS transistors, N-channel which use negatively charged carriers to transport current through its channel, and P-channel which use positively charged carriers for transport. Figure 3.1 shows a schematic cross section of an *N-channel MOS* (*NMOS*) transistor, which consists of a positively doped (p) *substrate* and two identical, strongly doped negative (n) regions. This is the opposite to a *P-channel MOS* (*PMOS*) transistor (not shown), which has a negatively doped substrate and two identical strongly doped positive regions. The two negatively doped regions of an NMOS are connected to the *drain* (D) and the *source* (S) terminals of the transistor, while the positively doped substrate is connected to the *body* or *bulk* (B) terminal of the transistor. The voltage on the B terminal controls the voltage of the electrical channel between the drain and source. Since the drain and the source of the transistor are identical, they are also interchangeable, and can only be distinguished by comparing the voltage levels on each terminal. In an NMOS transistor, the terminal with the higher voltage is called the drain, while the terminal with the lower voltage is called the source.

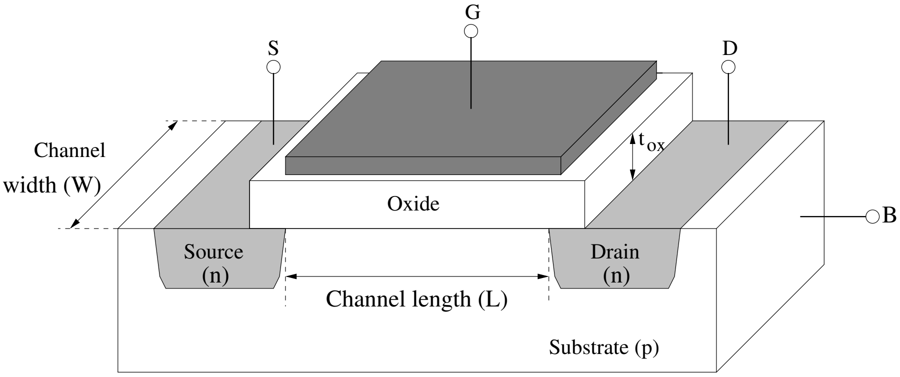


Figure 3.1. Schematic cross section of an NMOS transistor.

Conduction between the drain and source terminals of the transistor is controlled by the voltage on the *gate* (G), which is usually constructed using polysilicon (poly-Si) and is isolated from the channel by a thin layer of silicon dioxide (SiO_2). A number of different aspects of the transistor construction can be used to control its behavior, the most important of which are the length (L) and width (W) of the channel, used to control channel conductivity, and the oxide thickness (t_{OX}), used to control the threshold voltage (V_T) of the transistor.

Figure 3.2 shows the symbols usually used to represent NMOS and PMOS transistors in electrical schematics, where the different terminals of the transistor and the direction of the drain current (I_D) are indicated. It is often the case that the body terminal (B) is dropped from the schematic symbol.

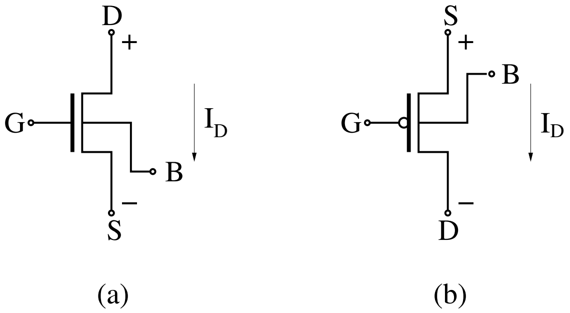


Figure 3.2. Electrical schematics symbols of (a) NMOS and (b) PMOS transistors.

3.1.2 MOS transistor behavior

In order to characterize the operation of the transistor, we need to identify a mathematical current-voltage relationship of the form

$$I_D = f(V_D, V_S, V_G, V_B) \tag{3.1}$$

that describes the impact of the voltage on each terminal of the transistor on the drain current flowing through it. Depending on the different terminal voltages and the threshold voltage (V_T), a transistor operates in three different *operation regions*, each with its own current-voltage relationship. In these relationships, differential voltages relative to the source voltage (V_S) are more commonly used than absolute terminal voltages. More specifically, the relationships use $V_{DS} = V_D - V_S$, $V_{GS} = V_G - V_S$, $V_{BS} = V_B - V_S$, in addition to V_S. The threshold voltage is calculated as follows [Weste94]:

$$V_T = V_{T0} + \gamma \left(\sqrt{2\phi_F + |V_{BS}|} - \sqrt{2\phi_F} \right) \tag{3.2}$$

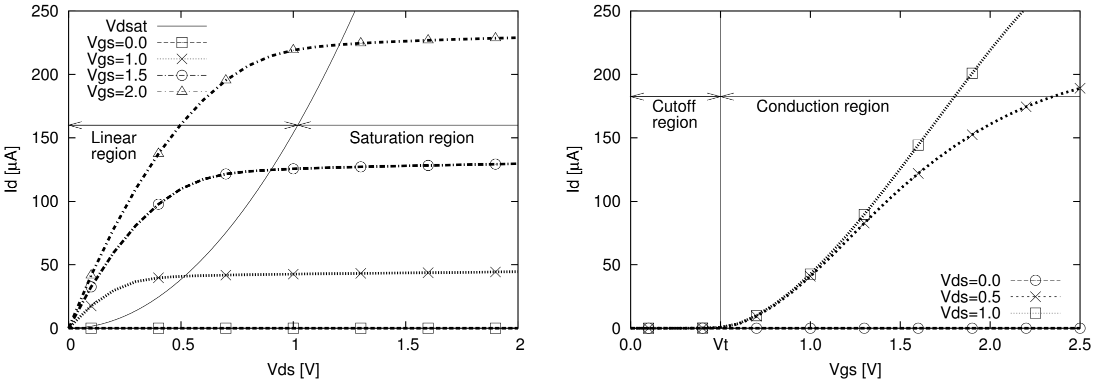
where V_{T0} is the threshold voltage at $V_{BS} = 0$ V. V_{T0} is positive for NMOS transistors and negative for PMOS transistors. Equation 3.2 states that the threshold voltage is made up of a basic fundamental component (V_{T0}) and an added value that is controlled by V_{BS}. The contribution of V_{BS} to V_T depends on the *body-effect coefficient (γ)* and on the *Fermi potential (ϕ_F)*.

The V_T equation and other equations in this section represent the simple *Level 1* Spice MOS transistor model (based on the ideal *Shockley equations*) [Weste94]. Simulations in this thesis, however, are predominantly performed using the Spice *Level 8* transistor model (also referred to as the *Berkeley BSIM3v3 model*), the industry standard and the result of years of analysis and updates [Cheng96]. Table 3.1 lists values of the BSIM3v3 model parameters equivalent to those found in the Level 1 equations discussed in this section. These values are extracted by wafer-level measurements performed at MOSIS to describe a 0.18 μm manufacturing process from TSMC (Taiwan Semiconductor Manufacturing Company) [MOSIS].

Table 3.1. MOS transistor parameters for 0.18 μm technology of TSMC [MOSIS].

Symbol	BSIM equiv.	NMOS value	PMOS value	Unit	Description
Level	`LEVEL`	8	8	—	Spice transistor model
t_{OX}	`TOX`	$4.1 \cdot 10^{-9}$	$4.1 \cdot 10^{-9}$	m	Oxide thickness
γ	`K1`	0.589	0.583	$V^{1/2}$	Body-effect coefficient
μ_0	`UO`	263	113	$cm^2/V{\cdot}s$	Surface mobility at T_{nom}
V_{T0}	`VTHO`	0.369	-0.394	V	Zero-bias threshold voltage
T_{nom}	`TNOM`	27	27	°C	Nominal temperature

Depending on V_T, V_{GS} and V_{DS}, a transistor can function in one of the following three regions of operation: cutoff region, linear region and saturation region. In the linear and saturation region, the transistor conducts current and, therefore, both regions can collectively be referred to as the *conduction region*. Figure 3.3(a) shows the Spice simulation results of the I_D-V_{DS} characteristics of a transistor, with an increasing V_{GS} parameter (V_{GS} = 0, 1, 1.5 and 2 V). The V_{dsat} curve in the figure indicates the **drain saturation voltage** that separates the linear region from the saturation region.



(a) I_D-V_{DS} characteristics (b) I_D-V_{GS} charateristics

Figure 3.3. Characteristics of a 0.18 μm NMOS transistor where I_D is plotted versus (a) V_{DS}, and versus (b) V_{GS}.

Figure 3.3(b) shows the Spice simulation results of the I_D-V_{GS} characteristics of a transistor, with an increasing V_{DS} parameter (V_{DS} = 0, 0.5 and 1 V). The vertical line at $V_{GS} = V_T$ separates the cutoff region from the conduction region of the transistor.

The cutoff region

The conditions for the NMOS and PMOS transistors to be in the in the cutoff region are as follows.

NMOS: $V_{GS} < V_T$
PMOS: $V_{GS} > V_T$

When the transistor operates in the cutoff region, there is no current flowing through it, no matter what the drain to source voltage might be [see Figure 3.3(b)]. Therefore, the current relation in this region is simple.

$$\boxed{I_D = 0} \tag{3.3}$$

The linear region

The conditions NMOS and PMOS transistors need to satisfy to be in the linear region are as follows.

NMOS: $V_{GS} \geq V_T$ and $V_{DS} < V_{GS} - V_T$
PMOS: $V_{GS} \leq V_T$ and $V_{DS} > V_{GS} - V_T$

Under these conditions, current starts to flow in the transistor, such that an increase in V_{DS} or in V_{GS} results in a corresponding increase in the drain current. The increase in I_D is quadratic with V_{DS} and linear with V_{GS}. This can be seen from the following current-voltage relationship in the linear region [Weste94].

$$\boxed{I_D = \tfrac{\beta}{2}\left[2(V_{GS} - V_T)V_{DS} - V_{DS}^2\right]} \tag{3.4}$$

where β is called the *transconductance parameter*, which has different values for NMOS and PMOS transistors. β can be calculated as follows.

$$\beta_N = \mu_N \frac{\epsilon_{OX}}{t_{OX}} \frac{W}{L} \qquad\qquad \beta_P = \mu_P \frac{\epsilon_{OX}}{t_{OX}} \frac{W}{L} \tag{3.5}$$

The constants μ_N and μ_P are called the *electron* and *hole surface mobility*, respectively. The value of the surface mobility increases with decreasing temperature, which means that I_D increases with decreasing temperature. ϵ_{OX} is the *gate oxide permittivity*, while t_{OX} is the gate oxide thickness.

The saturation region

The transistor operates in the saturation region if V_{DS} becomes high enough to reach the *pinch-off point*, where the drain current gets saturated. For NMOS and PMOS transistors, these conditions are as follows.

NMOS: $V_{GS} \geq V_T$ and $V_{DS} \geq V_{GS} - V_T$
PMOS: $V_{GS} \leq V_T$ and $V_{DS} \leq V_{GS} - V_T$

Under these conditions, the drain current barely increases with increasing drain voltage, but saturates and stays almost constant. This behavior is expressed in the following relationship [Weste94].

$$\boxed{I_D = \frac{\beta}{2}(V_{GS} - V_T)^2\,(1 + \lambda V_{DS})} \tag{3.6}$$

This equation has two factors, one of them is a function of V_{GS} and the other is a function of V_{DS}. The dependence on V_{DS} is controlled by λ, the *channel length modulation parameter*, which has a relatively small value.

3.1.3 Transistor as a switch

As mentioned in the beginning of this section, MOS transistors are mainly used as switches in electrical circuits. Figure 3.4 shows the application of NMOS and PMOS transistors as switches (or **pass transistors**) to a load capacitor C, that represents the capacitive circuit load this switch is connected to. The shown transistor-capacitance configurations are very important for DRAMs, since they represent the way individual DRAM cells are modeled electrically. In the following, the charge up and discharge behavior of the circuit configurations of Figure 3.4 are discussed.

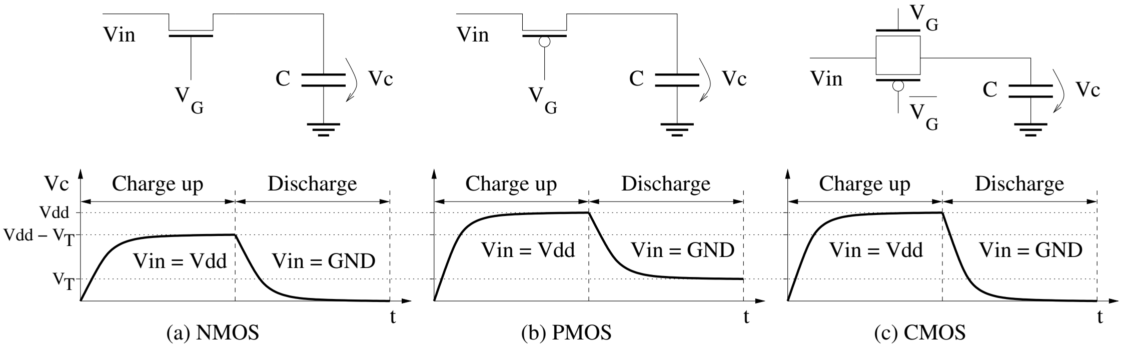


Figure 3.4. Switch implementation using (a) an NMOS transistor, (b) a PMOS transistor, and (c) a CMOS transmission gate.

NMOS pass transistor Assume that the capacitance in Figure 3.4(a) starts out fully discharged ($V_c = 0$ V) and with gate voltage V_G set to GND, thereby setting the transistor in the cutoff region ($I_D = 0$). If the voltage on the input and then the gate voltage are pulled high ($V_{in} = V_G = V_{dd}$), the pass transistor starts to conduct and starts to charge V_c up toward V_{dd}. As V_c approaches $V_{dd} - V_T$, the NMOS transistor starts to turn off since $V_{GS} = V_G - V_c$ approaches V_T, which results in setting the transistor into the cutoff region. The pass transistor cannot transfer a voltage higher than $V_{in} - V_T$ from the input to the load capacitance, which means that an NMOS transistor *degrades* a logic high as it passes from one of its terminals to the other. The discharge sequence of the transistor starts by setting $V_{in} = 0$ V, upon which the transistor starts to discharge V_c toward GND. As V_c approaches

GND, the NMOS transistor starts to turn off since V_{DS} approaches 0 V. The pass transistor, therefore, successfully discharges V_c to GND, which indicates that an NMOS does not degrade a logic low as it passes from one of its terminals to the other.

PMOS pass transistor Assume that the capacitance in Figure 3.4(b) starts out fully discharged ($V_c = 0$ V) and with gate voltage V_G set to V_{dd}, thereby setting the transistor in the cutoff region ($I_D = 0$ when $V_{GS} > V_T$ in Equation 3.3). If the V_{in} is pulled to V_{dd}, and then V_G is pulled to GND, the pass transistor starts to conduct and starts to charge V_c up toward V_{dd}. As V_c approaches V_{dd}, the PMOS transistor starts to turn off since V_{DS} approaches 0 V, which means that the transistor succeeds in transferring the full V_{dd} from the input to V_c. The discharge sequence of the transistor starts by setting $V_{in} = 0$ V, upon which the transistor starts to discharge V_c toward GND. As V_c approaches GND+$|V_T|$, the PMOS transistor starts to turn off since $V_{GS} = V_G - V_c$ approaches V_T, which sets the transistor into the cutoff region. The pass transistor cannot transfer a voltage lower than V_T from the input to the load capacitance, which means that an PMOS transistor *degrades* a logic low as it passes from one of its terminals to the other.

CMOS transmission gate A transmission gate is constructed using complementary MOS (CMOS) technology, where both NMOS and PMOS devices are implemented together in one circuit [see Figure 3.4(c)]. The sources and drains of both transistors of the transmission gate are connected together, and the gates of the NMOS and PMOS transistors are connected to V_G and $\overline{V_G} = V_{dd} - V_G$, respectively. The transmission gate is able to transport both the *full* voltage high *and* voltage low from one terminal to the other, since if one type of transistor stops conducting drain current at a given voltage, the other transistor continues until the full voltage level is achieved.

3.2 Electrical DRAM model

The electrical model of a memory consists of the *electrical schematics* of its circuits, which generally consists of transistors, resistors and capacitors. Each functional block in the functional model of the memory has its own specific implementation at the electrical level, which may vary in its complexity from the rather simple (such as the memory cell) to the rather complex (such as the control circuits). In this section, only a part of the electrical model of the memory is shown, only that part that is later used in the simulation and analysis of the memory behavior.

3.2.1 Electrical memory circuits

Each bus in the memory, be it the data, the address or the control bus, has its own *signal path*, which is, to a large extent, electrically independent from other memory signal paths. A **signal path** of a given bus consists of the memory circuits that receive the signals of the bus as inputs and then develop and shape the response of the memory according to those inputs. There are three signal paths in each memory, the *data path*, the *address path* and the *control path*, each of which is associated with a corresponding signal bus on the external interface of the memory [see Figure 2.2].

In this thesis, we are mainly concerned with simulating and analyzing the faulty behavior of the data path of the memory, rather than the address or the control paths. On the one hand, defects on the data path pose a more complex problem for test generation than those on the address path, for which a more theoretical evaluation of the behavior is sufficient [vdGoor04a]. On the other hand, analyzing the data path gives a general insight into the faulty behavior of the DRAM (due to the standard structure of the data path), as opposed to the more device-specific results gained from analyzing the control path (since it has a device specific implementation in each memory).

Figure 3.5 shows a block diagram of the data path of a DRAM [compare with Figure 2.7]. The figure shows a block for memory cells that is connected to the sense amplifier block through the true and complement bit lines (BT and BC), which then terminate at the access devices block. The BT and BC are also connected to the precharge and equalize block that ensures a proper voltage is set on the BLs at the beginning of each memory operation, as discussed later in this section. If a write operation is performed on the memory, the access devices connect BT and BC to the *true and complement write data lines* (*WDT* and *WDC*), which are driven by the data-in buffer. On the other hand, if a read operation is performed on the memory, the access devices connect BT and BC to the *true and complement read data lines* (*RDT* and *RDC*), which drive the data-out buffer. In the following, the electrical circuits that make up each block in the figure are discussed in detail.

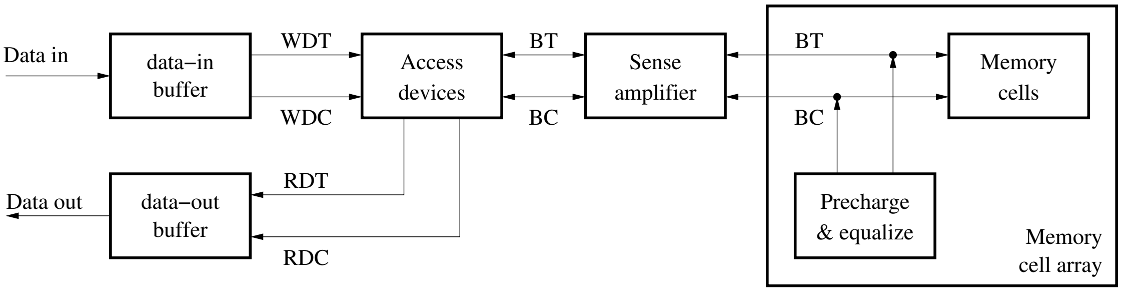
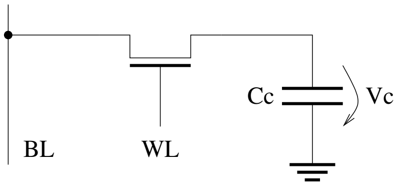


Figure 3.5. Block diagram of the data path of a DRAM.

3.2.2 Memory cell

The DRAM memory cell can be constructed in a number of different ways, the simplest and most widely used one of them is the so-called **one transistor DRAM**

cell (1T cell), which consists of a single pass transistor that controls access to a single, relatively large cell capacitor [see Figure 3.6(a)]. The pass transistor is controlled by a WL and connects the cell capacitor to a BL. This structure is very similar to the structure of a switch discussed in Section 3.1.3, where it has been shown that a single transistor (be it NMOS or PMOS) is not able to transport the full voltage level of both logic values from one input to the other. The transmission gate was proposed as a solution to this problem, making use of both NMOS and PMOS transistors. This solution is rather expensive for a memory cell, since it significantly increases the silicon area used by the cell, especially when noting that the memory cells in a DRAM occupy about 60% of the memory chip.



(a) 1T cell schematic

Name	Value	Description
C_c	40 fF	Cell capacitance
V_{boost}	3.5 V	WL boost voltage
V_{dd}	2.5 V	Voltage of logic high

(b) Value examples

Figure 3.6. Model of the 1T DRAM cell giving (a) the schematics, and (b) value examples.

A different solution is commonly used to solve this problem by applying a **boost voltage** (V_{boost}) to the WL, where the WL voltage (V_{WL}) is increased above V_{dd} for an NMOS or below GND for a PMOS to ensure a full voltage transfer into the cell. In order to achieve this, the boost voltage needs to satisfy the following conditions.

$$\text{NMOS}: \quad V_{boost} \geq V_{dd} + V_T \tag{3.7}$$

$$\text{PMOS}: \quad V_{boost} \leq V_T \tag{3.8}$$

Usually, a voltage well beyond these limits is used, in order to accommodate for manufacturing process variations and increase the robustness of the memory cells. The downside of a high boost voltage, however, is the need to use pass transistors with larger t_{OX} to prevent *gate oxide breakdown*. One positive outcome of such a increased gate oxide thickness is the reduction of the sub-threshold leakage currents into the cell, which contributes to increasing the data retention time of the memory. Figure 3.6(b) lists examples of the values used for the 1T DRAM cell schematic [Vollrath00, Vollrath97].

Cell charge up

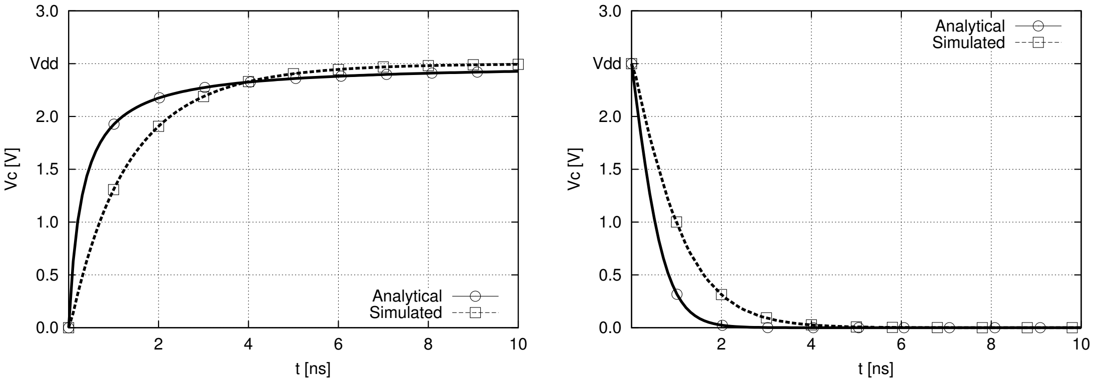
If we assume an NMOS pass transistor, a fully discharged cell capacitor ($V_c = 0$ V), a high voltage on the BL ($V_{BL} = V_{dd}$), and a boost voltage on the word line ($V_{WL} = V_{boost} = V_{dd}+V_T$), then the pass transistor would be in the linear region and starts to charge V_c up. From the I_D equation in the linear region (Equation 3.4), and using the electrical schematic of the cell in Figure 3.6(a), the cell capacitor charge up equation is:

$$\begin{aligned}
I_D &= C_c \frac{dV_c}{dt} \\
&= \frac{\beta}{2}\left[2(V_{GS}-V_T)V_{DS}-V_{DS}^2\right] \\
&= \frac{\beta}{2}\left[2(V_{dd}+V_T-V_c-V_T)(V_{dd}-V_c)-(V_{dd}-V_c)^2\right] \\
&= \frac{\beta}{2}\left(V_{dd}-V_c\right)^2 \\
C_c \frac{dV_c}{dt} &= \frac{\beta}{2}\left(V_{dd}-V_c\right)^2 \qquad (3.9)
\end{aligned}$$

Using the initial condition $V_c(0) = 0$ V, the solution of this differential equation with respect to $V_c(t)$ is as follows:

$$V_c(t) \;=\; \frac{A\,t}{A\,t+1}\,V_{dd}, \;\; \text{where } A = \frac{\beta\,V_{dd}}{2\,C_c} \qquad (3.10)$$

Figure 3.7(a) shows a simulation of the charge up sequence of the 1T memory cell, compared with a plot of the analytical charge up Equation 3.10 derived above (with $A = 1.3810^{9}$ s^{-}1). Both curves have the same general shape, increasing fast as the charge up process starts, and then asymptotically approaching (but never actually reaching) V_{dd}. The differences between the two curves can be attributed to the differences between the Level 1 NMOS model used in the analytical derivation performed above, and the complex behavior of the simulated BSIM3v3 transistor. The latter is the model that should be used, because it is more representative of the behavior on silicon.



(a) Charge up behavior

(b) Discharge behavior

Figure 3.7. Simulated vs analytical 1T DRAM cell curves of (a) the charge up, and (b) discharge behavior.

Cell discharge

If we assume an NMOS pass transistor, a fully charged cell capacitor ($V_c = V_{dd}$), a low voltage on the BL ($V_{BL} = 0$ V), and a boost voltage on the word line ($V_{WL} = V_{boost} = V_{dd} + V_T$), then the pass transistor would be in the linear region and starts to discharge V_c. From the I_D equation in the linear region (Equation 3.4), and using the electrical schematic of the cell in Figure 3.6(a), the cell capacitor discharge equation is:

$$\begin{aligned}
I_D &= -C_c \frac{dV_c}{dt} \\
&= \frac{\beta}{2}\left[2(V_{GS} - V_T)V_{DS} - V_{DS}^2\right] \\
&= \beta\left[(V_{dd} + V_T - V_T)V_c - \frac{1}{2}V_c^2\right] \\
&= \beta(V_{dd} - \frac{1}{2}V_c)\ V_c \\
-C_c \frac{dV_c}{dt} &= \beta(V_{dd} - \frac{1}{2}V_c)\ V_c \qquad (3.11)
\end{aligned}$$

Using the initial condition $V_c(0) = V_{dd}$, the solution of this differential equation with respect to $V_c(t)$ is as follows:

$$V_c(t) = \frac{2\ V_{dd}}{1 + e^{2At}}, \quad \text{where } A = \frac{\beta\ V_{dd}}{2\ C_c} \qquad (3.12)$$

A plot of Equation 3.12 with $V_c(0) = 0$ V is shown in Figure 3.7(b), along with a Spice simulation of the discharge process. The discharge operation starts with $V_c = V_{dd}$ and exponentially approaches GND, but never attains it. A comparison between the Equation 3.10 and 3.12 reveals that the charge up process is proportional to $1/t$, while the discharge process is proportional to the much faster e^{-t}. This is true with the NMOS transistor used as the pass transistor in the cell, but the opposite would be true if we use a PMOS for the pass transistor of the cell.

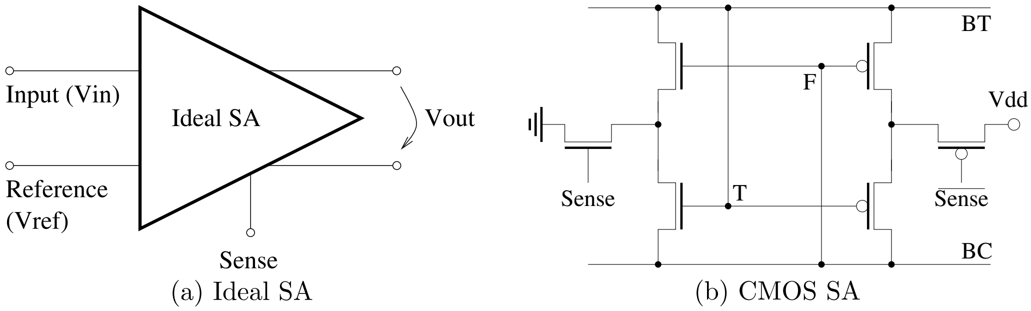
Write back window

One important design decision is to choose the minimum time given for the memory cell to reach its desired voltage when a write 0 (discharge process) or a write 1 (charge up process) takes place. For an NMOS pass transistor, the write 1 operation is slower than the write 0, which means that the charge up process is the one to decide the minimum write period. The opposite situation takes place for a PMOS pass transistor. This minimum time is referred to as the *write back window*, which is reflected in the external specifications of the memory by the timing parameter t_{WR} [see Table 2.1]. Since V_c never actually reaches V_{dd} or GND during the charge up and discharge processes, the designers must choose an acceptable fraction $0 < p < 1$ of the full voltage level as a target, and identify the minimum time period needed to attain it. If an NMOS is used as the pass transistor, then the slow charge up process is the one to decide $t_{WR,min}$, which is the time needed for V_c to reach a target level of $p \times V_{dd}$. Using Equation 3.10, the following condition must be satisfied:

$$t_{WR,min} > \frac{p}{A(1-p)}, \quad \text{where } A = \frac{\beta\, V_{dd}}{2\, C_c} \tag{3.13}$$

3.2.3 Sense amplifier

The sense amplifier (SA) is the part of the memory that "senses" the small charges corresponding to the stored logic levels in the memory cell, and then "amplifies" them before they are forwarded to the output. Although there are many types of SAs, each with its own principle of operation, most of them operate according to the same concept: a positive voltage differential is amplified by the sense amplifier to a full voltage high, while a negative voltage differential is amplified to a full voltage low.



(a) Ideal SA (b) CMOS SA

Figure 3.8. Circuit representation of (a) an ideal SA, and (b) a cross-coupled CMOS SA.

Sense amplifier structure

Based on the functional requirements of the SA, one can define a so-called **ideal sense amplifier** as shown in Figure 3.8(a), where only two different parameters (the input voltage, V_{in}, and the reference voltage, V_{ref}) on the input define the resulting voltage on the output (V_{out}) according to the following relation:

$$V_{out} = \begin{cases} V_{dd} & : \; V_{in} > V_{ref} \\ \text{GND} & : \; V_{in} < V_{ref} \end{cases} \tag{3.14}$$

The "Sense" signal in the figure gives the timing, indicating the instant the SA should be activated and sensing should take place.

The most common electrical circuit used to achieve the functionality of an ideal SA in memories is the *cross-coupled CMOS sense amplifier* shown in Figure 3.8(b).

It consists of two NMOS transistors connected on one side to a pull-down voltage path to GND, and two PMOS transistors connected on one side to a pull-up voltage path to V_{dd}. The nodes T (true) and F (false) connect one side of an NMOS to one side of a PMOS, and constitute both the input and the output of the sense amplifier. At the same time, the T and F nodes control the gates of the NMOS and PMOS on the other side of the SA. The sense amplifier is called a **bistable element**, which means that it has two stable states, one with T being high and F being low, while the other with T being low and F being high. The T node of the SA is connected to the true BL (BT) on one side, while the F node of the SA is connected to the complementary BL (BC). The power pull-down path to GND is controlled by the "Sense" signal, while the power pull-up path to V_{dd} is controlled by the complementary signal "$\overline{\text{Sense}}$".

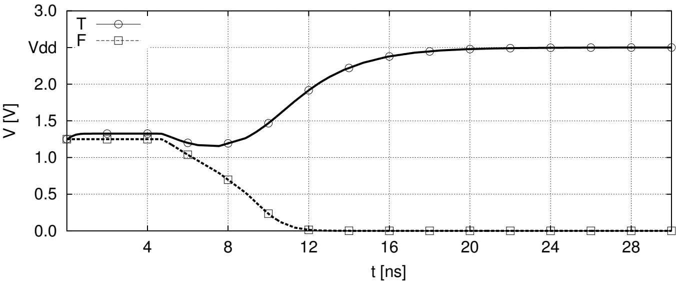
Sense amplifier operation

The SA operates in two main stages, the first of which is *voltage development stage*, and the second of which is the *sense and amplification stage* [Kang96]. The voltage development stage starts with the T and F nodes being *precharged* to the same *mid-point voltage* ($V_{dd}/2$), during which the "Sense" signal is kept low, thereby deactivating the SA by disconnecting it from the power supply. Then, a voltage differential ΔV develops across the T and F nodes of the SA, as a result of a memory cell being accessed. The amount of ΔV developed by the end of this stage is commonly referred to as the **signal margin of the SA**. The second stage (the sense and amplification stage) starts with pulling the "Sense" signal up, thereby activating the SA by connecting it to the power supply. The SA, then, senses the node that has the higher voltage and pulls it up all the way to V_{dd}, while pulling the voltage on the other node to GND.

The voltage development stage of SA operation lasts for a specific amount of time Δt_1 usually called the **signal development time**, which should be long enough for a sufficiently large signal margin ΔV to develop. The sense and amplification stage of SA operation lasts for Δt_2, which should allow enough time for the SA to charge the bit lines to a proper voltage level. Example values for Δt_1 and Δt_2 are 5 ns and 10 ns, respectively, for a 64 Mb DRAM manufactured in a 0.19 μm technology [Vollrath02]. The sum of these two timing parameters indicates the total length of time the SA should be given to sense the data from a memory cell and to restore it back into the cell. This takes place after the activate (Act) command is issued to the memory and before the write (Wr) or read (Rd) commands are issued. Therefore, the internal timing parameters Δt_1 and Δt_2 dictate the minimum amount of row-column delay time needed, as reflected by the external timing parameter t_{RCD} [see Table 2.1].

$$t_{RCD,min} > \Delta t_1 + \Delta t_2 \tag{3.15}$$

Figure 3.9(a) shows a simulation output of the operation of the sense amplifier for a positive differential voltage ΔV between the T and F nodes. The voltage development stage in the figure starts at 0 ns and ends at 4 ns, which is when a fully developed ΔV is present across the inputs of the SA. At 4 ns, the sense and amplify stage starts and the voltage differential is amplified, pulling T to a full V_{dd} and F to GND.



(a) Simulation of SA operation

Cause of V_{min}	Value
Transistor param.	27 mV
BL capacitances	10 mV
Coupling noise	9 mV
Total ΔV_{min}	46 mV
Typical ΔV	200–300 mV

(b) Minimum signal margin

Figure 3.9. (a) Simulation of sense amplifier operation, and (b) minimum signal margin for proper sensing.

Sense amplifier sensitivity

The ideal SA is perfectly balanced around its reference voltage. This is not the case for a real cross-coupled SA, which has a number of different sources of imbalance that cause a biased sensing of a 0 or a 1 on the output. The SA imbalance is a result of variations in the fabrication process of the memory, which makes different SAs behave differently. Causes of sense amplifier imbalance include [Sarpeshkar91]:

1. transconductance (β) of the sense amplifier transistors
2. gate-source parasitic capacitance of transistors
3. threshold voltage (V_T) of transistors
4. bit line capacitance

These deviations mean that, in order to ensure proper sensing for all SAs on a chip, a minimum differential voltage ΔV_{min} across the inputs of the SA is needed. The exact amount of ΔV_{min} depends on the design of the memory and the fabrication technology. Examples of such values are listed in Figure 3.9(b), based on measurements performed on a 16 Mb DRAM memory manufactured by Siemens [Geib92]. The table states that the imbalance in the transistor parameters requires a minimum voltage of 27 mV for proper sensing, the imbalance in the BL capacitances requires a minimum voltage of 10 mV, while the imbalance caused by BL-BL and BL-WL coupling noise requires a minimum voltage of 9 mV for proper sensing.

The typical amount of signal margin available for DRAM SAs during sensing is reported to be between 200 mV and 300 mV [Vollrath97]. This amount of signal margin depends on the cell capacitance (C_c), the bit line capacitance (C_b), the voltage to be sensed within the cell (V_c), and the voltage present on the bit line ($V_{dd}/2$).

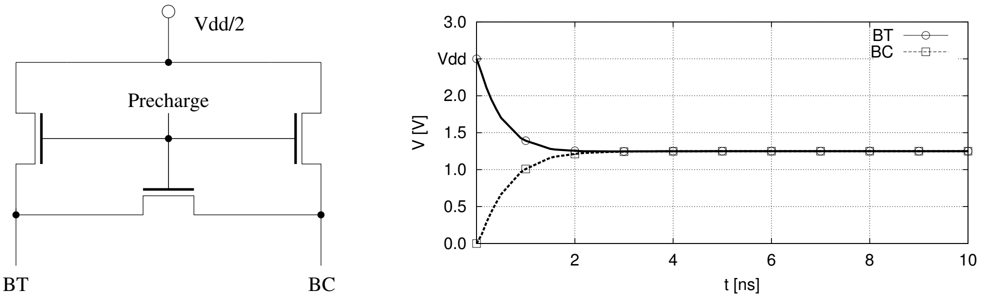
$$\Delta V = \frac{C_c}{C_c + C_b} \left(V_c - V_{dd}/2\right) \tag{3.16}$$

3.2.4 Other DRAM circuits

This section presents the electrical schematics, and discusses the operation of the rest of the DRAM circuits present in the data path in Figure 3.5. Three different circuits are discussed here: the precharge circuits, the access devices and the data buffer.

Precharge circuits

The operation of DRAM devices makes extensive use of the idea of precharging, which stands for setting the signal lines in the memory to a given predefined voltage before accessing them. Precharge circuits are very simple electrical constructs that connect the lines to be precharged to a known voltage level. Figure 3.10 shows one implementation of the precharge circuits that perform precharging to $V_{dd}/2$.



(a) Precharge circuits (b) Simulation results

Figure 3.10. (a) Schematic of the precharge circuits and (b) the voltage signals during precharging.

Figure 3.10(a) shows a three transistor implementation of the precharge circuits, two connecting each BL to the voltage supply, with the third transistor connecting the two BLs together at precharge. This third transistor is called the balance or equalization transistor, which ensures that both BLs are set to the exact same voltage level during precharging. All three transistors are controlled by the Precharge signal, which determines when the precharge action is to take place. The Precharge signal is only pulled high (activating the precharge circuits) during the precharge command (Pre) of the DRAM, while it remains low (deactivated) for all other DRAM commands.

Figure 3.10(b) shows the voltage signals on BT and BC when the precharge circuits are activated. The simulation starts with $V_{BT} = V_{dd}$, $V_{BC} = 0$ V, and with the Precharge signal pulled high. As a result, both BT and BC are pulled gradually toward each other, and eventually they reach the $V_{dd}/2$ level after about 2 ns from the beginning of precharging.

Access devices

Access devices are simple pass transistors used to ensure that the correct BL pair is connected to the outside world during read and write operations. According to Figure 3.5, the access devices are located between the sense amplifier on one side, and the data buffers on the other. Figure 3.11 shows a detailed view of the access devices, where two sets of devices are identified: the column gating and the read/write gating.

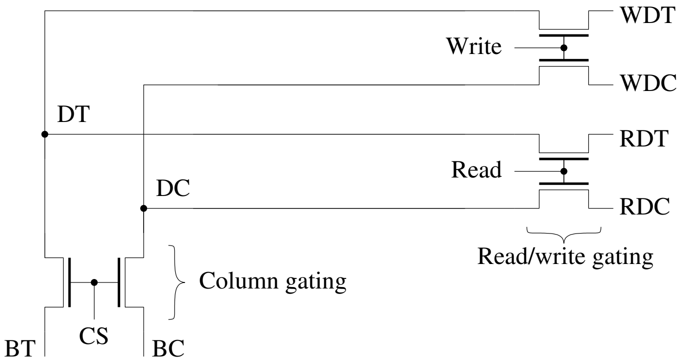


Figure 3.11. Schematic of the access devices.

The *column gating* decides which BL pair should be connected to the *data lines* (*DLs*) that carry read and write information to the BLs. The access transistors in the column gating are controlled by the column select lines (CSs) which, in turn, are controlled by the column decoder. When a given column address is selected, the corresponding CS is pulled high which connects BT to the *true data line* (DT) and BC to the *complement data line* (DC).

The *read/write gating* decides whether the DLs on the other side get connected to the read data lines (RDT and RDC) or to the write data lines (WDT and WDC), depending on the type of performed memory operation. During a read operation, the Read signal is pulled up and the Write signal is pulled down, which connects DT to RDT and DC to RDC. On a write operation, the Write signal is pulled up and the Read signal is pulled down, which connects the DT to WDT and DC to WDC.

Data buffer

The data buffer is the gate of the memory to the outside world, where input data is latched and made ready for the rest of the memory to use, and where output data is held steady for external circuits to read. One simplified implementation of the data buffer is shown in Figure 3.12, which represents the schematics of both the data-in buffer as well as the data-out buffer shown in Figure 3.5. In case of the data-in buffer, the lines connected to the buffer should be the data-in line, the WDT and the WDC, while in the case of the data-out buffer, the lines connected to the buffer become the data-out buffer, the RDT and the RDC.

The buffer consists of a simple clocked *latch circuit*. The design of the latch is very similar to that of the cross-coupled sense amplifier discussed in Section 3.2.3. The latch is a bistable element that, once set, keeps its data as long as it remains connected to the power supply. The activation and deactivation of the latch is controlled by the Latch signal, which is pulled low (activated) when the data become available to the latch (for example, at the rising edge of a clock cycle during a write operation).

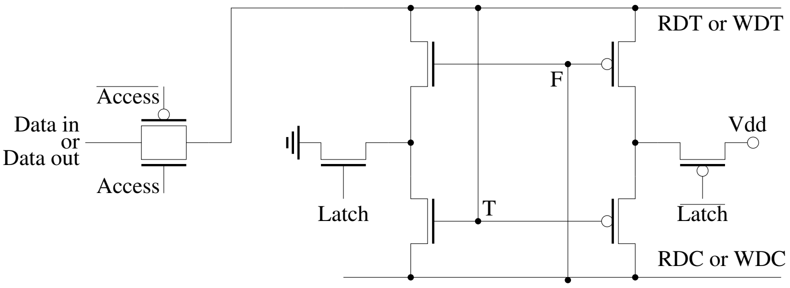


Figure 3.12. Schematic of the I/O data buffer.

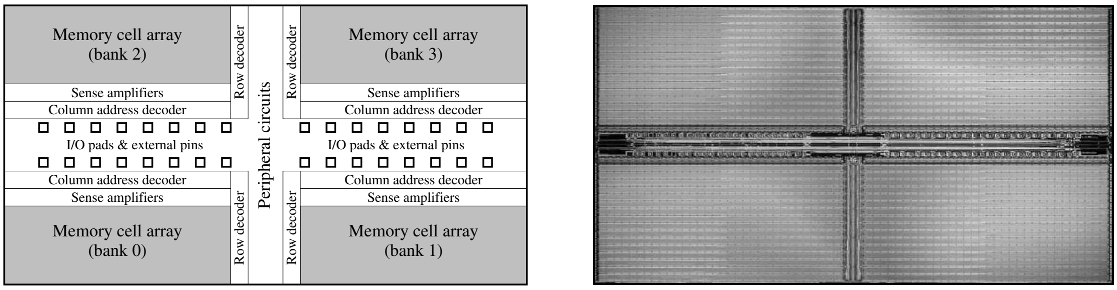
The external Data-in or Data-out pin is connected to the T (true) node of the latch through a transmission gate that ensures transferring a full voltage level to and from the buffer. The transmission gate is controlled by the Access signal which regulates the access between the memory and the external data pin. During a write operation, the Access signal is activated for a short time (t_1) *before* the Latch signal is activated, which ensures that the input signal has enough time to set the correct voltage at the buffer. The Access signal should also remain active for a short time (t_2) *after* the Latch is activated to ensure proper latching. These two time periods dictate the value of a couple of external timing parameters, the data setup time (t_{DS}) and the data hold time (t_{DH}) as follows [see Table 2.1].

$$t_{DS} > t_1 \tag{3.17}$$

$$t_{DH} > t_2 \tag{3.18}$$

3.3 DRAM layout model

The layout model (or geometrical model) is the most detailed description level of IC devices, where the physical structure of the circuits is given as it is manufactured on silicon. Layout design starts by the preparation of a **floor plan** of the memory chip, where the placement of the different circuits of the chip is decided. Figure 3.13(a) shows the floor plan of a typical multi-megabit DRAM chip, while Figure 3.13(b) shows the chip layout after manufacturing. One important characteristic of the floor plan of a multi-megabit DRAM is the cross-shaped layout area in the center of the chip, which is used for the I/O pads and the peripheral memory circuits. This structure is important for modern high-speed memories since it reduces the maximum length of the path external control signals have to travel to reach internal memory circuits [Itoh01]. The cross-shaped region makes it necessary to split the cell array into 4 different memory banks, each with its own column and row address decoders and sense amplifiers.



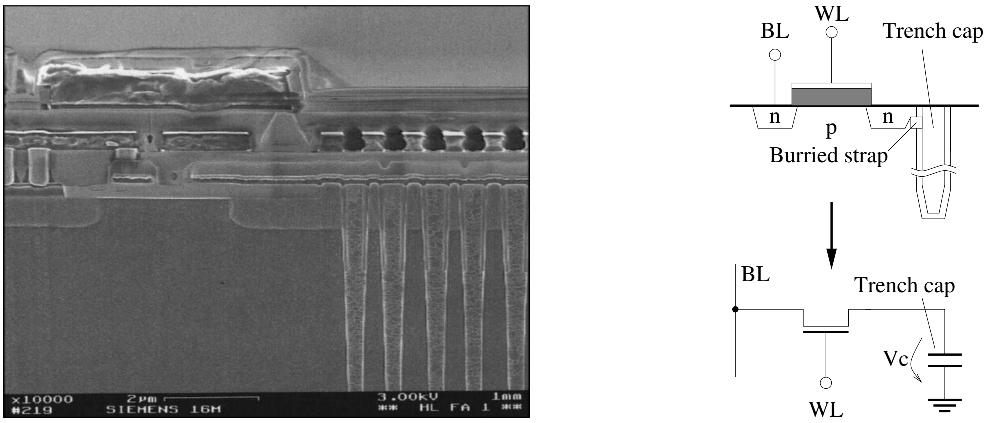
(a) Floor plan (b) Memory chip

Figure 3.13. Typical layout of multi-megabit DRAM. (a) Floor plan, and (b) Infineon memory chip (source: Infineon Technologies).

Detailed DRAM layout models of specific circuits are rarely reported in the literature due to the proprietary nature of this information, and the high complexity of the information itself. In this section, we restrict our discussion to the layout of the 1T DRAM cell, where the following two alternatives are the most commonly used cell implementations for multi-megabit DRAMs: the trench capacitor cell, and the stacked capacitor cell. These cell implementations use on-wafer, three dimensional capacitance structures, meant to increase the surface area of the capacitor plates vertically, instead of expanding horizontally and using expensive silicon area in the process. Needless to say, these three dimensional structures are difficult to manufacture, and require special optimization of the manufacturing process. They are, however, the best cost-effective solution so far to ensure the high cell capacitance needed for memory operation, as feature size miniaturization continues.

3.3.1 DRAM trench capacitor

The trench capacitor, championed by IBM and Infineon, started to feature in DRAM circuits at the 4 Mb generation and beyond, to keep a large enough surface area of the cell capacitor plates, by placing them on the sides of a narrow trench etched vertically into silicon. An added advantage to the trench capacitor is the greatly reduced defect levels per unit area, when compared to the defect levels common in previous capacitor technologies [Adler95]. Figure 3.14(a) shows a silicon cross-section where a number of trench capacitors are visible. Figure 3.14(b) shows a sketch of a 1T trench capacitor memory cell, where the gate of the pass transistor is controlled by the WL, while the source/drain nodes of the transistor are connected to the BL and to the trench capacitor. The trench capacitor itself consists of a top plate (made of a conductive polysilicon material that fills the trench), a thin insulator layer that lines the side walls of the trench (made of a silicon dioxide material), and a bottom plate (made of a negatively doped buried semiconductor diffusion layer).



(a) Trench capacitor (b) Electrical equivalent

Figure 3.14. (a) Cross-section of a trench capacitor, and (b) the electrical modeling of the layout (source: Infineon Technologies).

The trench capacitor is connected to the pass transistor using a **buried strap connection**, which acts as a bridge through the thin oxide insulator that separates the two plates of the trench capacitor. This buried strap is an essential part of the trench capacitor, that is generated using a special DRAM-oriented process demanding special care to be successful. One way to quantify the quality of the trench capacitor manufacturing process is using the **aspect ratio** of the trench which is calculated as follows.

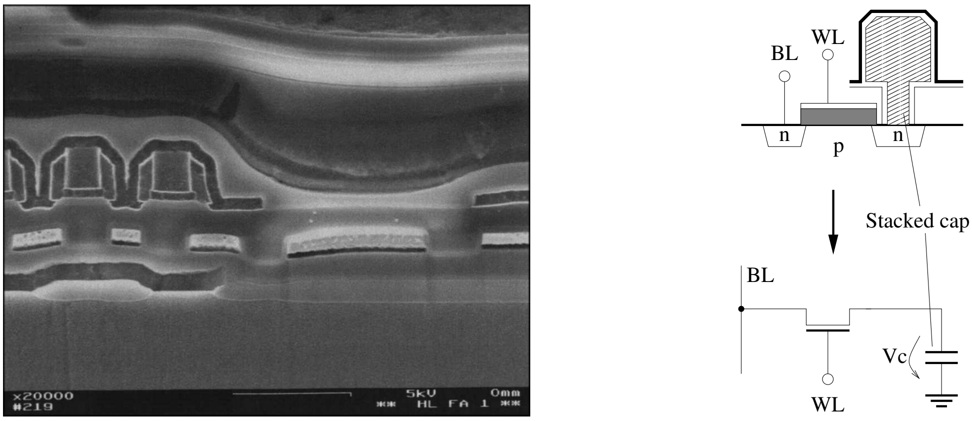
$$\text{aspect ratio} = \frac{\text{trench depth}}{\text{trench diameter}} \tag{3.19}$$

Using current day technology, it is possible to construct so-called *deep trench capacitors*, which have an impressive aspect ratio of 50-60. The diameter of these trenches is referred to as the *critical dimension* (or *CD*) of the trench, which gradually scales down as the minimum feature size decreases. For a 110 nm fabrication technology, the trench CD has a value of about 145 nm, and has a depth of about $8\mu\text{m} = 55 \times 145$ nm (aspect ratio × trench diameter). The CD of such a trench is small enough to fit within the small space of the memory cell, but large enough to ensure the high cell capacitance needed for proper memory operation [Rudolph04]. These deep trenches are manufactured using a challenging process that employs a high-energy ion etch, followed by a polishing etch to achieve an entirely flat silicon surface. This flat surface makes DRAMs that use trench capacitors compatible with a logic manufacturing process that attempts to include an on-chip embedded DRAM (*e*DRAM) module.

The trench capacitor provides an effective solution to the limited surface area problem, but it is in constant industrial competition with another equally effective solution, the so-called stacked capacitor, described next.

3.3.2 DRAM stacked capacitor

The stacked capacitor was born out of need for a capacitor with a large capacitance and a simple manufacturing process at the same time. Instead of using the area under the silicon surface, as it is the case with the trench capacitor, the stacked capacitor uses the space above the wafer surface. Figure 3.15(a) shows a cross-section of silicon where a number of stacked capacitors are clearly visible. Figure 3.15(b) shows a simple sketch of the layout of a 1T DRAM cell that uses the stacked capacitor, and the way this cell is modeled electrically. The bottom capacitor plate is built using a cylindrical polysilicon electrode connected to a self-aligned BL contact, and separated from the top polysilicon plate by an insulator.



(a) Stacked capacitor

(b) Electrical equivalent

Figure 3.15. (a) Cross-section of a stacked capacitor, and (b) the electrical modeling of the layout (source: Infineon Technologies).

In a similar way to the trench capacitor, the aspect ratio concept of the cell capacitor can be applied to the stacked capacitor in the following way:

$$\text{aspect ratio} = \frac{\text{stacked capacitor height}}{\text{capacitor diameter}} \tag{3.20}$$

In the case of the stack capacitor, it is not desirable to have a very high aspect ratio, since the increased height of the capacitor means an increased difference in height between the memory cell array and the surrounding peripheral circuits, making it difficult to construct straight metal wiring on top.

In general, the capacitance of the DRAM cell capacitor can be calculated as follows:

$$C_c = \epsilon_o \epsilon_d \frac{A_p}{d_p} \tag{3.21}$$

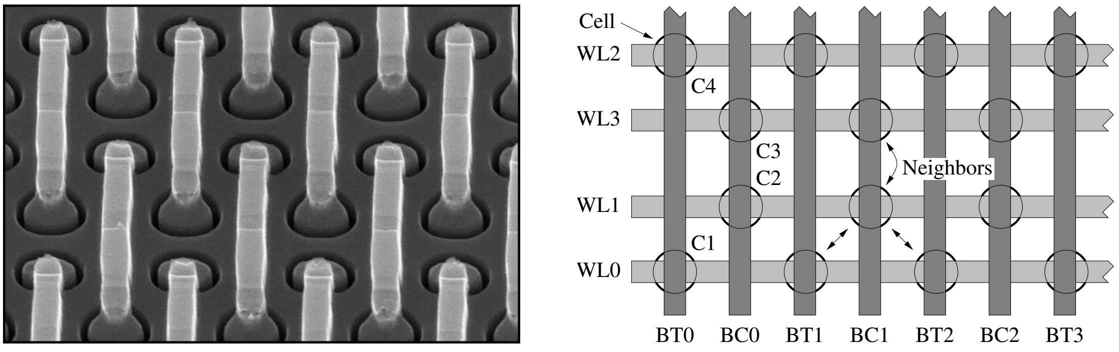
where A_p is the overlap area of the two capacitor plates, d_p is the distance between the two plates, ϵ_o is the permittivity of free space, and ϵ_d is the permittivity of the dielectric isolating material between the plates. Therefore, instead of increasing the aspect ratio, the capacitance of the stacked capacitor can be increased by choosing a dielectric material with a higher ϵ_d. A number of different dielectric materials with high ϵ_d are being considered for future stacked capacitor technology. Since higher ϵ_d dielectrics are difficult to deposit into the rather thin enclaves of the trench capacitor, without having voids created in them, these materials are not as suitable for trench capacitors as they are for stacked capacitors [Cataldo98].

3.3.3 Cell array layout

This section describes the wafer-level organization of the memory cell array, and shows the placement of memory cells on wafer. The wafer-level organization is rather important for memory devices in general, and DRAMs in particular, especially in the cell array region, because of the repetitive nature of the cell array. An organization that succeeds in reducing the average area of the cell even slightly may lead to a significant reduction on the overall area of the cell array, which translates to a reduction in price.

Figure 3.16(a) shows a *scanning electron microscope* (*SEM*) image of a typical layout organization in the cell array region of a current-day, multi-megabit DRAM. The figure clearly shows the repetitive nature of the cells in the memory cell array region of the memory. Figure 3.16(b) shows a sketch of the cell array region, where the memory cells are outlined along with the word lines and bit lines connected to them. This figure shows two important aspects of the layout-level organization of the memory cell array: *cell neighborhood* and *cell address scrambling*.

Cell neighborhood The set of memory cells separated from a given cell by some minimum distance on the layout are called the neighborhood of that cell. The closest neighborhood of a memory cell is rather significant from a manufacturing



(a) SEM image (b) Sketch

Figure 3.16. Layout-level organization of the memory cell array shown in (a) an SEM image, and (b) a layout sketch (source: Infineon Technologies).

point of view, since cells should be kept at a minimum distance from one another to prevent them from interacting with each other. This minimum distance is usually pushed to its last possible limit to achieve the smallest possible cell area. Therefore, the closest neighborhood is also important from a testing point of view, to ensure that different cells are properly isolated, and that no leakage is present between them. In the organization shown in Figure 3.16(b), any given cell has three closest neighboring cells, one connected to the same bit line at a vertical distance from the cell, and two connected to the two adjacent bit lines at a diagonal distance from the cell. As the state-of-the-art fabrication technology changes, and new techniques are introduced to increase the packing density, the cell array organization changes leading to different types of closest cell neighborhoods. As a result, the way a memory is properly tested depends of the specific layout of the memory.

Cell address scrambling The organization of the cell array is optimized to achieve the least possible cell area, and pack as many cells as possible on the surface of the memory chip. In many cases, this requirement makes it necessary to use an internal sequence of cells (called the *physical cell address*) that is different from the external one, as described by regular cell addresses (also referred to as the *logical cell address*). Internal address resequencing is usually referred to as *cell address scrambling*. For many memory tests, the sequence of cells accessed by the test plays a significant role in the effectiveness of the test. Therefore, these tests require using the actual physical cell sequence, rather than the one indicated externally by the logical address sequence of the cells. A memory test that employs the actual physical addressing sequence of the memory is said to use an address descrambler. In Figure 3.16(b), for example, the physical address order of the cells on BL pair 0 is as follows: C1, C2, C3, C4, etc. The logical cell addresses, on the other hand, are shown in Figure 2.9 and have the following sequence: C1, C2, C4, C3, etc. As it is the case for the cell neighborhood, the specific cell address scrambling used for a given memory depends on the design and fabrication technology of the memory.

Summary

This chapter discussed the internal structure and implementation of DRAM chips at both the electrical and the layout levels. The electrical description focused on the data path circuits since they are the most important parts used later in this thesis. The main issues presented in this chapter are the following.

- Introduction to the basics of MOS transistor operation, the fundamental building blocks for memory devices. The behavior has been discussed using the ideal Shockley equations, since they describe the most important aspects of transistor operation.
- Description of the concept of memory signal paths. This concept is important for the electrical analysis of the memory structure, since different paths are, to a large extent, electrically independent from each other. Out of the three presented signal paths (the data path, the address path and the control path), the data path was discussed in detail.
- Detailed analysis of the electrical circuits of the data path. Five circuits have been analyzed: the memory cell, the sense amplifier, the precharge circuits, the access devices, and the data buffer. Simulation output results have been given for some circuits to show how they behave electrically.
- Discussion of the floor plan of the DRAM, which shows the location of different memory circuits on chip. The floor plan has the characteristic cross-shaped layout area, commonly found in multi-megabit DRAMs.
- Presenting two alternatives of the layout level implementation of memory cells (the trench capacitor cell and the stacked capacitor cell), along with the advantages and disadvantages of both.

Contents of this chapter

4.1 Definition of fault models
4.2 Generic space of faults
4.3 DRAM-specific faults
4.4 Industrial test practices

4

Modeling faulty memory behavior

Fault modeling is an important part of memory fault analysis, since it serves as the link between the ever more complex layout level faulty behavior and the seemingly simple structure of memory tests. Proper fault modeling implies simplicity, to keep the fault analysis problem manageable, and comprehensiveness, to be able to represent any type of faulty behavior that might take place. This chapter discusses a memory fault modeling language that is both simple and comprehensive, able to describe any faulty behavior taking place in DRAM devices today.

Section 4.1 starts with a discussion of the concepts of fault modeling and ends with a formal definition of fault models. Section 4.2 describes the generic space of possible faults that may take place in any RAM. It then enumerates a number of important classes of these faults. Section 4.3 develops the space of DRAM-specific fault models, by extending the generic fault space described in Section 4.2 to accommodate for the behavior of DRAMs. Section 4.4 discusses the most important practical aspects of industrial memory testing.

4.1 Definition of fault models

By performing a number of memory operations, and observing the behavior of any component functionally modeled in the memory, *functional fault models* (*FFMs*) can be informally understood as the deviation of the observed memory behavior from the functionally specified one, under a given sequence of performed memory operations. Therefore, two basic ingredients are needed to define any FFM: (1) a sequence of performed memory operations, and (2) a list of corresponding deviations in the observed behavior from the expected one.

4.1.1 Representing operation sequences

Any sequence of performed operations on the memory is called an *operation sequence*. An operation sequence that results in a difference between the observed and the expected memory behavior is called a **sensitizing operation sequence** (S). For example, the operation sequence for an up transition fault (TF_1) in a cell is $S = 0w1$, which requires the cell to be initialized to 0, followed by an attempt to write a 1 into the cell. The observed memory behavior that deviates from the expected one is called a **faulty behavior** or simply a *fault*. For TF_1, the faulty behavior is the inability of the write 1 operation to replace the 0 stored in the cell by a 1.

In order to describe any faulty behavior in the memory, it is important to be able to describe any possible operation sequence performed on the memory. A sensitizing operation sequence must list the *initial data* in the accessed cells and the *operations* performed on them in order to sensitize the fault. The initial data represents the data in the memory cells prior to the start of a test; this may be random (due to power-on, for example) or deterministic (due to a previously applied test). The operations, on other hand, represent operations performed to sensitize the faulty behavior; these can either be writes w or reads r. Therefore, any operation sequence, expected to result in a faulty behavior, can be represented by the following notation:

$$d_{c_1} \ldots d_{c_i} \ldots d_{c_m} \, Od_{c_1} \ldots Od_{c_j} \ldots Od_{c_n} \tag{4.1}$$

where c_x: cell address used,
O: type of operation on c, $O \in \{w, r\}$,
d: data present in c, $d \in \{0, 1\}$,
m: number of initializations, and
n: number of operations.

The initial data is described for m cells (denoted as c_i), while the operations are applied to n cells (denoted as c_j). Note that the value of d in a read operation of the form rd_{c_j} represents the *expected value* of the read operation. This value may be different from the actual read value detected on the output in case of a faulty memory. As an example of the notation, if an operation sequence is denoted by $0_c w1_c r1_c$ then the sequence starts by accessing cell c (which contains a 0) and writing a 1 into it, then reading the written 1.

Sometimes, a fault is sensitized because a cell spontaneously loses its stored state, without the need to perform any operation on the memory. Hence, simply setting the cell into a known initial state is enough to sensitize the fault. This situation can also be described using the operation sequence notation above by limiting S to the initial data and eliminating any performed operation. For example, observing the state of cell c which contains a 0 without accessing it can be denoted by 0_c.

4.1.2 Fault primitives and fault models

The second ingredient needed to specify a fault model is a list of deviations in the observed behavior from the expected one. The only functional parameters considered relevant to the faulty behavior are the stored logic value in the cell and the output value of a read operation. Therefore, any difference between the observed and expected memory behavior can be denoted by the following notation $<S/F/R>$, referred to as a **fault primitive** (**FP**) [vdGoor00]. S describes the operation sequence that sensitizes the fault; F describes the value of the faulty cell, $F \in \{0, 1\}$; and R describes the logic output level of a read operation, $R \in \{0, 1, -\}$. R has a value of 0 or 1 when the fault is sensitized by a read operation, while the "–" is used when a write operation sensitizes the fault. For example, in the FP $<0_c\ w1_c/0/->$, which is a TF_1, $S = 0_c w1_c$ means that cell c is assumed to have the initial value 0, after which a 1 is written into c. The fault effect $F = 0$ indicates that after performing a $w1$ to c, as indicated by S, c remains in state 0. The output of the read operation, $R = -$, indicates that S does not end with a read operation. Since only one cell c is involved in the faulty behavior, the notation for the FP $<0_c\ w1_c/0/->$ can be simplified to $<0w1/0/->_c$, and when the fault can take place in any cell in the memory, FP can be further simplified to $<0w1/0/->$ by eliminating c.

The notion of FPs makes it possible to give a precise definition of an FFM as understood for memory devices. This definition is presented next.

A **functional fault model** (**FFM**) is a non-empty set of fault primitives (FPs).

For example, the FFM called transition fault (TF) consists of a set of FPs that contains the up transition FP ($<0w1/0/->$) and the down transition FP ($<1w0/1/->$). In other words, TF = $\{<0w1/0/->, <1w0/1/->\}$.

4.2 Generic space of faults

As FPs are the building blocks of functional faulty behavior, an analysis of them helps to understand the underlying principles of memory faults, and to appreciate their complexity. Since all components present in the definition of FPs (S, F and R) can be enumerated, the space of FPs can also be enumerated. In the following, a classification of FPs is given first, followed by a discussion of the most important FPs (static single-cell and static two-cell FPs) along with their properties.

4.2.1 Classification of fault primitives

FPs can be classified into different classes, depending on S, as shown in Figure 4.1. Let $\#C$ be the total number of *different* memory cells initialized (c_i) and accessed (c_j) in S, and let $\#O$ be the number of operations (w or r) performed in S. For example, if $S = 0_{c_1}\ 0_{c_2}\ w1_{c_2}$ then $\#C = 2$ since two cells (c_1 and c_2) are present in S, while $\#O = 1$ since only one operation is performed ($w1$ to c_2).

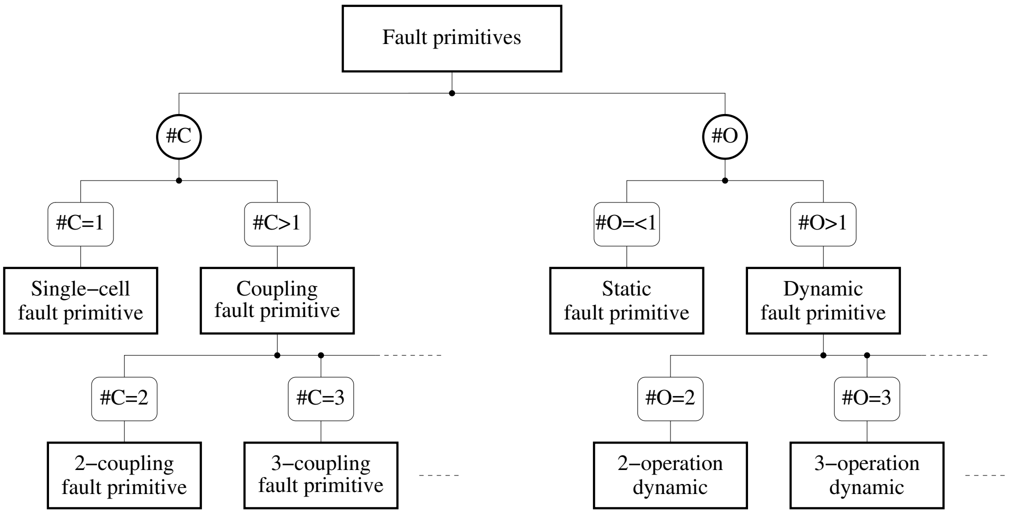


Figure 4.1. Taxonomy of fault primitives.

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

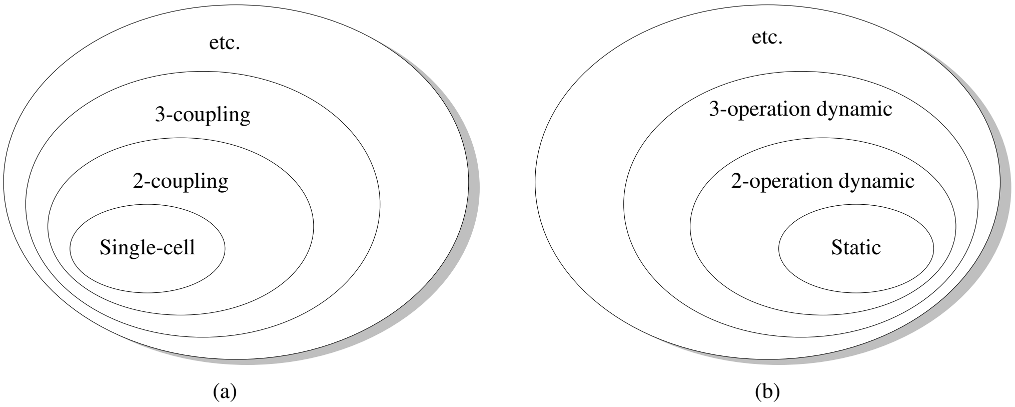
- If $\#C = 1$, then the FP sensitized by the corresponding S is called a *single-cell FP*.
- If $\#C > 1$, then the FP sensitized by the corresponding S is called a *coupling FP*. If $\#C = 2$ then it is described as a *two-coupling FP* or a *two-cell FP*. If $\#C = 3$ then it is described as a *3-coupling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

In case an FP is a coupling FP ($\#C > 1$), then one of the cells in the S should be considered as a **victim** (v) while the other cells are considered as **aggressors** (a). In any FP, the described faulty behavior is related to the victim, while the aggressors are considered to contribute to the fault.

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

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

Clearly, a hierarchy in S results in a hierarchy in FPs. Figures 4.2(a) and (b) represent the two different types of hierarchies in FPs defined according to $\#C$ and $\#O$, respectively. As shown in the figure, two-coupling FPs are higher in the hierarchy than single-cell FPs, and the larger the number of coupled cells the higher the hierarchical level a fault primitive has. In the same way, a dynamic FP is higher in the hierarchy than a static FP, and the larger the number of operations of a dynamic FP the higher it becomes in the hierarchy. A higher hierarchical level involves more cells and/or more operations, hence has a higher test cost.



(a) (b)

Figure 4.2. Hierarchical organization of FP classes defined according to (a) $\#C$ and (b) $\#O$.

4.2.2 Static fault models

Static fault models consist of static FPs, sensitized by at most a single memory operation. In other words, these are faults that describe an incorrect behavior of either the data stored in a cell, or a single memory operation performed on it. In order to describe static fault models, it is not necessary to use the general notation for the sensitizing operation sequence presented above, but the following notation is sufficient: d_{c_1} ... d_{c_i} ... d_{c_m} Od_c, where at most one operation (O) is performed on cell c. In the following, the two most important classes of static faults are described, single-cell static faults and two-cell static faults.

Single-cell static faults

As mentioned earlier, a particular fault primitive is denoted by $<S/F/R>$. S describes the value or operation that sensitizes the fault, and for single-cell static fault primitives $S \in \{0, 1, 0w0, 0w1, 1w0, 1w1, 0r0, 1r1\}$. F describes the logic value stored in the faulty cell; $F \in \{0, 1\}$. R describes the logic output value of a read operation, in case a read is the operation that sensitizes the fault; $R \in \{0, 1, -\}$. A "$-$" in the notation means that the output data is not applicable to the associated sensitizing operation. For example, if $S = 0w1$, which defines a write operation, no data is expected at the memory output, and therefore R is replaced by a "$-$".

Now that the possible values for S, F and R are defined for single-cell static FPs, it is possible to list them all using this notation. Table 4.1 lists all possible combinations of the values, in the $<S/F/R>$ notation, that result in FPs. The remaining combinations of the S, F and R values do not represent a faulty behavior. For example, $<0w0/0/->$ corresponds to a correct $w0$ operation after which the cell contains a 0 as expected. The column "Fault model" of Table 4.1 states the fault model name defined by the corresponding fault primitive. The table lists the following fault models.

Table 4.1. List of single-cell, static FPs in the $<S/F/R>$ notation.

#	S	F	R	$<S/F/R>$	Fault model
1	0	1	–	$<0/1/->$	State-0 fault (SF_0)
2	1	0	–	$<1/0/->$	State-1 fault (SF_1)
3	$0w0$	1	–	$<0w0/1/->$	Write-0 destructive fault (WDF_0)
4	$0w1$	0	–	$<0w1/0/->$	Up transition fault (TF_1)
5	$1w0$	1	–	$<1w0/1/->$	Down transition fault (TF_0)
6	$1w1$	0	–	$<1w1/0/->$	Write-1 destructive fault (WDF_1)
7	$0r0$	0	1	$<0r0/0/1>$	Incorrect read-0 fault (IRF_0)
8	$0r0$	1	0	$<0r0/1/0>$	Deceptive read-0 destructive fault ($DRDF_0$)
9	$0r0$	1	1	$<0r0/1/1>$	Read-0 destructive fault (RDF_0)
10	$1r1$	0	0	$<1r1/0/0>$	Read-1 destructive fault (RDF_1)
11	$1r1$	0	1	$<1r1/0/1>$	Deceptive read-1 destructive fault ($DRDF_1$)
12	$1r1$	1	0	$<1r1/1/0>$	Incorrect read-1 fault (IRF_1)

1. **State fault (SF_x)**—A cell is said to have a *state fault* if the logic value of the cell changes before it is accessed (read or written), even if no operation is performed on it[1]. This fault is special in the sense that no operation is needed to sensitize it and, therefore, it only depends on the initial stored value in the cell. There are two types of state faults:

   - State-0 fault (SF_0) = $\{<0/1/->\}$, with FP #1
   - State-1 fault (SF_1) = $\{<1/0/->\}$, with FP #2

   S, in the fault primitive notation, denotes the initial value of the cell, which is either 0 or 1. Initializing the cell is sufficient to sensitize the fault, since the cell immediately flips to the opposite logic value. F is represented by the opposite value to the initialization. The value of R is "–", which denotes that there is no expected output.

---

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

2. **Transition fault ($\mathbf{TF}_x$)**—A cell is said to have a *transition fault* if it fails to undergo a transition ($0 \rightarrow 1$ or $1 \rightarrow 0$) when it is written. This fault is sensitized by a write operation and depends on both the initial stored logic level and the type of the write operation. There are two types of transition faults:

    - Up transition fault (TF_1) = $\{<0w1/0/->\}$, with FP #4
    - Down transition fault (TF_0) = $\{<1w0/1/->\}$, with FP #5

    S, in the fault notation, denotes a write operation that tries to change the value of the cell. The failure to change the value of the cell constitutes the fault. Therefore, F is represented by the initial stored logic value, while the value of R is "–", which denotes that there is no expected output.

3. **Read destructive fault ($\mathbf{RDF}_x$)**—A cell is said to have a *read destructive fault* if a read operation performed on the cell changes the data in the cell and returns an incorrect value on the output. There are two types of read destructive faults:

    - Read-0 destructive fault (RDF_0) = $\{<0r0/1/1>\}$, with FP #9
    - Read-1 destructive fault (RDF_1) = $\{<1r1/0/0>\}$, with FP #10

    S, in the fault notation, denotes either a read 0 or a read 1 operation, which is the operation that sensitizes the fault. F is represented by the faulty value that r sets into the cell, while R denotes the faulty read value on the output.

4. **Write destructive fault ($\mathbf{WDF}_x$)**—A cell is said to have a *write destructive fault* if a non-transition write operation ($0w0$ or $1w1$) causes a transition in the cell. There are two types of write destructive faults:

    - Write-0 destructive fault (WDF_0) = $\{<0w0/1/->\}$, with FP #3
    - Write-1 destructive fault (WDF_1) = $\{<1w1/0/->\}$, with FP #6

    This fault is similar to the transition fault, where a write operation fails to function properly. S, in the fault notation, stands for a failing write operation, F is represented by a faulty transition, while the value of R is "–" which means that there is no expected output.

5. **Incorrect read fault ($\mathbf{IRF}_x$)**—A cell is said to have an *incorrect read fault* if a read operation performed on the cell returns the incorrect logic value, while keeping the correct stored value in the cell. There are two types of incorrect read faults:

    - Incorrect read-0 fault (IRF_0) = $\{<0r0/0/1>\}$, with FP #7
    - Incorrect read-1 fault (IRF_1) = $\{<1r1/1/0>\}$, with FP #12

S, in the fault notation, denotes either a read 0 or a read 1 operation, which is the operation that sensitizes the fault. F represents the correct stored value in the cell, while R represents the faulty memory output.

6. **Deceptive read destructive fault (DRDF$_x$)**—A cell is said to have a *deceptive read destructive fault* if a read operation performed on the cell returns the correct logic value, while changing the contents of the cell [Adams96]. There are two types of deceptive read disturb faults:
    - Deceptive $r0$ destructive fault (DRDF$_0$) = $\{<0r0/1/0>\}$, with FP #8
    - Deceptive $r1$ destructive fault (DRDF$_1$) = $\{<1r1/0/1>\}$, with FP #11

    S, in the fault notation, is denoted by either a read 0 or a read 1 operation, which is the operation that sensitizes the fault. F represents an incorrect stored value in the cell, while R represents the correct memory output.

From the above list, two observations can be made about the capability of this notation to describe different sorts of single cell static faulty behavior.

- A write operation is capable of sensitizing 4 FPs.
- A read operation is capable of sensitizing 6 FPs.

In total, if precisely one operation is performed, then 10 FPs can be sensitized.

Two-cell static faults

In order to describe two-cell static fault models, it is not necessary for S to be represented by the general notation of the sensitizing operation sequence shown in Expression 4.1, since two-cell faults can be described by a simplified notation. Since the FPs are static, at most one operation should be performed. Moreover, two different cells need to be considered in order to sensitize a two-cell fault. Therefore, a two-cell static FP can be represented as follows $<S/F/R> = <S_a; S_v/F/R>_{a,v}$. Table 4.2 enumerates all possible two-cell static FPs this notation can distinguish. Only the combinations that describe a fault in the memory are listed, since the remaining combinations of the S, F and R values do not represent any faulty behavior. The table indicates the presence of 36 FPs, a lot more than the number of single-cell FPs. The table lists the following faults.

1. **State coupling fault (CFst)**—Two cells are said to have a *state coupling fault* if the victim is forced into a given logic state only if the aggressor is in a given state, without performing any operation on the victim. This fault is special in the sense that no operation is needed to sensitize it and, therefore, it only depends on the initial stored values in the cells. There are four state coupling FPs: CFst$_{0;0}$ = $\{<0;0/1/->\}$ with FP #1, CFst$_{0;1}$ = $\{<0;1/0/->\}$ with FP #2, CFst$_{1;0}$ = $\{<1;0/1/->\}$ with FP #3 and CFst$_{1;1}$ = $\{<1;1/0/->\}$ with FP #4.

Table 4.2. Combinations of $<S/F/R>$ values that result in two-cell static fault primitives.

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

2. **Disturb coupling fault (CFds)**—Two cells are said to have a *disturb coupling fault* if an operation (write or read) performed on the aggressor forces the victim into a given logic state. Here, any operation performed on the aggressor is accepted as a sensitizing operation for the fault, be it a read, a transition write or a non-transition write operation. There are 12 disturb coupling FPs: $\text{CFds}_{0w0;0} = \{<0w0;\ 0/1/->\}$ with FP #5, $\text{CFds}_{0w0;1} = \{<0w0;\ 1/0/->\}$ with FP #6, $\text{CFds}_{1w1;0} = \{<1w1;\ 0/1/->\}$ with FP #11, $\text{CFds}_{1w1;1} = \{<1w1;\ 1/0/->\}$ with FP #12, $\text{CFds}_{0w1;0} = \{<0w1;\ 0/1/->\}$ with FP #7, $\text{CFds}_{0w1;1} = \{<0w1;\ 1/0/->\}$ with FP #8, $\text{CFds}_{1w0;0} = \{<1w0;\ 0/1/->\}$ with FP #9, $\text{CFds}_{1w0;1} = \{<1w0;\ 1/0/->\}$ with FP #10, $\text{CFds}_{0r0;0} = \{<0r0;\ 0/1/->\}$ with FP #13, $\text{CFds}_{0r0;1} = \{<0r0;\ 1/0/->\}$ with FP #14, $\text{CFds}_{1r1;0} = \{<1r1;\ 0/1/->\}$ with FP #15 and $\text{CFds}_{1r1;1} = \{<1r1;\ 1/0/->\}$ with FP #16.

3. **Transition coupling fault (CFtr)**—Two cells are said to have a *transition coupling fault* if the state of the aggressor results in the failure of a transition write operation performed on the victim. This fault is sensitized by a write operation on the victim, while the aggressor is in a given state. There are four transition coupling FPs: $\text{CFtr}_{0;0} = \{<0;\ 0w1/0/->\}$ with FP #19, $\text{CFtr}_{0;1} = \{<0;\ 1w0/1/->\}$ with FP #21, $\text{CFtr}_{1;0} = \{<1;\ 0w1/0/->\}$ with FP #20 and $\text{CFtr}_{1;1} = \{<1; 1w0/1/->\}$ with FP #22.

4. **Write destructive coupling fault (CFwd)**—A cell is said to have a *write destructive coupling fault* if a non-transition write operation performed on the victim results in a transition, while the aggressor is in a given logic state. There are four write destructive coupling FPs: $\text{CFwd}_{0;0}$ = {<0; $0w0/1/-$>} with FP #17, $\text{CFwd}_{0;1}$ = {<0; $1w1/0/-$>} with FP #23, $\text{CFwd}_{1;0}$ = {<1; $0w0/1/-$>} with FP #18 and $\text{CFwd}_{1;1}$ = {<1; $1w1/0/-$>} with FP #24.

5. **Read destructive coupling fault (CFrd)**—Two cells are said to have a *read destructive coupling fault* if a read operation performed on the victim destroys the data stored in the victim, while a given state is present in the aggressor. There are four read destructive coupling FPs: $\text{CFrd}_{0;0}$ = {<0; $0r0/1/1$>} with FP #29, $\text{CFrd}_{0;1}$ = {<0; $1r1/0/0$>} with FP #31, $\text{CFrd}_{1;0}$ = {<1; $0r0/1/1$>} with FP #30 and $\text{CFrd}_{1;1}$ = {<1; $1r1/0/0$>} with FP #32.

6. **Incorrect read coupling fault (CFir)**—Two cells are said to have an *incorrect read coupling fault* if a read operation performed on the victim returns the incorrect logic value, while the aggressor is in a given state. There are four incorrect read coupling FPs: $\text{CFir}_{0;0}$ = {<0; $0r0/0/1$>} with FP #25, $\text{CFir}_{0;1}$ = {<0; $1r1/1/0$>} with FP #35, $\text{CFir}_{1;0}$ = {<1; $0r0/0/1$>} with FP #26 and $\text{CFir}_{1;1}$ = {<1; $1r1/1/0$>} with FP #36.

7. **Deceptive read destructive coupling fault (CFdr)**—A cell is said to have a *deceptive read destructive coupling fault* if a read operation performed on the victim returns the correct logic value and changes the contents of the victim, while the aggressor is in a given logic state. There are four deceptive read destructive coupling FPs: $\text{CFdr}_{0;0}$ = {<0; $0r0/1/0$>} with FP #27, $\text{CFdr}_{0;1}$ = {<0; $1r1/0/1$>} with FP #33, $\text{CFdr}_{1;0}$ = {<1; $0r0/1/0$>} with FP #28 and $\text{CFdr}_{1;1}$ = {<1; $1r1/0/1$>} with FP #34.

4.2.3 Complexity of fault primitives

As operations are added to S, in order to investigate the dynamic faulty behavior of the memory, the possible number of different Ss, and the associated number of dynamic FPs, increases rapidly. For a single-cell FP, S typically starts with an initialization of either 0 or 1, followed by one of three possible memory operations $w0$, $w1$ or r for each increment in $\#O$. As a result, the possible number of different Ss can be calculated by [Al-Ars99]:

$$\#S = 2 \cdot 3^{\#O} \tag{4.2}$$

When no operation is performed ($\#O = 0$) the number of possible FPs is 2, and when one or more operations are performed, S is able to sensitize 10 different FPs for each increment in $\#O$ [see Table 4.1], as summarized in the following relation:

$$\#\text{single-cell FPs} = \begin{cases} 2 & : \#O = 0 \\ 10 \cdot 3^{(\#O-1)} & : \#O \geq 1 \end{cases} \tag{4.3}$$

These equations indicate an exponential relationship between $\#S$ and $\#O$ on the one hand, and between #FP and $\#O$ on the other. Therefore, a straightforward attempt to investigate the dynamic faulty behavior of the memory, by directly applying all possible Ss, is limited by practical analysis time and available computing power. Using the analysis approach presented in Chapter 5, the total *infinite* space of dynamic faulty behavior can be approximated within a short amount of analysis time.

4.2.4 Fault primitive detection and march tests

In order to inspect memory devices for possible faulty behavior, memory testing is performed on all produced memory components. Many types of memory tests are being used today, each with its own advantages and disadvantages. March tests are among the most popular memory tests, due to their low complexity and high fault coverage.

Definition of march tests

The idea of march tests is to construct a number of operation sequences and to perform each sequence on all memory cells, one after the other, before performing the following sequence in the test. Therefore, a **march test** can be defined as a sequence of march elements, where a **march element** is a sequence of memory operations performed sequentially on all memory cells. In a march element, the way one proceeds from one cell to the next is specified by the *address order*, which can be increasing (denoted by $\Uparrow$) or decreasing (denoted by $\Downarrow$). The $\Downarrow$ address order has to be the exact opposite of the $\Uparrow$ address order. For some march elements, the address order can be chosen arbitrarily as increasing or decreasing, which is denoted by the $\Updownarrow$ symbol. In a march element, it is possible to perform a write 0 operation ($w0$), write 1 ($w1$), read 0 ($r0$) and read 1 ($r1$) operation. The 0 and 1 after read operations represent the expected values of the read on the output. An example of a march element is $\Uparrow(r0, w1)$, where all memory cells are accessed in an increasing address order while performing $r0$ then $w1$ on each cell, before continuing to the next cell.

By arranging a number of march elements one after the other, a march test is constructed. An example of a march test is $\{\Updownarrow(w0); \Uparrow(r0, w1); \Downarrow(r1, w0)\}$, which is the well known march test called MATS+ [vdGoor98]. It consists of three march elements denoted as M0, M1 and M2. The test begins with writing 0 into all memory cells in an increasing or decreasing order, then on each cell a read 0 and a write 1 operation is performed in an increasing order, and finally on each cell a read 1 and a write 0 operation is performed in a decreasing order.

Generation of march tests

The analytical approach to memory testing begins with an analysis of the faulty behavior of the memory, which is then described by a number of FPs, that give an exact description of the way the faulty behavior is sensitized. These FPs are used to identify so-called **detection conditions**, which describe an incomplete set of march elements specifying the minimum requirements a march test has to fulfill in order to detect the faulty behavior. Detection conditions can easily be used to generate the necessary memory tests to detect the observed faulty behavior.

As an example, assume that the fault analysis of a given memory indicates that the memory suffers from an up transition fault $\mathrm{TF}_1 = \{<0w1/0/->\}$. The FP gives a precise description of the way the observed fault can be sensitized. In order to detect this fault, a march test needs to fulfill the detection condition $\Updownarrow$(..., $w0$, ...) $\Updownarrow$(..., $w1$, ...); $\Updownarrow$(..., $r1$, ...), which states that the test has to initialize all cells to 0, then perform an up transition write operation on each cell, and finally it has to read the written 1 from all cells.

It is possible to generate many different march tests that satisfy this detection condition. Here are some examples:

- $\{\Updownarrow(w0, w1, r1)\}$
- $\{\Updownarrow(w0); \Updownarrow(w1, r1)\}$
- $\{\Updownarrow(w0); \Updownarrow(w1); \Updownarrow(r1)\}$
- $\{\Uparrow(w0, w1, r1)\}$
- $\{\Uparrow(w0); \Downarrow(w1, r1)\}$
- etc.

Multiple detection conditions needed to detect a number of different FPs in the faulty behavior can also be used to generate a single march test to fully test the memory for possible faulty behavior.

4.3 DRAM-specific faults

As discussed in the previous section, the space of memory faults is infinitely large, which makes it impossible to inspect a given memory for all possible faults. Instead, every memory is inspected for a limited number of relevant memory-specific faults that represent a part of the full space of faults. This section introduces a classification of the space of memory faults specific to DRAMs, along with a discussion of each class. The classification takes *time* into consideration, an aspect that is not modeled in the generic fault space above.

Any memory fault can be represented by the FP notation $<S/F/R>$, where S stands for the sensitizing operation sequence that results in sensitizing the fault. In

DRAMs, however, S can be divided into two parts: the *initialization part* (I) and the *sensitization* or *fault activation part* (A). I represents the initial data present in the cells, along with the operations performed to ensure that all relevant cells are set to known predefined states. A is the sequence of operations needed to sensitize (or activate) the fault. Therefore, a DRAM fault primitive is denoted by FP = $<S/F/R>$ = $<IA/F/R>$. In terms of time, the operations performed in I are supposed to take place before the operations performed in A. Using this idea, it is possible to identify the part of S that represents I, and the part that represents A, for the single-cell and two-cell static faults defined in Section 4.2, as follows [see Expression 4.1]:

- If $\#O = 0$, or in other words, if state faults or state coupling faults are taking place, then there is no sensitization part, and $S = I$.
- If $\#O = 1$, and S has only one operation, then A consists of this operation ($A = O_{c_1}$), while I consists of the initializations of S ($I = d_{c_1}$ for single-cell faults and $I = d_{c_1}d_{c_2}$ for two-cell faults).

The classification of DRAM-specific faults is based on the concepts of the initialization and the sensitization parts of S, as described in Table 4.3. As the operations in S are being performed, DRAM faults may take place in four different stages: during I, between I and A, during A, and after A. DRAM faults have two main causes:

- improperly set voltages resulting in voltage dependent faults, and
- leakage currents resulting in time dependent faults.

Faults caused by improper voltages stem from the inability of a memory operation in a defective memory to set the full voltage levels expected at different nodes of the memory, resulting in two different fault modes: 1. improper voltages present within the memory cell, and 2. improper voltages on the nodes of the peripheral circuits. Improper voltages within the cell cause *partial faults*, while improper voltages in periphery cause *dirty faults*.

Leakage currents, on the other hand, cause time dependent faults to take place, and depending on the direction of the leakage with respect to the performed operation, either *soft faults* or *transient faults* take place due to a supporting or an opposing leakage current, respectively. In the following, these DRAM-specific faults are discussed in more detail.

4.3.1 Voltage dependent faults

In a DRAM, operations are supposed to properly set the voltage levels on different nodes (cells or bit lines, for example) in the memory by charging or discharging the capacitors, corresponding to that node, to a predefined high or low voltage level. In general, however, a voltage across a capacitor may take any value from a

Table 4.3. Classification of DRAM-specific faults.

Cause of problem	Mode of problem	Resulting fault model			
		During I	Between I & A	During A	After A
Improper voltages (voltage dependent)	Within cell	Partial faults	—	Partial faults	—
	In periphery	—	Dirty faults	—	Dirty faults
	In cells & periphery	Partial faults	Dirty faults	Partial faults	Dirty faults
Leakage currents (time dependent)	Supports operation	Transient faults			
	Opposes operation	Soft faults			

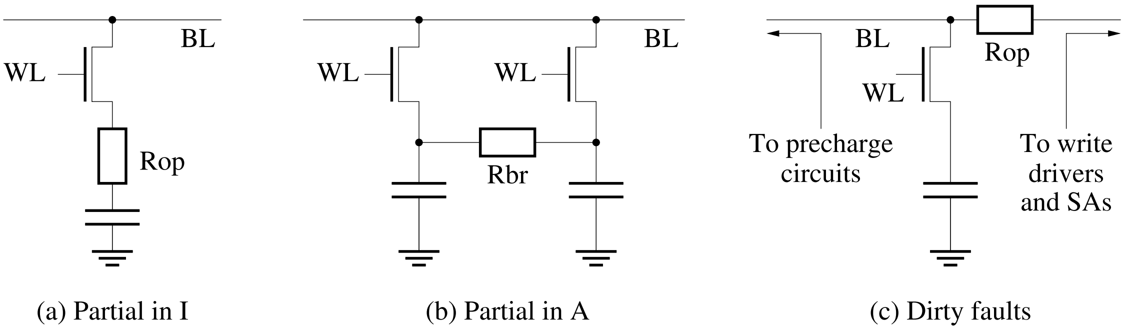
continuous range of real voltages. Therefore, operations performed on a defective memory may set improper voltage levels on different memory nodes, and therefore require a special sequence of operations to ensure setting them properly. Improper voltages may cause two types of DRAM faults: partial faults and dirty faults.

Partial faults

Definition—These are faults that can only be sensitized when a specific memory operation is successively repeated a number of times, either to properly initialize the faulty cell, or to properly sensitize the fault in the cell [Al-Ars02a]. The definition indicates that there are two different sorts of partial faults: 1. *partial faults during initialization* I, and 2. *partial faults during sensitization* A.

Root-cause—Partial faults result from the fact that DRAM cells represent data as analog voltages in a storage capacitor that is supposed to be fully charged or discharged after a single write operation has been performed on a defect-free cell. Figure 4.3(a) shows an example of an open (R_{op}) in the cell, causing a partial fault in I. R_{op} prevents fully initializing the cell to the required voltage with only one operation, which means that full initialization requires repeating the operation a number of times. Figure 4.3(b) shows an example of a bridge (R_{br}) between two cells, causing a partial fault in A. When a cell is written, R_{br} forces the other cell to be written as well, and for sufficiently large values of R_{br}, multiple operations are needed to sensitize the partial fault in A.

Fault modeling—This is achieved by performing a special sequence of operations in I or in A. Partial faults assume that in order to ensure a proper voltage in a given cell, an operation Ox should be performed an h (or *hammer*) number of times, to ensure sensitizing the fault. Therefore, partial faults require that any Ox operation



(a) Partial in I (b) Partial in A (c) Dirty faults

Figure 4.3. Defects causing (a) partial faults in I, (b) in A, and (c) causing dirty faults.

be substituted by Ox^h. For example, if a single-cell fault of the form $<xOy/F/R>$ becomes partial in A, it should be modeled as $<xOy^h/F/R>$, which means that repeating the sensitizing operation of A on the cell multiple times causes a fault.

Dirty faults

Definition—These faults assume that after proper initialization or sensitization, the state of the memory (voltages on the BLs, the WLs, or in data buffers) is corrupted, such that subsequent detection is prevented. In order to ensure that the sensitized fault is detectable, additional operations must be performed to correct the corrupted state of the memory [Al-Ars01c].

Root-cause—These faults result from the fact that writes and reads are complex operations, requiring a properly set environment in the memory to function correctly. A number of memory defects may result in disrupting this balanced environment. Figure 4.3(c) shows an example of an open defect (R_{op}) on the BL that causes dirty faults. This defect disconnects memory cells from the write drivers, thereby limiting the ability of the memory to properly write the cells. At the same time, this defect disconnects the precharge devices from part of the BL, which prevents properly precharging the BL. As a result, a $w0$ operation that fails to write 0 in the cell may end up preconditioning the BL to properly sense a 0 in a subsequent read operation, thereby preventing the detection of the faulty $w0$ and causing a dirty fault [Al-Ars02a].

Fault modeling—This is achieved by the introduction of *completing operations* to the FP, which need to be performed after the initialization (I) or after the sensitization (A) part of S. Assuming that I_o represents the original initialization part of S, and that C represents the *completing operation sequence*, then the completed initialization is represented by $I = I_o[C]$. In the same way, the completed sensitization part of S is represented by $A = A_o[C]$, where A_o is the original sensitization part of S. There are two different defects known to cause dirty faults in DRAMs, one is an open on the BL, which results in improperly set BL voltages, and the other is an open in the sense amplifier, which results in improperly set data buffers

[Al-Ars02a]. Both defects cause dirty faults that can be detected using a write completing operation with data opposite to the data in the victim, performed to a cell different from the victim but positioned on the same BL. This gives the following FP: $<xO_vy[w_a\overline{y}]/\overline{y}/->_{a,v\in\mathrm{BL}}$.

4.3.2 Time dependent faults

Time dependent faulty behavior in DRAMs (sometimes in other memories as well) cannot be described using the generic fault space discussed in Section 4.2. This section presents an extended FP notation that is able to take time dependent faults into consideration. This is done using a special classification of faults, based on leakage currents, which divides faults into: hard faults, soft faults and transient faults. The section starts with a basic description of DRAM functionality, followed by a classification of FPs according to their time dependency [Al-Ars04a].

Basic DRAM functionality

As shown in Figure 4.4, a DRAM cell consists of an access transistor controlled by the WL, which connects the BL to a cell capacitor. A DRAM cell stores its logic value in a leaky storage capacitor, which is not directly connected to a power supply node, a fact that results in the gradual loss of the stored charge in the capacitor. In order to store data in a DRAM for a long period of time, the stored data in the cell needs to be refreshed regularly to prevent the total depletion of stored information. There are many causes of leakage current, some pull the cell voltage up, while others pull the cell voltage down, such that the net voltage change within the cell is determined by the net effect of all active leakage mechanisms for a given memory cell [Keshavarzi97].

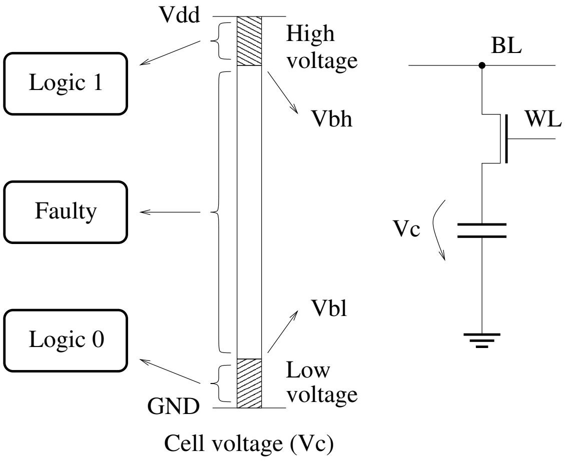


Figure 4.4. Basic structure and ranges of stored voltages of a DRAM cell.

Directly after a write operation, the voltage in the capacitor should be set to a high (or a low) enough level that allows enough time before the voltage is completely destroyed by leakage. Therefore, it is possible to divide stored cell voltages, present directly after performing a write operation, into three regions [see Figure 4.4]. The "logic 1" region directly after performing a write 1 operation, the "logic 0" region directly after performing a write 0, and the "faulty" region directly after performing a faulty write 1 or 0 operation. The logic 1 region extends from the high power supply voltage (V_{dd}) to the border high voltage (V_{bh}), the logic 0 region extends from GND to the border low voltage (V_{bl}), while the faulty region takes on the voltages between V_{bh} and V_{bl}.

The exact values of the voltages in the figure depend on a number of factors, such as the specific memory design, the fabrication technology and the type and severity of the defect in a defective memory. As an example, these voltages can assume the following values in a defect-free memory: $V_{dd} = 2.5$ V, $V_{bh} = 95\% \cdot V_{dd} = 2.38$ V, $V_{bl} = 5\% \cdot V_{dd} = 0.13$ V.

Assume that a defective memory cell has a net leakage current that pulls the cell voltage down, then the voltage ranges corresponding to a stored logic 0 and stored logic 1, directly after performing a write operation, are shown in Figure 4.5. The figure shows that, in addition to the region of proper operation, there are three faulty regions: the hard fault region, the soft fault region and the transient fault region. In the following, each of the three types of faults is described in more detail.

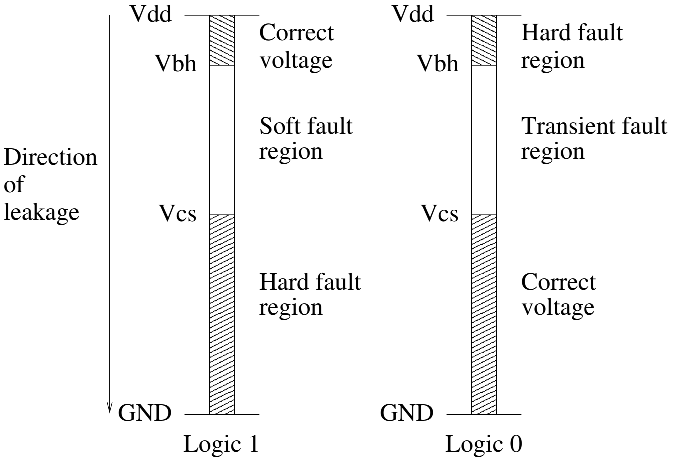


Figure 4.5. Voltage ranges in DRAM cells directly after an operation.

Hard faults

Definition—Hard faults are memory faults that do not depend on time in any way, neither for sensitization nor for detection. They are directly sensitized after performing a number of memory operations, without the need to wait for a while for the faulty behavior to take effect. Furthermore, once they are sensitized they remain sensitized unless overwritten, giving unlimited amount of time for their subsequent detection.

Root-cause—These faults result from defects in the memory that prevent proper functionality under all circumstances for a specific sequence of memory operations. Figure 4.5 shows the region of hard fault voltages, directly after performing a write operation on a DRAM cell having a net leakage current that gradually pulls the cell voltage down. The voltage level V_{cs} in the figure is the *cell sense threshold voltage*, above which the sense amplifier detects a stored logic 1, and below which the sense amplifier detects a stored 0. Therefore, in the column representing logic 1 in the figure, any voltage below V_{cs} is considered as a hard fault, since it can be directly detected by a read operation. In the column representing logic 0 in the figure, the hard fault range is between V_{dd} and V_{bh}, since it takes more time than the refresh time to deplete the faulty charge in the cell to a detectable 0.

Fault modeling—Since hard faults should not depend on time for their sensitization nor detection, neither S nor F, in the FP notation of a hard fault, should have a time parameter attached to it. This is identical to the FP description of any generic fault, as listed in Tables 4.2 and 4.1, as all of these faults are considered to be time independent.

Examples—Figure 4.6 shows examples of memory defects that generate hard faults for both DRAM and SRAM cells. For the DRAM cell, the open defect (R_{op}) between the pass transistor and the cell capacitor restricts current flow to and from the cell and prevents write operations from changing the value stored in the cell, thereby causing an up transition fault ($<0w1/0/->$) as well as a down transition fault ($<1w0/1/->$) for high open resistance values [Al-Ars01a]. The transition faults here represent hard faults since they are sensitized once the write operations are performed. For the SRAM cell, the open defect (R_{op}) at the gate of the pull-down transistor causes a down transition fault ($<1w0/1/->$) when the open resistance has a high value [Hamdioui00]. This type of fault requires a write operation, and will always be sensitized once the faulty operation is performed.

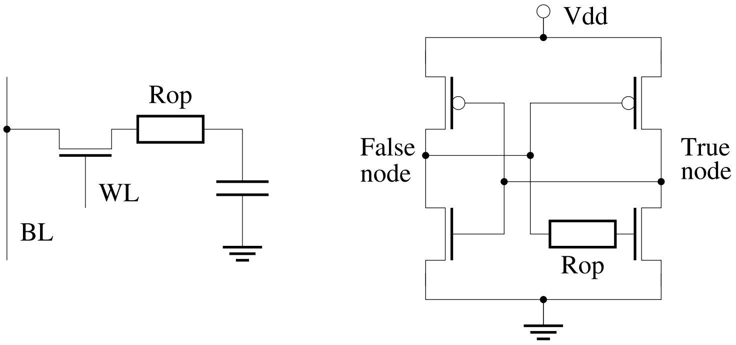


Figure 4.6. Opens in DRAM and SRAM cells causing hard as well as soft faults, each for a specific range of defect resistances.

Soft faults

Definition—Soft faults are memory faults sensitized by a sequence of memory operations, that only become detectable after some time from their sensitization. These faults have usually been tested for by the addition of a *delay* within the test, to facilitate the detection of the fault, as it is the case for the *data retention fault*, for example [Dekker90].
Root-cause—Soft faults are caused by writing weak voltages into memory cells, that soon get depleted by naturally occurring leakage currents. Figure 4.5 shows the region of soft fault voltages within a DRAM cell having a net leakage current to GND. With such leakage current, soft faults take place after a $w1$ operation that sets a cell voltage between V_{bh} and V_{cs}, since this faulty voltage sensitizes a fault when time passes to deplete the voltage to a level below V_{cs}.
Fault modeling—In terms of the FP notation, soft faults are represented as $<S_T/F/R>$, where the sensitizing operation sequence has an added time parameter T to indicate that some time should first elapse before the fault effect is completely sensitized. The fact that *time* is needed to sensitize soft faults is particularly frustrating for test designers, since test time is a major cost factor in the whole process of memory testing. Therefore, soft faults are primarily tested for using specialized *design-for-testability* (*DFT*) techniques or using stresses rather than introducing idle time into the testing process, as discussed later in Section 4.4.
Examples—Figure 4.6 shows examples of memory opens that cause soft faults in both DRAM and SRAM cells. For the DRAM cell, if the open defect has an intermediate resistance value that is not too high (causing hard faults) and not too low (not causing a fault at all), write operations succeed but are only able to write a *weak voltage* into the cell. As time passes, and due to naturally occurring leakage in DRAM cells, a weakly written voltage is depleted gradually, thereby losing the stored information over time. The assumed net leakage current toward GND results in soft faults associated with weak logic 1 voltages, while a net leakage current toward V_{dd} results in soft faults associated with weak logic 0 voltages. Therefore, this defect causes what we may call *soft transition faults*, and using the FP notation, these can be written as $<0w1_T/0/->$ and $<1w0_T/1/->$. The SRAM open shown in Figure 4.6 is located at the gate of the pull-down transistor, a defect position that (in combination with a low floating gate voltage) degrades the ability of GND to compensate for leakage currents through the pull-up transistors. Therefore, a stored 0 voltage on the defective side of the cell will be gradually degraded through leakage, until the cell flips and looses its stored voltage a while later. This faulty behavior is referred to as data retention fault, and has the FP notation $<0_T/1/->$.

Transient faults

Definition—Transient faults are memory faults that do not remain sensitized indefinitely, but tend to correct themselves after a period of time [Al-Ars01b]. Transient faults are tested for by performing all the operations in the fault in *back-to-back* *mode* directly after each other, and following them with a detecting read operation directly afterwards.

Root-cause—Transient faults are caused by writing weak faulty voltages into memory cells, that soon get corrected by naturally occurring leakage currents. Figure 4.5 shows the region of transient fault voltages in a DRAM cell having a net leakage current to GND. With such leakage current, transient faults are only sensitized after a $w0$ operation that sets a cell voltage between V_{bh} and V_{cs}, since such a faulty voltage is automatically corrected by leakage after some idle time.

Fault modeling—In terms of the FP notation, transient faults are represented as $<\underline{S}/F_L/R>$, where the underscore below S means that the operations in S should be performed in back-to-back mode directly after each other, and that the faulty cell value F has an added time parameter L (*life time*) to indicate that these faults are time limited. In terms of detection conditions, an underscore below operations in a transient fault means that the operations have to be performed after each other within one march element. For example, if $S = w1\underline{w0}$ then the detection condition should be $\Updownarrow(..., w1, w0, ...)$.

Examples—As an example of transient faults, consider the DRAM open shown in Figure 4.6, where R_{op} limits the ability of write operations to charge up and discharge cell voltages. For a specific range of R_{op} values, write operations set a faulty voltage within the cell that is not strong enough to qualify as a hard fault. As time passes, and due to naturally occurring leakage in DRAM cells, a weakly written faulty voltage is depleted gradually, thereby correcting the faulty information over time. Assuming a net leakage current toward GND results in transient faults associated with faulty logic 0 voltages, while a net leakage toward V_{dd} results in transient faults associated with faulty logic 1 voltages. Therefore, the shown defect causes what we may call *transient transition faults*, and using the FP notation, these can be written as $<0\underline{w1}/0_L/->$ and $<1\underline{w0}/1_L/->$. Transient faults have not yet been observed in SRAMs.

4.3.3 Space of DRAM faults

Any generic memory fault, described in Section 4.2, can represent a DRAM-specific fault by adding a DRAM-specific fault attribute to it. For example, it is possible to construct a number of DRAM-specific versions of the transition fault, such as the partial transition fault, the dirty transition fault, the soft transition fault, and so on. As discussed above, there are five DRAM-specific attributes, classified into two different classes. The first class consists of the following two voltage dependent faults: partial faults (p), and dirty faults (d). The second class consists of the following three time dependent faults: hard (h), soft (s) and transient (t) faults. It is important to note here that there are two different types of partial faults, one is the initialization related partial faults ($\mathrm{p_i}$), while the other is the activation (or sensitization) related partial faults ($\mathrm{p_a}$).

In addition to these individual attributes, it is possible to have multiple attributes at the same time associated with a given generic fault model. As a result, it is possible to establish the whole space of DRAM faults, by considering the possibility that multiple attributes apply to a fault at the same time, for a given defect.

First of all, note that any voltage dependent attribute can be combined with any time dependent attribute, without restrictions. The reason behind this is the fact that these two classes of DRAM-specific faults are based on two physically independent root causes, which results in faults that are independent as well. This means that each generic fault can be associated with a voltage dependent attribute in combination with a timing dependent attribute in the following way:

$$\text{Fault} = \left\{ \begin{array}{c} \text{Voltage dependent} \\ \text{attribute} \end{array} \right\} \left\{ \begin{array}{c} \text{Timing dependent} \\ \text{attribute} \end{array} \right\} \text{FP} \tag{4.4}$$

Furthermore, note that the different attributes of time dependent faults (h, s and t), are not compatible with each other. The reason for this is that each specific attribute of time dependent faults, influencing memory behavior, is based on the strength of applied memory operations and on the direction of the leakage current, either supporting or opposing the applied operations [see Section 4.3.2]. As a result, only one time dependent behavior can take place at a given time, and for a given defect. More precisely, the set of timing dependent faults is equal to {h, s, t}. It is worth noting here that the hard fault attribute does not modify a generic fault in any way, which means that it is identical to the absence of an attribute (symbolized by -).

In contrast, the different attributes of voltage dependent faults *are* compatible and can be combined with each other. The reason for this is that voltage dependent faults influence the behavior of the memory in different time instances, while the sensitizing operation sequence is applied. Table 4.3 shows that partial faults influence the behavior during the initialization (I) and activation (A), while dirty faults influence the behavior between I and A, and after A. More precisely, the set of voltage dependent faults is equal to {-, p, d, pd}, where - stands for no attribute, while pd stands for the combined attribute "partial dirty". It is important to note here that there are three different combinations of partial faults (p): the initialization related partial faults ($\mathrm{p_i}$), the activation related partial faults ($\mathrm{p_a}$), and the initialization and activation related partial fault ($\mathrm{p_{ia}}$).

In conclusion, Expression 4.4 can be expanded to describe all possible DRAM-specific faults as follows:

$$\text{Fault} = \left\{ \begin{array}{c} \text{-} \\ \text{p} \\ \text{d} \\ \text{pd} \end{array} \right\} \left\{ \begin{array}{c} \text{h} \\ \text{s} \\ \text{t} \end{array} \right\} \text{FP} \tag{4.5}$$

Expression 4.5 indicates that any generic fault model can be either regular (-), partial (p), dirty (d) or partial dirty (pd), while being hard (h or -), soft (s) or transient (t) at the same time. In total, this gives a space of $4 \times 3 = 12$ different attributes for DRAM-specific faults. Table 4.4 lists all these different attributes, and shows an example of how they can be attributed to the transition 0 fault (TF_0).

Table 4.4. Space of DRAM-specific faults.

#	Fault	FP	Name
1	hTF_0	$<1w0/1/->$	hard transition 0 fault
2	$p_{ia}hTF_0$	$<w1^i w0^a/1/->$	partial hard TF_0
3	$dhTF_0$	$<1w0[C]/1/->$	dirty hard TF_0
4	$p_{ia}dhTF_0$	$<w1^i w0^a[C]/1/->$	partial dirty hard TF_0
	$ghTF_0$	$<w1^i w0^a[C^d]/1/->$	general hard TF_0
5	sTF_0	$<1w0_T/1/->$	soft transition 0 fault
6	$p_{ia}sTF_0$	$<w1^i w0^a_T/1/->$	partial soft TF_0
7	$dsTF_0$	$<1w0[C]_T/1/->$	dirty soft TF_0
8	$p_{ia}dsTF_0$	$<w1^i w0^a[C]_T/1/->$	partial dirty soft TF_0
	$gsTF_0$	$<w1^i w0^a[C^d]_T/1/->$	general soft TF_0
9	tTF_0	$<1\underline{w0}/1_L/->$	transient transition 0 fault
10	$p_{ia}tTF_0$	$<\underline{w1}^i \underline{w0}^a/1_L/->$	partial transient TF_0
11	$dtTF_0$	$<1\underline{w0}[\underline{C}]/1_L/->$	dirty transient TF_0
12	$p_{ia}dtTF_0$	$<\underline{w1}^i \underline{w0}^a[\underline{C}]/1_L/->$	partial dirty transient TF_0
	$gtTF_0$	$<\underline{w1}^i \underline{w0}^a[\underline{C}^d]/1_L/->$	general transient TF_0

Fault #1 in the table is the hTF_0, which is identical to the generic TF_0. Fault #2 is the partial hard TF_0, which is denoted by the FP $<w1^i\ w0^a/1/->$. The i in the sequence $w1^i$ is caused by the initialization related partial fault (p_i), where it stands for the number of times the initializing $w1$ operation should be performed ($i \geq 0$). The a in the sequence $w0^a$ is caused by the activation related partial fault (p_a), where it stands for the number of times the activating $w0$ operation should be performed ($a \geq 1$). Fault #3 is the dirty hard TF_0, which is obtained by adding a completing sequence of operations ($[C]$) to the sensitizing operation sequence (S). Fault #4 in the table is the partial dirty hard TF_0, which is denoted by the FP $<w1^i\ w0^a\ [C]/1/->$. This fault contains i initializing $w1$ operations, it contains a activating $w0$ operations, in addition to the completing operation sequence $[C]$. The four types of hard transition faults can be represented by the general hard TF_0 ($ghTF_0$), denoted by $<w1^i\ w0^a\ [C^d]/1/->$, where $i \geq 0$, $a \geq 1$ and $d \in \{0, 1\}$. The $ghTF_0$ can be reduced to any type of hard TF_0 by properly setting its parameters i, a and d. For example, it can be reduced to the hTF_0 by using $i = 0$, $a = 1$ and $d = 0$; it can be reduced to the $p_{ia}hTF_0$ by using $d = 0$; and it can be reduced to the $dhTF_0$ by using $i = 0$, $a = 1$ and $d = 1$.

Fault #5 in the table is the soft TF_0, since it adds a delay time (T) to the sensitizing operation of the generic transition fault. In the same way, a soft version of the partial, dirty, and partial dirty TF_0 is obtained by adding a T to their respective sensitizing sequences. And finally, all these soft TF_0 can be represented by the general soft TF_0 (gsTF_0), denoted as $<w1^i\ w0^a\ [C^d]_T/1/->$, where $i \geq 0$, $a \geq 1$ and $d \in \{0, 1\}$. The gsTF_0 can be reduced to any type of soft TF_0 by properly setting its three parameters i, a and d. It can be reduced to the sTF_0 by using $i = 0$, $a = 1$ and $d = 0$, it can be reduced to the $\mathrm{p_{ia}sTF}_0$ by using $d = 0$, and it can be reduced to the dsTF_0 by using $i = 0$, $a = 1$ and $d = 1$.

Faults #9 through #12 in the table represent the four different types of transient TF_0, since they add a life time (L) to the fault effect F. In addition, an underscore is added to each operation in S, which means that these operations must be performed in *back-to-back* mode (i.e., instantaneously after each other and without delay)[2]. The general transient TF_0 (gtTF_0) is represented by the FP $<\underline{w1}^i\ \underline{w0}^a\ [\underline{C}^d]/1_L/->$, where $i \geq 0$, $a \geq 1$ and $d \in \{0, 1\}$. The gtTF_0 can be reduced to any type of transient TF_0 by properly setting its three parameters i, a and d.

4.4 Industrial test practices

Industrial testing of memories is a complex and involved process that uses as many operational aspects of the memory as possible to stress its functionality, in an attempt to induce a failure. Industrially used test aspects can mainly be classified into two categories, the used stress combinations, and the applied memory-specific operations. This section gives a general discussion of those industrial aspects of memory testing.

4.4.1 Stress combinations

A *stress combination* (*SC*) consists of a number of stresses with specific assigned values, applied together to the memory under test. A **stress** (**ST**) represents some way of facilitating the fault detection process, which includes for example cycle time t_{cyc}, temperature T, and supply voltage V_{dd}. Several papers have been published, showing the effectiveness of stresses [Goto97, Schanstra99, vdGoor99].

As mentioned in Section 4.3.2, STs are used to accelerate the detection process of soft faults. Without STs, the detection of soft faults can only take place by introducing idle time in the memory test, something that significantly increases the test cost. Figure 4.7 shows how the maximum voltage achievable by a $w1$ operation performed on a cell containing GND (V_{w1}) decreases gradually by decreasing the access time to the memory cell. When timing is significantly shortened by aggressively driving it below the boundaries set in the specifications, all cells (functional

---

[2] In terms of detection conditions, an underscore below operations in a transient fault means that the operations have to be performed after each other within one march element. For example, if $S = w1\underline{w0}$ then the detection condition should be $\Updownarrow(..., w1, w0, ...)$.

and defective) are brought closer to failure, and are actually forced to fail when $V_{w1} \leq V_{cs}$.

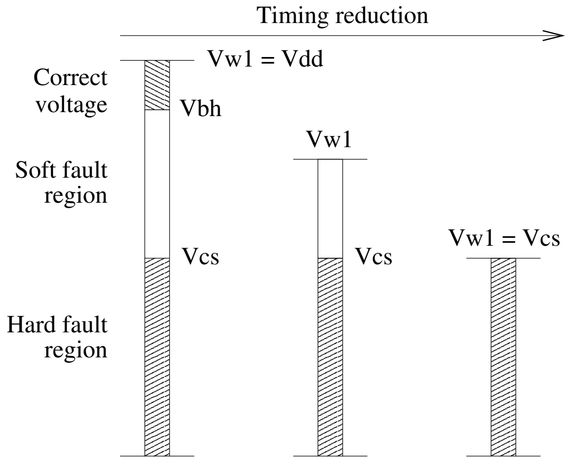


Figure 4.7. Eliminating soft faults by forcing a directly detectable fault using timing for a $w1$.

When an ST is selected to eliminate the soft fault problem of a specific defect, it should satisfy two conditions.

1. It is necessary that the ST is able to force a directly detectable fault in *any* defective cell, no matter how minor the defect is, which can only be achieved by bringing even functional cells to the verge of failure. STs able to achieve this for a given defect are called **decisive** STs for the considered defect, while STs that only influence, but do not force detectable faults are called *indecisive* STs for that defect.

2. The degradation in functionality induced by the decisive ST should take place gradually in such a way that badly defective cells are killed first and functional cells are killed last. These STs are referred to as **continuously** decisive STs.

It follows directly from this discussion that:

It is sufficient to identify only *one* continuously decisive stress for a given defect in order to eliminate soft faults from the faulty behavior of that defect.

There are practical difficulties, however, in applying STs to eliminate soft faults in DRAMs. The most serious of which is related to the amount of degradation required to eliminate the soft fault problem. Since the fabrication process of integrated circuits is not a perfect one, the characteristics of functional memories represent a statistical distribution around an ideal norm, which makes it practically impossible to identify an exact border that separates functional components from faulty ones. Therefore, great care should be taken when defining the values for decisive STs in order to prevent cutting into the range of functional components. In the industry, it is commonly accepted to eliminate some functional (but weak) components in order ensure the high quality of the memories delivered to the customer [Vollrath00].

As an example, a well-known industrial stress condition is the so-called V_{dd} bump test, which uses one of the supply voltages as the stress parameter [Al-Ars01c]. The stresses commonly used in industry can be divided into the following classes:

1. **Algorithmic stresses**—They specify more precisely the way a test algorithm should be applied. This can be in terms of:
   (a) The specific *addressing direction* (*AD*) used to access memory cells in the test [vdGoor04a]. There is, for example, fast-x addressing (denoted by ${}_{x}\Updownarrow$) and fast-y addressing (denoted by ${}_{y}\Updownarrow$).
   (b) The specific *counting method* (*CM*) used in the test to proceed from one cell to the next [vdGoor04a]. There is, for example, the binary CM, the Gray code CM, etc.
   (c) The specific data to be read/written. This usually is denoted as the data background (BG). There is, for example, the solid BG (where all cells use the same value) and the checkerboard BG (where the vertical and horizontal neighbors of the cell contain the complement value stored in the cell).
2. **Environmental stresses**—These are stresses which relate to the environment the chip has to be tested in. Temperature, voltage and timing are typical examples of this class of stresses.

The exact way these stresses are used depends on the specific design and construction of the memory under test.

4.4.2 Device-specific fault primitives

This section discusses the shortcomings of the current FP notation and suggests ways to improve it. The discussion here is related to the way external operations performed on the memory result in sensitizing faults. The improvements to the FPs concentrate on a given type of DRAMs as the memory of interest, but similar strategies may be used to take other types of memory into consideration.

Shortcomings of current FPs

The FP notation presented in Section 4.2 is a generic notation that is compatible with almost all RAM devices today. Such a generic notation does not address the faulty behavior needs of specific types of memory devices (DRAMs, for example). The inability of FPs to describe DRAM-specific faulty behavior originates from the limited types of operations S can describe.

The definition of S is based on a reduced memory model, as shown in Figure 4.8(a), that is the same for all RAM devices [vdGoor98]. This model assumes that a memory has only three input/output terminals: an address input bus (Address), a data input/output bus (Data in/out), and a command input ($\mathrm{R}/\overline{\mathrm{W}}$) to perform either a read or a write. For example, this model can represent a TF_1 transition fault as in $<0w1/0/->_c$, where the address bus contains the address of the cell c, the $\mathrm{R}/\overline{\mathrm{W}}$ input encodes a write operation and the data in/out bus contains the written data 1. The advantage of this model is that it is simple, thereby keeping the needed analysis of the faulty behavior simple. The model is also generic, which makes analysis results based on this model applicable to many memory devices. The disadvantage of the model, however, is that it neglects a number of parameters that could effect the behavior of the memory (e.g., temperature and voltage). An attempt has been made to improve on this model by including voltages and temperature (T), found to be important in testing [Offerman97]. Still, the model does not include timing or memory-specific operations.

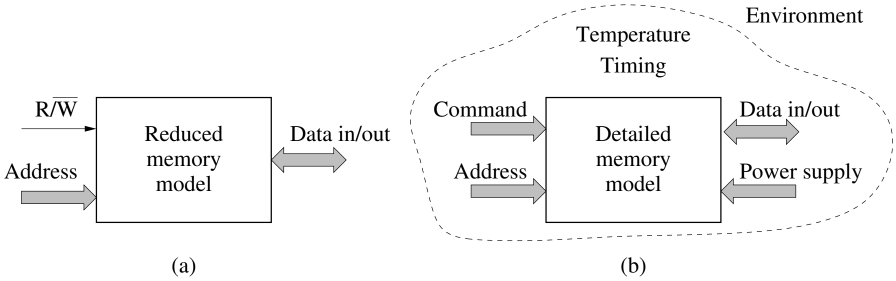


Figure 4.8. Memory models used for defining S: (a) a reduced memory model, (b) a detailed memory model.

Figure 4.8(b) shows such a memory model with 3 input and/or output terminals, and which takes supply voltages, temperature and timing into consideration. The data in/out bus and the address bus are the same in the reduced as well as the detailed models. The command bus in the new model replaces the $\mathrm{R}/\overline{\mathrm{W}}$ input in the old model.

In order to carry out this extension on the new model shown in Figure 4.8(b), we need to select a specific memory product in order to give the exact definitions of the command bus and specify the type of supply voltages and timing parameters to be modified. Table 4.5 gives these definitions for a current Infineon DRAM product [Falter00].

Table 4.5 identifies two voltages for the power supply and two clock related parameters to control timing. In addition, the table lists the five DRAM-specific primitive commands.

Table 4.5. Definitions of command, voltage, timing and temperature parameters for a DRAM.

Terminal	Definition	Description
Command	Act	Activate: access a row of cells and sense their data content.
	Wr	Write: write data into a memory cell.
	Rd	Read: forward data from the sense amplifier to output.
	Pre	Precharge: restore data to cells and precharge bit lines.
	Nop	No operation
Supply voltage	$\mathrm{V}_{\mathrm{arr1}}$	First array voltage: power supply to the cell array.
	$\mathrm{V}_{\mathrm{arr2}}$	Second array voltage: power supply to the cell array.
Timing	Frq	Frequency of the clock signal
	τ	Duty cycle of the clock signal
Temperature	T	Ambient temperature

Extending the FP notation

Extending the current FP notation can be done by enabling S to describe any possible operation sequence performed on the new model shown in Figure 4.8(b) using the specific terminal definitions of the current Infineon DRAM product listed in Table 4.5. The needed extensions involve: 1) describing the five DRAM-specific commands, 2) describing the algorithmic stresses, and 3) describing the environmental stresses of the memory. Each one of these extensions is described below.

Describing memory-specific commands The first step to account for the five DRAM commands of Table 4.5 is to modify the set of possible operations in FPs to include all of them. This way the set of possible performed operations should be $\{\mathrm{Act}_{c_w}, \mathrm{Rd}d_{c_b}, \mathrm{Wr}d_{c_b}, \mathrm{Pre}, \mathrm{Nop}\}$. Unlike the traditional read (r) and write (w) operations, these five commands are not independent from each other, which means that some Ss should not be allowed. Therefore, the condition described in Expression 2.11 should be satisfied to limit the space of S to those practically possible.

$$\mathrm{Act}\ \mathrm{C}d_1\ ...\ \mathrm{C}d_i\ ...\ \mathrm{C}d_n\ \mathrm{Pre}\ \mathrm{Nop}\ ...\ \mathrm{Nop},\ \mathrm{C}d_i \in \{\mathrm{Wr0}, \mathrm{Wr1}, \mathrm{Rd0}, \mathrm{Rd1}, \mathrm{Nop}\}$$

Describing algorithmic stresses Algorithmic stresses refer to the specific addressing used to detect a given fault, or to the needed data background. The needed data background is already described by the current FP notation. In order to take addressing into consideration, a number of attributes should be added to the FP to specify the topological relation between the cells accessed within the FP. The most important attributes are BL (cells are along the same bit line), WL (cells are along the same word line), and DG (cells are along the diagonal). For example, the two cells in the fault $<0_a 1_v/0/->_{BL}$ are indicated by BL to be along the same bit line. In addition to the topological relationships, it is important to identify adjacency relationships of the cell in an FP. A well-known example of these relations is the four closest neighbors of a given cell (the cells to the north, south, east and west), symbolized by the $\diamond$ in the fault $<0_a 1_v/0/->_{\diamond}$, which means that the aggressor is one of the four neighbors of the victim.

Describing environmental stresses Unlike performed operations on the command bus, environmental stresses are not discrete but continuous quantities. This means that it is not possible to identify *all* allowed stresses and individually integrate them into S. Rather, a parameterization of the stresses can be considered by introducing *stress defining variables* for each stress. Therefore, five stress variables should be introduced into S, one for each stress listed in Table 4.5. To distinguish stresses from operations, stresses are included in square brackets. For example, S = [$\mathrm{V}_{\mathrm{arr1}}$ = 3.5] Act_{c_w} $\mathrm{Wr}d_{c_b}$ Pre [$\mathrm{V}_{\mathrm{arr1}}$ = 3.0] Act_{c_w} $\mathrm{Rd}d_{c_b}$ Pre means that cell c (with row address c_w and column address c_b) is written with data d at an array voltage of 3.5 V and then read at a voltage of 3.0 V.

4.4.3 Examples of new notation

In this section, examples are given to justify the need for extending the FP notation. The examples concern the memory-specific commands and the environmental stresses included in the new FP notation.

Memory-specific commands

Consider a bridge defect that connects two DRAM memory cells together. This defect causes a write operation to the aggressor to affect the stored voltage in the victim. If the bridge resistance is high enough, it would take a number of write operations to the aggressor to change the stored voltage in the victim.

Assume that the victim stores a 0, and that it takes 3 $w1$ operations to the aggressor to flip the state of the victim. The traditional FP notation describes this faulty behavior as $<1_a w1_a w1_a w1_a 0_v/1/->$. This description cannot be uniquely translated into the DRAM primitive commands since it is possible to perform the w and r operations in single cycle mode and in fast page mode. In the new notation, the faulty behavior can be uniquely described using fast page mode as follows $<0_a$ Act_{a_w} $\mathrm{Wr1}_{a_b}$ $\mathrm{Wr1}_{a_b}$ $\mathrm{Wr1}_{a_b}$ Pre $0_v/1/->$.

Stress combinations

Again, consider the example where a bridge defect connects two DRAM cells together. This bridge results in the following FP <0_a Act$_{a_w}$ Wr1$_{a_b}$ Wr1$_{a_b}$ Wr1$_{a_b}$ Pre 0_v/1/−>. To help sensitize this FP, stress combinations can be used to optimize the S so that the FP would be detected more easily.

For this faulty behavior in particular, increasing the supply voltage from its nominal value (V_n) to a higher value (V_h) while writing would shorten the time needed to deplete the victim capacitor. At the same time, reducing the operation temperature to a lower T_l decreases the resistance of the bridge in favor of the failure mechanism. Identifying the most optimal values for V_h and T_l depends on the nature of the bridge being considered. Taking these stress combinations into consideration, the new FP description of the failure mechanism would be <$[V_{arr1} = V_h, T = T_l]$ 0_a Act$_{a_w}$ Wr1$_{a_b}$ Wr1$_{a_b}$ Wr1$_{a_b}$ Pre 0_v/1/−>.

Summary

This chapter discussed the modeling of faulty behavior of memory devices, both in general and specifically for DRAM devices. The DRAM-specific fault modeling approach presented here reduces the complexity of the general analysis of memory faults and restricts the fault space to those faults relevant to DRAMs. The main issues presented in this chapter are the following:

- Formally defining the concepts of fault primitives and functional fault models, the cornerstones of current-day memory fault analysis. These concepts restrict the attention of fault analysis activities to those aspects of the faulty behavior relevant for effective and efficient test generation.
- Enumeration of the two most important classes of memory faults: single-cell static faults and two-cell static faults. These two classes have been used repeatedly to successfully generate memory tests for a large number of memory devices.
- Discussion of the exponential complexity needed to analyze the full space of dynamic memory faults, as each increment of the number of operations to be analyzed results in a large relative increase in the number of faults a memory should be inspected for.
- Introduction of the different types of DRAM-specific faults: those related to improperly set voltages, and those related to time dependence due to leakage.
- Identification of the space of DRAM-specific faults, using five individual fault attributes, which result in 12 different attribute combinations. These 12 attribute combinations modify the behavior of a given generic FP in such a way that describes all DRAM-specific faults.
- Generalization of the FP notation to take DRAM-specific commands into consideration, along with the stress combinations necessary for DRAM testing.

Contents of this chapter

5.1 Conventional analysis
5.2 Approximate simulation
5.3 Two dimensional analysis
5.4 DRAM-specific simulation

5

Fault analysis approximation methods

Although electrical simulation has become a vital tool in the design process of memory devices, memory testing has not yet been able to employ electrical simulation as an integral part of the test generation and optimization process. This is due to the exponential complexity of the simulation based fault analysis, a complexity that made such an analysis impractical. This chapter describes new methods to reduce the complexity of the fault analysis from exponential to constant with respect to the number of analyzed operations, thereby making it possible: 1) to use electrical simulation to generate test patterns, and 2) to perform simulation-based stress optimization of tests.

Section 5.1 gives shows the high time complexity of simulation, along with an example of a conventional simulation-based fault analysis performed by simulating all needed operation sequences precisely and without approximation. This section serves as an example to compare with the approximate analysis method presented in Section 5.2. Section 5.3 presents a generalization of the approximate analysis method, in order to deal with multiple floating nodes in a defective memory. Finally, Section 5.4 tailors the approximate simulation method to the needs of the DRAM, and shows how to use simulation to optimize stresses.

5.1 Conventional analysis

In this section, the direct simulation approach for fault analysis, called the *precise simulation*, is presented. The section starts with a discussion of the time complexity of simulation, followed by an example and the properties of the precise simulation.

5.1.1 Complexity of simulation

The increasing complexity of the faulty behavior of memory devices, associated with the ever increasing costs of memory testing, makes it important to look for new innovative ways to tackle fault analysis and test issues for memories [Iyer99]. The complexity of the fault analysis is particularly demanding for DRAMs, as a result of their vulnerability to dynamic faults (which involve multiple memory operations), since each additional analyzed operation requires an exponential increase in fault analysis time [see Section 4.2.3]. Previous work on dynamic faults has either been limited to the impact of specific types of memory operations (sequences of reads, for example), or only concerned with analyzing a limited number of dynamic sequences to limit simulation time [Al-Ars03a, Al-Ars00].

As discussed in Section 4.2, the faulty behavior of memory devices can be represented using the FP notation $<S/F/R>$, where S stands for the sensitizing operation sequence needed to sensitize the faulty behavior, F stands for the value stored in the faulty cell, while R is the value on the output. The most well-known class of FPs is the class of single-cell static FPs, where at most one operation and only one cell is associated with sensitizing the fault [see Table 4.1]. Transition faults (TFs) are an example of single-cell static FPs, where a transition write operation ($0w1$ or $1w0$) fails to flip the cell to the opposite value. In total, there are 12 different single-cell static FPs, as described by the FP notation of Table 4.1.

As operations are added to S, in order to investigate the dynamic faulty behavior of the memory, the possible number of different Ss, and the associated number of dynamic FPs, increases rapidly. For a single-cell FP, S typically starts with an initialization of either 0 or 1, followed by one of three possible memory operations $w0$, $w1$ or r for each increment in $\#O$. As a result, the possible number of different Ss can be calculated by Equation 4.2:

$$\#S = 2 \cdot 3^{\#O}$$

When no operation is performed ($\#O = 0$) the number of possible FPs is 2, and when one or more operations are performed, S is able to sensitize 10 different FPs for each increment in $\#O$, as summarized in Equation 4.3:

$$\#\text{single-cell FPs} = \begin{cases} 2 & : \#O = 0 \\ 10 \cdot 3^{(\#O-1)} & : \#O \geq 1 \end{cases}$$

Figure 5.1 plots the number of possible Ss and the number of possible FPs against $\#O$, where the exponential nature of these relations can be clearly seen. It is clearly not practically possible to investigate the dynamic faulty behavior of the memory, by directly applying all possible Ss, since such a task would take an *infinite* amount of time to perform. Using the analysis approach presented in this chapter, the total *infinite* space of dynamic faulty behavior can be approximated within a short amount of analysis time.

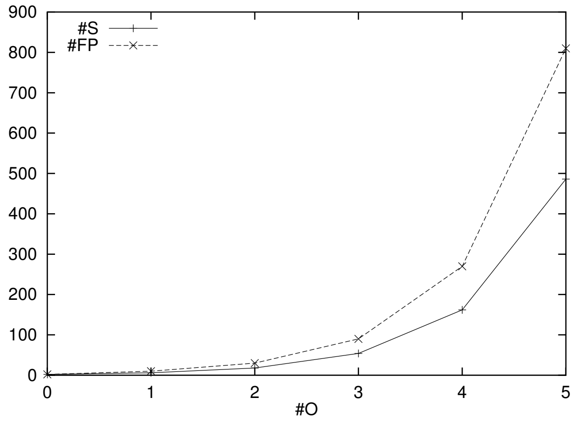


Figure 5.1. Plot of $\#S$ and #FP as a function of $\#O$.

5.1.2 Example of conventional analysis

Consider the open defect (R_{op}) within a DRAM cell, as shown in Figure 5.2, where Spice simulations are to be used to analyze the faulty behavior resulting from this open. The analysis takes a range of possible open resistances into consideration ($1\ \Omega \leq R_{op} \leq 10\ \text{M}\Omega$, for example). The injected open in the cell model creates a **floating node** of the cell capacitor, the voltage of which (V_c) may vary between $V_{dd} = 2.4$ V and GND. A floating node is a memory node that is not properly controlled by a memory operation as a result of the defect. The improper control the memory operation has on the floating node results in setting an improper voltage on the node by the end of the operation. Determining which memory node an injected open causes to be floating depends on the type of the open and the design of the memory. The floating node for opens within memory cells is taken to be the node connected to the cell capacitor, since the other node is controlled by the pass transistor.

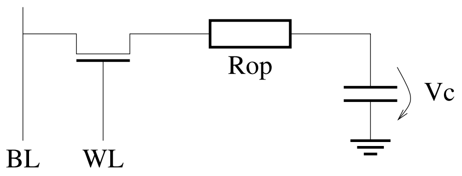


Figure 5.2. DRAM memory cell with an open defect.

Next, the fault analysis is performed for some points in the (V_c, R_{op}) plane, which is called the *analysis space*. Therefore, a number of values for V_c and R_{op} are selected to perform the fault analysis. This usually corresponds to applying a grid on the analysis space, giving rise to a number of intersection points for which the analysis is performed.

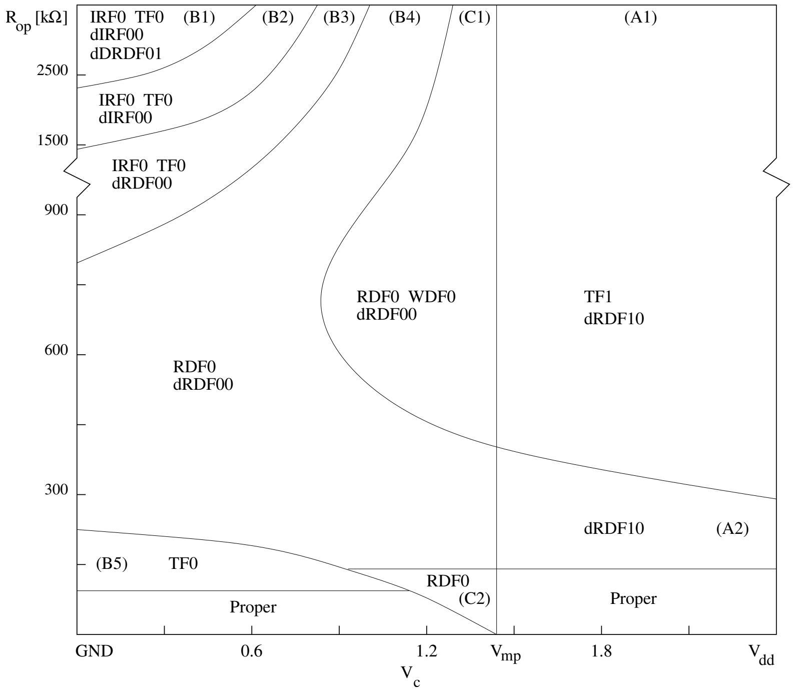


Figure 5.3. Fault analysis results of the open in Figure 5.2 in the analysis space.

For each analysis point in the analysis space, the faulty behavior of the memory is analyzed by simulating a number of memory operations. The more operations are simulated, the more accurate our understanding of the faulty behavior becomes. However, the number of different possible Ss grows exponentially with respect to $\#O$ according to Equation 4.2: $\#S = 2 \cdot 3^{\#O}$. Therefore, the more operations are performed, the more time it takes to carry out the analysis, which makes it important to limit the number of used operations. For example, if sequences of only two operations ($\#O = 2$) are considered to be performed on a single memory cell, then there are $2 \cdot 3^{\#O} = 18$ different sequences of $w0$, $w1$ and r operations possible. Each of these sequences has to be performed for each analysis point in the analysis space.

Figure 5.3 shows the fault analysis results performed for the open shown in Figure 5.2. The results were generated using an analysis space of 10 voltage points (x-axis) and 15 resistance points (y-axis), and using Ss of at most 2 operations [Al-Ars03b]. The V_{mp} vertical line in the middle of Figure 5.3 represents the middle-point voltage in the cell. If a cell has a voltage below V_{mp}, we consider the cell to contain a stored 0, while if a cell has a voltage above V_{mp}, we consider the cell to contain a stored 1. The analysis results are organized as fault regions in the analysis space that change gradually (i.e., continuously) with respect to V_c and R_{op}. For example, Region B5 in Figure 5.3 contains the static fault TF_0 ($<0w1/0/->$), while Region A1 contains the static fault TF_1 ($<1w0/1/->$) and the dynamic fault dRDF_{10} ($<1w0r0/1/1>$).

5.1.3 Fault analysis time

This section estimates the time needed to perform the fault analysis, using the precise simulation approach. The time needed can be described by the following relation:

$$T_{psim} = \#P \cdot \#S \cdot T_s \tag{5.1}$$

where $\#P$ is the number of points in the analysis space, $\#S$ is the number of Ss to be performed for each point (equals $2 \cdot 3^{\#O}$), and T_s is the time needed to simulate each S. Furthermore, $\#P$ can be further decomposed into $\#P = \#X \cdot \#Y$, where $\#X$ is the number of points taken along the x-axis of the analysis space, and $\#Y$ is the number of points taken along the y-axis of the analysis space. T_s can also be further decomposed into $T_s = T_o \cdot \#O$, where T_o is the simulation time needed for a single memory operation. In summary, the simulation time needed for the precise analysis can be written as:

$$T_{psim} = \#X \cdot \#Y \cdot 2 \cdot 3^{\#O} \cdot T_o \cdot \#O \tag{5.2}$$

We use the analysis performed in Figure 5.3 as an example, where V_c is taken to be the x-axis and R_{op} is taken to be the y-axis.

1. $\#X = 10$ points (10 V_c values on a linear scale, GND $\leq V_c \leq 2.4$ V)
2. $\#Y = 15$ points (2 R_{op} values per decade on a logarithmic scale, $1\Omega \leq R_{op} \leq 10$ MΩ)
3. $\#S = 18$ (2-operation Ss)
4. $T_o = 10$ s of simulation time
5. $\#O = 2$

This adds up to $T_{psim} = \#X \cdot \#Y \cdot \#S \cdot T_o \cdot \#O = 10 \cdot 15 \cdot 18 \cdot 10 \cdot 2 = 54000$ s = 15 hours. Note that despite the restriction of the analyzed $\#O$ to 2, the simulation still takes a long time to perform.

5.2 Approximate simulation

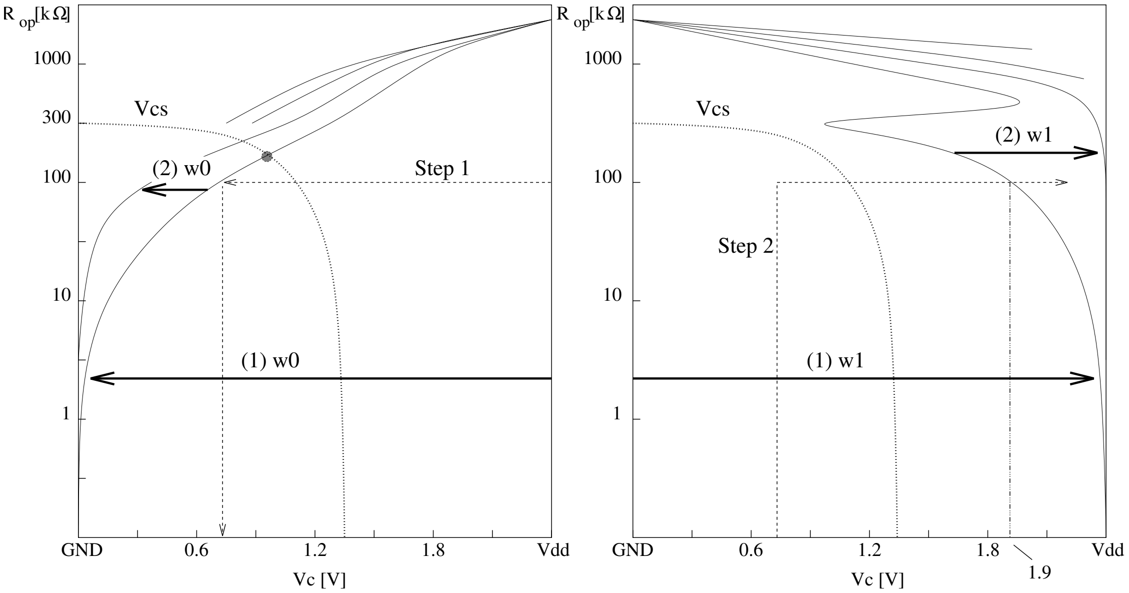
This section presents the new fault analysis approach, called the *approximate simulation* [Al-Ars02b]. Here, only the relatively simple **one dimensional** (**1D**) analysis of the approximate simulation is discussed, leaving the more complex two dimensional analysis to the next section. The section starts with an example, then the properties of the approximate simulation approach are discussed.

5.2.1 Example of 1D analysis

The new approximate simulation approach differs from precise simulation in that it enables the investigation of the analysis space for operation sequences with *any* $\#O$, with a limited amount of analysis time. It achieves this by compromising the accuracy of the results.

Consider the defective DRAM cell shown in Figure 5.2, where an open (R_{op}) creates one floating node of the voltage across the cell capacitor (V_c). The analysis takes a range of possible open resistances ($1\ \Omega \leq R_{op} \leq 10\ \text{M}\Omega$) and possible floating cell voltages ($\text{GND} \leq V_c \leq V_{dd}$) into consideration. Since only *one* floating node is being analyzed here, this approximate simulation is referred to as the *one dimensional* (*1D*) analysis. In the next section, a method is described that takes *two* floating nodes into consideration, which is therefore referred to as the *two dimensional* analysis.

In the 1D analysis, three different (V_c, R_{op}) *result planes* are generated, one for each memory operation ($w0$, $w1$, and r). Each result plane describes the impact of successive $w0$, successive $w1$, or successive r operations on V_c for a given value of R_{op}. Figure 5.4 shows the result planes for the $w0$ and $w1$ memory operations performed for the open shown in Figure 5.2.



(a) Plane of w0

(b) Plane of w1

Figure 5.4. Result planes of the approximate simulation for the operations (a) $w0$, and (b) $w1$.

Plane of $w0$: This result plane is shown in Figure 5.4(a). To generate this figure, the floating cell voltage V_c is initialized once to V_{dd}, and once to GND, and then the operation sequence $w0w0...w0$ is applied to the cell. When the initial voltage is

V_{dd}, the net result of the write sequence is the gradual decrease (depending on the value of R_{op}) of V_c towards GND, as reflected in the figure. When the initial voltage is GND, the write sequence has *no* effect on V_c and, therefore, has no impact on the figure. The voltage level after each $w0$ operation is recorded on the result plane, resulting in a number of curves. Each curve is indicated by an arrow pointing in the direction of the voltage change. The arrows are numbered as $(n)w0$, where n is the number of $w0$ operations needed to get to the indicated curve. We stop performing the $w0$ sequence when the voltage change ΔV_c as a result of $w0$ operations becomes $\Delta V_c \leq 0.24$ V, a value that is arbitrarily selected at first, but can afterwards be reduced if it turns out that more $w0$ operations are needed to describe the faulty behavior. This selection of ΔV_c results in identifying up to four different $w0$ curves in the plane. The V_{cs} curve (shown in the figure as a dotted line) is the cell sense threshold voltage, which is the cell voltage above which the sense amplifier reads a 1, and below which the sense amplifier reads a 0. The V_{cs} curve is copied from the plane of the r operation, which is explained in detail below [see "Plane of r" below].

Plane of $w1$: This result plane is shown in Figure 5.4(b). To generate this figure, V_c is initialized once to V_{dd} and once to GND and then the operation sequence $w1w1...w1$ is applied to the cell. With an initial voltage of GND, the result of the sequence is a gradual increase of V_c towards V_{dd}. With an initial voltage of V_{dd}, the write sequence has *no* effect on V_c and, therefore, has no impact on the figure. The voltage level after each $w1$ operation is recorded on the result plane, which gives a number of curves in the plane. The curves are indicated in the same way as the curves in the plane of $w0$. We stop the $w1$ sequence when ΔV_c becomes less than some arbitrarily selected small value (0.24 V in this example). It is interesting to note the bump in the curve $(1)w1$ of Figure 5.4(b) at about $R_{op} = 1000$ kΩ, starting from $R_{op} = 300$ kΩ till about 3000 kΩ. Remember that any memory operation starts with the activation command (Act), where the sense amplifier senses the voltage stored in the cell and amplifies it. Above $R_{op} = 300$ kΩ, the sense amplifier fails in sensing a stored 0, and senses a 1 instead, which helps the $w1$ operation in charging up the cell to a higher V_c. This increases the $(1)w1$ voltage gradually, starting from $R_{op} = 300$ till 1000 kΩ. V_{cs} is shown in the figure as a dotted line.

Plane of r: This result plane is shown in Figure 5.5. To generate this figure, first V_{cs} is established and indicated on the result plane (shown as a bold curve in the figure). This is done by performing a read operation for a number of V_c values, and recursively identifying the V_c border above which the sense amplifier detects a 1 and below which the sense amplifier detects a 0. As R_{op} increases, V_{cs} turns closer to GND which means that it gets easier to detect a 1 and more difficult to detect a

0[1]. Then the sequence $rrr...r$ is applied twice: first for V_c that is marginally lower than V_{cs} (0.12 V lower in this example), and a second time for V_c that is marginally higher than V_{cs} (0.12 V higher). The voltage level after each r operation in both read sequences is recorded on the result plane, which generally results in two sets of curves on the plane. Each set of curves is indicated in the same way as for the curves in the plane of $w0$. Note that with $V_c < V_{cs}$ (the part below the bold curve in the figure) only one r operation is enough to set V_c to GND, and therefore there are no curves in this part of the plane.

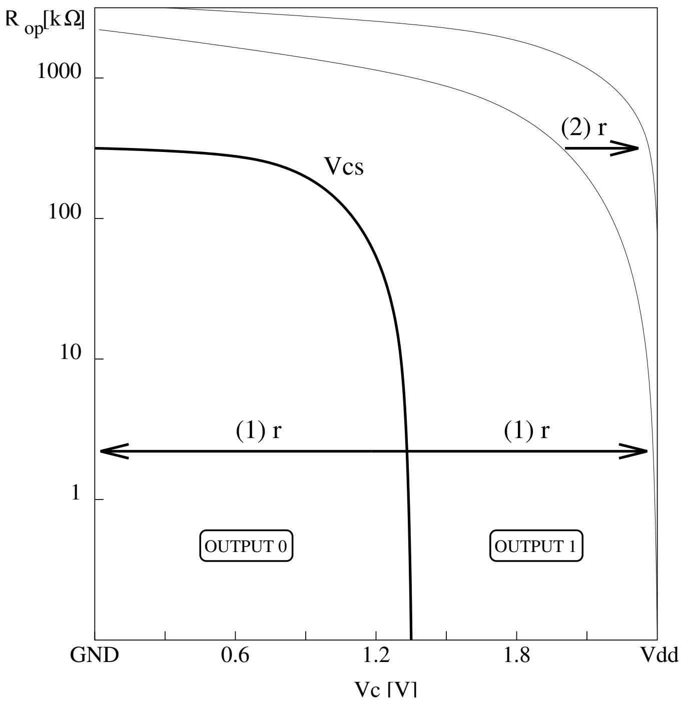


Figure 5.5. Result plane of the approximate simulation for the r operation.

It is possible to use the result planes of Figure 5.4 to analyze a number of aspects of the faulty behavior. We mention three aspects here and show how to derive them from the figure.

1. Approximate the behavior resulting from any operation sequence performed on the defective memory.
2. Indicate the **critical resistance** (R_{cr}), which is the R_{op} value where the cell *starts* to cause faults on the output for at least one sequence of operations.

---

[1] This is caused by the fact that the precharge cycle sets the bit line voltage to V_{dd}. Therefore, as R_{op} increases, a 0 stored in the cell fails to pull the bit line voltage down during a read operation, and the sense amplifier detects a 1 instead of a 0.

3. Generate a test that detects the faulty behavior of the defect for any resistance value and any initial floating voltage.

(1) The result planes can be used to approximate the faulty behavior of *any sequence* of memory operations. For example, the results shown in Figure 5.4 can be used to find out the behavior of, say, $1w0w1r1$ for $R_{op} = 100$ kΩ. The behavior is evaluated as follows:

1. Starting with an initial $V_c = V_{dd} = 2.4$ V, check the value of V_c after performing one $w0$ operation, $(V_c = V_{dd}) \xrightarrow{w0} (V_c = 0.7 \text{ V})$ [see dashed line of Step 1 in Figure 5.4(a)].

2. Using the new $V_c = 0.7$ V, check the value of V_c after performing one $w1$ operation, $(V_c = 0.7 \text{ V}) \xrightarrow{w1} (V_c > 1.9 \text{ V})$ [see dashed line of Step 2 in Figure 5.4(b)]. The figure shows that starting with $V_c =$ GND, one $w1$ operation pulls V_c up to 1.9 V. This means that starting with 0.7 V > GND in the cell, a $w1$ operation should pull V_c up to at least 1.9 V.

3. Using $V_c > 1.9$ V, check the behavior of the read operation, $(V_c > 1.9 \text{ V}) \xrightarrow{r} (V_c > 2.2 \text{ V})$, output = 1.

This means that the memory behaves properly and no fault is detected using the sequence $1w0w1r1$ for $R_{op} = 100$ kΩ.

(2) The approximate simulation can also be used to state the *critical resistance*, which is the R_{op} value below which the memory behaves properly for at least one operation sequence. For the fault analysis shown in Figure 5.4, the memory would behave properly for any operation sequence as long as $R_{op} < 200$ kΩ. To understand why, note that a fault would only be detected when a $w1$ operation fails to charge V_c up above V_{cs}, or a $w0$ fails to discharge V_c to below V_{cs}, where V_{cs} is indicated by the dotted curve in Figures 5.4(a) and (b). In both cases, performing a r after the w detects the faulty behavior. This situation takes place on the result planes at the intersection between the first write operation curves, $(1)w0$ or $(1)w1$, and the V_{cs} curve. The $(1)w0$ curve intersects V_{cs} curve at $R_{op} = 200$ kΩ, as indicated by the dot in Figure 5.4(a). Note that the curve $(1)w1$ in Figure 5.4(b) does not intersect the V_{cs} curve, which means that $w1$ operations can never result in detecting a fault.

(3) The approximate simulation can also be used to generate a test that detects the faulty behavior caused by *any* defect resistance R_{op} for *any* initial floating voltage V_c, in case a fault can be detected. In the case of Figure 5.4, faults can be detected with $R_{op} \geq 200$ kΩ. Inspecting the figure shows that with $R_{op} \geq 200$ kΩ, and with any voltage V_c, the sequence $w1w1w0$ will sensitize a fault, and a subsequent $r0$ will detect it. This, in turn, means that the faulty behavior can be represented by the FP $=<w1w1w0/1/->$. For $R_{op} = 200$ kΩ, this can be validated by noting that performing two $w1$ operations charges V_c up from any voltage (GND or higher) to V_{dd}. With $V_c = V_{dd}$, the sequence $w0r0$ detects a fault as discussed in point (2) above. As R_{op} increases, the faulty behavior becomes more prominent and easier to detect since V_{cs} decreases rapidly towards GND. With $R_{op} \geq 300$, any read operation with any initial V_c results in 1 on the output, which means that any sensitizing $w0$ fails to bring about a successful $r0$. Therefore, this faulty behavior can be modeled using the FP $<w1^2\ w0/1/->_c$, which is a partial fault in the initialization (I). Since the open defect does not force leakage current to any specific direction, all time dependent faults are possible (hard, soft and transient). This means that the following two additional FPs are also possible: $<w1^2\ w0_T/1/->_c$ and $<\underline{w1^2}\ \underline{w0}/1_L/->_c$. To detect the hard and transient faults, the detection condition $\Updownarrow$ (..., $w1$, $w1$, $w0$, $r0$, ...) is sufficient. The soft fault requires the detection condition $\Updownarrow$ (..., $w1$, $w1$, $w0$, Del, $r0$, ...).

5.2.2 Fault analysis time

The approximate simulation is much less time consuming than the precise simulation. The time needed can be described using Equation 5.1 as follows:

$$T_{asim} = \#P \cdot \#S \cdot T_s$$

where $\#P$ is the number of points in the analysis space, $\#S$ is the number of Ss to be performed for each point, and T_s is the time needed to simulate each S. In the approximate simulation, $\#S = 6$ since we use only 3 operation sequences, a sequence of $w0$, $w1$ and r, each with 2 different initial voltages. Furthermore, $\#P = \#Y$, where $\#Y$ is the number of points taken along the y-axis of the analysis space ($\#X$ is dropped since we do not take any point on the x-axis). T_s can be further decomposed as $T_s = T_o \cdot \#O$, where T_o is the simulation time needed for a single memory operation.

The $\#O$ for the approximate simulation depends on how fast a given S charges the memory cell. However, in order to keep a simulation accuracy along the x-axis that is approximately as good as that of the precise simulation, we need at most $\#X$ points along the x-axis, which means that we need at most $\#O = \#X$. Operations, however, usually charge V_c fast for most of the R_{op} range, thereby reducing the average $\#O$ needed. In summary, the worst case simulation time needed for the approximate analysis can be written as:

$$T_{asim} = \#X \cdot \#Y \cdot T_o \tag{5.3}$$

We use the analysis performed in Figure 5.4 as an example, where V_c is taken to be the x-axis while R_{op} is taken to be the y-axis.

1. $\#Y = 15$ points (2 R_{op} values per decade on a logarithmic scale)

2. $\#S = 6$ ($w0$, $w1$ and r Ss, each with 2 initial voltages)

3. $T_o = 10$ s of simulation time

4. $\#O \approx 2$ (this is the average per S over the R_{op} range)

This adds up to $T_{asim} = \#Y \cdot \#S \cdot T_o \cdot \#O = 15 \cdot 6 \cdot 10 \cdot 2 = 1800$ s = 0.5 hours, which is about 30 times faster than precise simulation. The difference can be much higher if the number of operations increases. The general theoretical worst case speedup of the approximate simulation approach can be derived from Equations 5.2 and 5.3 as:

$$\boxed{\text{Speedup} = \frac{T_{psim}}{T_{asim}} = 2 \cdot \#O \cdot 3^{\#O} \qquad (5.4)}$$

which is an exponential speedup with respect to $\#O$.

5.3 Two dimensional analysis

Some defects in the memory result not only in a floating cell node, but also in a floating node elsewhere, which the memory is not able to control properly during write and read operations. As a result, the approximation method presented in Section 5.2 for the case of a single floating memory node is not sufficient to approximate the faulty behavior. This section introduces a generalization of the approximation method that is able to deal with two floating nodes in a defective memory. This method is referred to as the **two dimensional** (**2D**) analysis. The 2D analysis presented here can be extended further to deal with multiple floating nodes.

Consider the bit line open shown in Figure 5.6, a resistive open (R_{op}) at this location on BT reduces the ability of the memory to control the voltage on the right hand side of the bit line and across all cells on that part of BT. Figure 5.6 also shows the cell capacitance (C_c), and the bit line capacitance (C_b) which represents the capacitance of the BT part to the right of the open. The figure shows that R_{op} causes *two* floating nodes: **(1)** the cell voltage V_c across the cell capacitor C_c, and **(2)** the bit line voltage V_b across the parasitic bit line capacitance C_b.

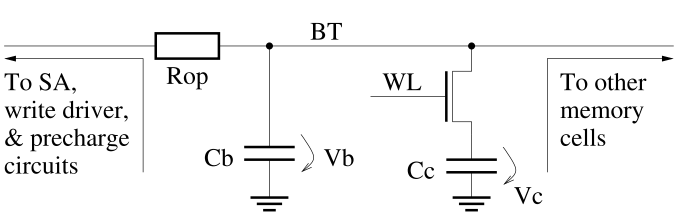


Figure 5.6. Open defect on BT that requires a 2D analysis.

The fact that R_{op} causes two floating nodes in the memory implies that the analysis of the faulty behavior of this open should take all possible BT voltages (GND $\leq V_b \leq V_{dd}$) and cell voltages (GND $\leq V_c \leq V_{dd}$) into consideration. In addition, the analysis should include a range of possible open resistances (10 kΩ $\leq R_{op} \leq$ 10 MΩ) into consideration. The results of the analysis can be represented as three *result spaces*, one for each memory operation ($w0$, $w1$ and r), where the x-axis of the result space stands for V_c, the y-axis stands for V_b, while the z-axis stands for R_{op}. Since it is difficult to visualize these types of figures, it is more convenient to represent the results as a number of (V_c, V_b) planes, one for each R_{op} value, which in turn are organized into three sets, one for each memory operation ($w0$, $w1$ and r). As it is the case for the 1D analysis, the only important part of the result space of r is the V_{cs} curve, which is included and discussed as part of the result spaces of $w0$ and $w1$. Therefore, only the result spaces of $w0$ and $w1$ are discussed next.

Planes for $w0$

Figures 5.7 and 5.8 show three different (V_c, V_b) planes for three R_{op} values, resulting from performing a sequence of $w0$ operations on the defective memory model shown in Figure 5.6. For example, Figure 5.7, with R_{op} = 200 kΩ, has six main curves numbered in the figure as 1 through 6. Curve 1 is a vector from point $(V_c, V_b) = (0, 0)$ to $(0.13, 0.08)$, describing the impact of performing $(1)w0$ on a memory initialized to $(0, 0)$. Curves 2, 3 and 4 describe the impact of performing $(1)w0$ operation on a memory initialized to $(0.0, 2.0)$, $(2.0, 2.0)$ and $(2.0, 0.0)$, respectively. Curve 5 is a bundle of vectors that describe the impact of the sequence $w0w0...w0$ on V_c and V_b. The net effect of performing the $(n)w0$ sequence is the gradual convergence of V_c and V_b voltages to a specific point P = $(0.16, 0.11)$ that is independent of the initialization.

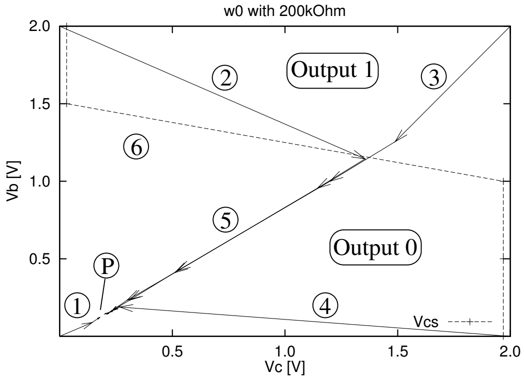
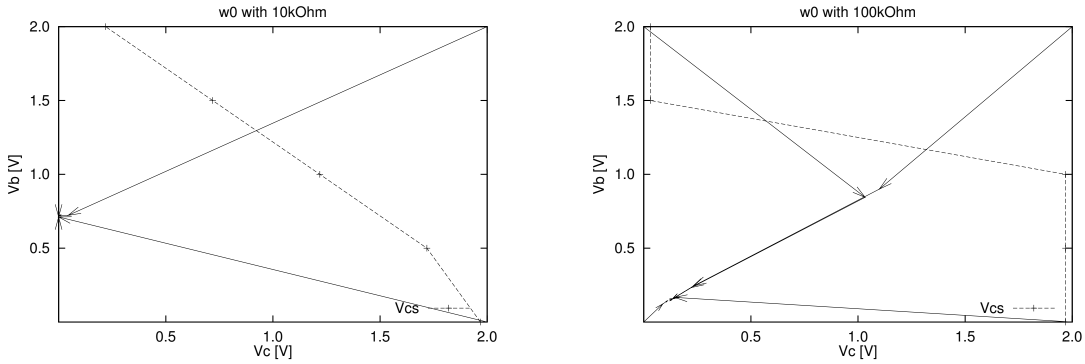


Figure 5.7. Simulation results for $w0$ in the (V_c, V_b) planes for R_{op} = 200 kΩ.

The sequence of $w0$ operations stops when the distance $(D = \sqrt{\Delta V_c^2 + \Delta V_b^2})$ between two (V_c, V_b) points resulting from consecutive $w0$ operations becomes less than D = 0.025 V, which is shown by simulations to be small enough to represent



(a) $w0$ plane with $R_{op} = 10$ kΩ (b) $w0$ plane with $R_{op} = 100$ kΩ

Figure 5.8. Simulation results for $w0$ in the (V_c, V_b) planes for (a) 10 kΩ, and (b) 100 kΩ.

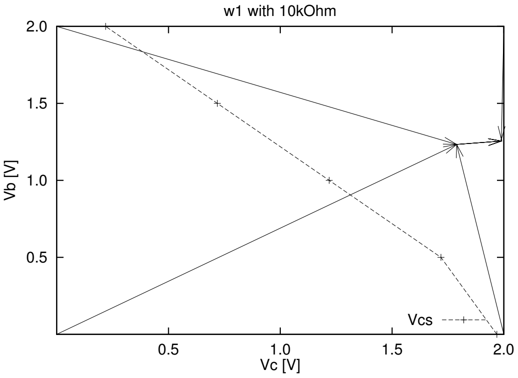
the faulty behavior. This selection of D is arbitrary at the first iteration of the fault analysis and may later be reduced if a more detailed analysis is required.

Curve 6 in Figure 5.7 is the sense threshold cell voltage (V_{cs}) curve, which indicates the cell voltage below which the sense amplifier detects a 0 and above which the sense amplifier detects a 1. This curve is generated by initializing V_b to a specific voltage between 0 V and 2.0 V and then identifying the V_c threshold that distinguishes a 0 and a 1 on the output. Five points have been chosen on V_b, which result in the following V_{cs} points (identified by a little "+" sign in Figure 5.7): $(1.77, 0.00)$, $(1.77, 0.45)$, $(1.77, 0.90)$, $(0.03, 1.35)$ and $(0.03, 1.80)$. Curve 3 in Figure 5.7 shows that, with an initial $(V_c, V_b) = (2.0, 2.0)$, a single $w0$ operation is not able to discharge V_c low enough for the sense amplifier to detect a 0, which means that $w0$ operations fail with $R_{op} = 200$ kΩ.

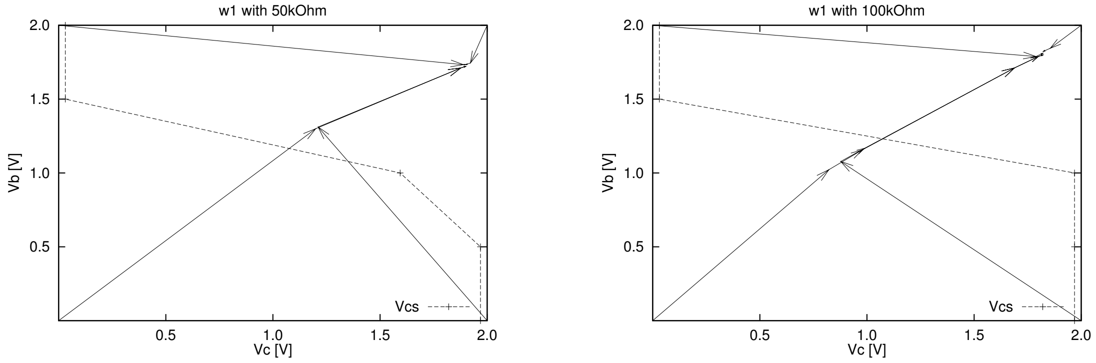
Figure 5.8(a) with $R_{op} = 10$ kΩ and Figure 5.8(b) with $R_{op} = 100$ kΩ are generated in the same way as Figure 5.7. Figure 5.8(a) shows that the $w0$ sequence gradually modifies the floating voltages in the memory such that they eventually settle at $(V_c, V_b) = (0.0, 0.7)$, where V_c has the proper $w0$ value of 0.0 V but V_b has a value of 0.7 V that is different from the proper precharge value of 1.0 V. Figure 5.8(b) shows that the $w0$ sequence results in a voltage equilibrium point of $(0.1, 0.15)$. Both Figures 5.8(a) and (b) show that, irrespective of the initialization, a single $w0$ operation sets (V_c, V_b) to a value below the V_{cs} curve, which means that $w0$ operations behave properly for 10 kΩ $\leq R_{op} \leq$ 100 kΩ.

Planes for $w1$

Figure 5.9 shows three different (V_c, V_b) planes for three R_{op} values, resulting from performing a sequence of $w1$ operations. To generate each plane in the figure, the V_c and V_b voltages are initialized to a given value and then a sequence of $w1w1...w1$ operations is performed on the cell. The net effect is the gradual convergence of V_c and V_b voltages toward a specific point in the (V_c, V_b) plane.



(a) $w1$ plane with R_{op} = 10 kΩ



(b) $w1$ plane with R_{op} = 50 kΩ

(c) $w1$ plane with R_{op} = 100 kΩ

Figure 5.9. Simulation results for $w1$ in the (V_c, V_b) planes for (a) 10 kΩ, (b) 50 kΩ and (c) 100 kΩ.

In the example shown in Figure 5.9, 4 (V_c, V_b) initializations have been used: $(0.0, 0.0)$, $(2.0, 0.0)$, $(2.0, 2.0)$, and $(0.0, 2.0)$; all values are in volts. The sequence of $w1$ operations is stopped when $D \leq 0.025$ V. This selection of $D = 0.025$ V is arbitrary at the first iteration of the fault analysis and may later be reduced if a more detailed analysis is required. In addition to the $w1$ vectors in each result plane, there is also the sense threshold cell voltage (V_{cs}) curve, that indicates the cell voltage (V_c) below which the sense amplifier detects a 0 and above which the sense amplifier detects a 1 for a given value of V_b and R_{op}.

Figure 5.9(a) with R_{op} = 10 kΩ shows that the $w1$ sequence gradually modifies the voltages in the memory toward the point $(V_c, V_b) = (2.0, 1.3)$, which is close to the proper value after $w1$ of $(2.0, 1.0)$. Figure 5.9(b) shows that, with R_{op} = 50 kΩ, the $w1$ sequence gradually modifies memory voltages towards the point $(1.9, 1.7)$. In these two planes, a single $w1$ operation is able to pull V_c up from any initial voltage to a voltage above the V_{cs} curve, which means that with 10 kΩ $\leq R_{op} \leq$ 50 kΩ no fail can be detected on the output.

Figure 5.9(c) with R_{op} = 100 kΩ shows that the $w1$ sequence gradually approaches a voltage point of $(1.8, 1.8)$ that is above the V_{cs} curve, which means that after a sequence of $w1$ operations the memory succeeds in writing a 1 into the cell. However, a single $w1$ operation with an initialization of $(0.0, 0.0)$ or $(2.0, 0.0)$ fails to write a high enough V_c voltage for the sense amplifier to detect a 1. Therefore, $w1$ operations start to fail with an R_{op} value of about 100 kΩ.

Critical resistance & detection condition

The result planes in Figures 5.7 and 5.9 can be used to identify important aspects of the faulty behavior, such as the value of the *critical resistance* (R_{cr}) and a detection condition to detect the defective memory.

The result planes indicate that both $w0$ and $w1$ fail for a specific R_{op} value and with some initial conditions. Since $w1$ fails at an open resistance of about 100 kΩ, which is less than the $R_{op} = 200$ kΩ needed for $w0$ to fail, R_{cr} corresponds to the lower 100 kΩ value.

The failing $w1$ operation can be detected with the sequence $w1r1$, in case V_c and V_b are properly initialized. Figure 5.9(c) shows that $w1$ fails for an initialization of $(0.0, 0.0)$ and $(0.0, 2.0)$, which means that $w1$ fails as long as V_c is equal to 0.0 V irrespective of the value of V_b. Therefore, in order to force $w1$ to fail for R_{op} as close as possible to R_{cr}, V_c should be brought as close as possible to 0.0 V. Figure 5.8(b) shows that, with $R_{op} = 100$ kΩ and for an initialization of $(2.0, 2.0)$, it takes four $w0$ operations (indicated by four arrows) to get to a saturation point P of $(0.1, 0.15)$. The figure also shows that starting with other initial voltages, it takes four or fewer $w0$ operations to get to the same point P. In other words, starting with any initial floating node voltages, the sequence $w0w0w0w0$ gradually modifies the floating voltages toward the point $(V_c, V_b) = (0.1, 0.15)$, which could be used as an initialization for V_c.

In conclusion, the FP that describes this faulty behavior is $<w0^4\ w1/0/->_{BTd}$, where the BTd indicates that the fault can be sensitized in any cell on the defective part of BT. This is a partial hard FP. Because the defect does not force leakage in any given direction, the FP can also be transient and soft. This results in the following two additional FPs: $<\underline{w0^4}\ \underline{w1}/0_L/->$ and $<w0^4\ w1_T/0/->$. The following detection condition is sufficient to detect the hard and transient FP caused by the BL open of Figure 5.6, that ensures the lowest possible failing R_{op} value: $\Updownarrow$(..., $w0$, $w0$, $w0$, $w0$, $w1$, $r1$, ...). The soft FP requires inserting a delay into the detection condition as follows: $\Updownarrow$(..., $w0$, $w0$, $w0$, $w0$, $w1$, Del, $r1$, ...).

5.4 DRAM-specific simulation

This section extends the approximate simulation method to deal with DRAM-specific commands, not only DRAM operations. Then, a method is shown to use simulation for stress optimization in the memory, in order to accelerate failure and reduce the delay time needed to detect DRAM-specific soft faults.

5.4.1 Approximation with DRAM commands

The 1D and 2D approximate simulation presented above are limited to the analysis of the impact of the generic memory operations $w0$, $w1$ and r. They are not capable of performing a detailed analysis of DRAMs, based on the five DRAM-specific commands: Act, Wr, Rd, Pre and Nop. Performing the analysis using DRAM-specific commands is important to generate DRAM-specific tests, used later in Chapter 9.

The 1D analysis of Section 5.2 made it possible to understand the faulty behavior of the memory, for any possible sequence of memory operations, despite the presence of an infinite number of possible sequences of operations. Since it is impossible to simulate all operation sequences, the 1D analysis simulates a limited number of sequences, called *basic sequences*, and then it uses those to *approximate* the behavior of any other sequence. The basic sequences for the 1D analysis above are $w0w0$ ... $w0$, $w1w1$ ... $w1$ and rr ... r.

Therefore, in order to perform an approximation of the faulty behavior using DRAM commands, we need to find those basic sequences made up of DRAM-specific commands that make such an approximation possible.

The main question this set of DRAM-specific basic sequences has to answer is: Given an arbitrary point in the (R_{op}, V_c) plane, how much approximately is the voltage change (ΔV_c) resulting from performing any given DRAM command? This problem is depicted in Figure 5.10, where Command stands for any DRAM command, V_{c1} stands for V_c before Command is applied, and V_{c2} stands for V_c after Command is applied.

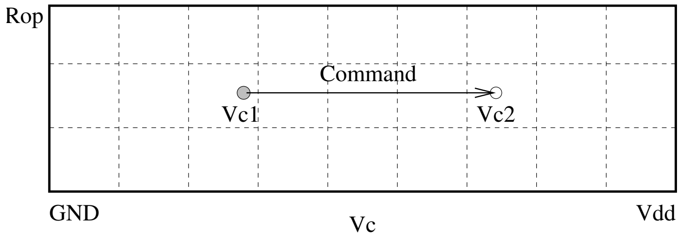


Figure 5.10. Effect of Command on the V_c in the DRAM result plane.

Table 5.1 lists 13 DRAM-specific basic sequences able to approximate ΔV_c for any other command sequence. The basic sequences are classified into four different groups: charge up group, discharge group, charge/discharge group (C/D group), and no charge group. The name in each group refers to the direction of expected voltage change within the memory cell during the application of the sequence. In the following, we show how these basic sequences can be used to approximate the impact of each of the five DRAM commands on V_c.

Wr: The impact of writing commands on V_c does not depend on the previously performed command, but on the data being written. Therefore, it is important

Table 5.1. Basic command sequences used to reconstruct any DRAM sequence.

#	Charge up group	#	Discharge group	#	C/D group	#	No charge group
1.	`Wr1 Wr1 Wr1 ... Wr1`	5.	`Wr0 Wr0 Wr0 ... Wr0`	9.	`Act Rd  Rd  ... Rd`	13.	`Pre Nop Nop ... Nop`
2.	`Wr1 Rd1 Rd1 ... Rd1`	6.	`Wr0 Rd0 Rd0 ... Rd0`	10.	`Act Nop Nop ... Nop`		
3.	`Wr1 Nop Nop ... Nop`	7.	`Wr0 Nop Nop ... Nop`	11.	`Act Act Act ... Act`		
4.	`Wr1 Pre Pre ... Pre`	8.	`Wr0 Pre Pre ... Pre`	12.	`Act Pre Pre ... Pre`		

to inspect the behavior of Wr1 commands (Sequence 1) and the behavior of Wr0 commands (Sequence 5).

Nop: The no operation represents wait states where the memory is supposed not to change its state. During Nop, however, the value of V_c does change (as a result of leakage, for example) and it, therefore, needs to be analyzed. The impact of Nop on V_c depends on the previously performed commands and on the amount of leakage current into the cell. Sequence 3 evaluates the impact of Nop after Wr1, Sequence 7 after Wr0, Sequence 10 after Act, and Sequence 13 after Pre.

Rd: Reading performs an external read from the sense amplifiers to the I/O buffers. The impact of Rd on V_c depends on the previously performed commands. Therefore, Sequence 2 identifies the impact for a previously performed Wr1, Sequence 6 for performed Wr0, and Sequence 9 for performed Act.

Act: Activation performs an internal read by accessing a row of cells and sensing the data the cells contain. Starting with a V_c above V_{cs}, Act increases V_c, while starting with a V_c below V_{cs}, Act reduces V_c. Therefore to identify the impact of Act on V_c, Sequence 11 in Table 5.1 is needed and should be performed twice: starting with V_c above V_{cs} and with V_c below V_{cs}.

Pre: Precharging closes the activated row of cells and sets the voltage of bit lines and data lines to their precharge levels. During this operation, the change in V_c is decided by the previously performed command, which might be Act, Wr1, Wr0, Rd or Nop. Rd and Nop are unimportant, since Rd only performs an external read, which means that it has little impact on internal behavior. The same is true for Nop, which simply represents a wait state and does not influence subsequent Pre operations. It *is* important, however, to identify the impact of Pre on V_c with a previously performed Wr1 (Sequence 4), with a previously performed Wr0 (Sequence 8), and with a previously performed Act (Sequence 12).

5.4.2 Stress optimization methodology

As discussed in Section 4.4, the effectiveness of memory tests does not merely depend on the applied sequence of memory operations. It heavily employs modifications to various operational parameters or *stresses* (*STs*), either to ensure a higher fault coverage of a given test or to target specific failure mechanisms not detected at nominal operational conditions [Falter00, McConnell98]. The STs usually used in testing are timing, temperature, supply voltage.

The fault analysis concept that enables simulation based optimization of STs is the ability to state the critical resistance (R_{cr}) of a defect [Al-Ars03c]. R_{cr} is the resistive value of a defect at which the memory starts to show faulty behavior. Using this important piece of information, the criterion to optimize any ST can be stated as follows:

> A change in a given ST should modify the value of the critical resistance in such a direction which maximizes the resistance range that results in a detectable functional fault.

Figure 5.11 shows a graphical representation of the optimization methodology. The figure shows two different result planes, one is generated with the stress ST1, while the other is generated using the stress ST2. The application of ST2 reduces the value of R_{cr}, thereby increasing the range of the Fail region, relative to the Fail region of ST1. Therefore, the stress ST2 is considered to be more stressful than the stress ST1.

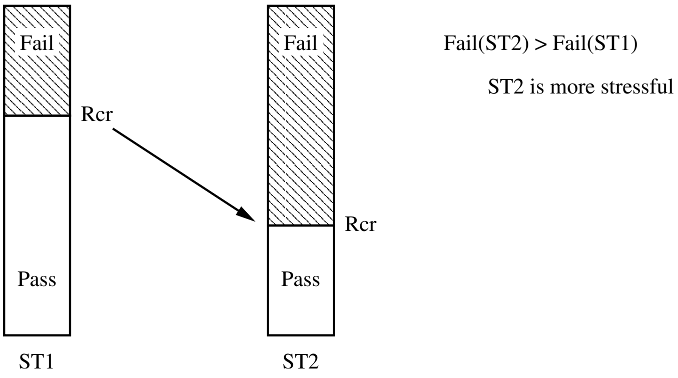


Figure 5.11. Graphical representation of stress optimization.

In the industry, a **Shmoo plot** is considered as an important method used to optimize STs for a given memory test [Baker97]. Two STs (S1 and S2) are chosen to be optimized in a given range. A test is then applied to the memory and, for each combination of S1 and S2, the pass/fail outcome of the test is registered on the Shmoo plot. This creates a two dimensional graphical representation of the pass/fail behavior of the memory under the applied test. Figure 5.12 shows an example of a Shmoo plot, where the x-axis represents the clock cycle time and the y-axis represents the supply voltage V_{dd}. The figure shows, for example, that a lower voltage and a shorter cycle time are the most stressful conditions for the applied test.

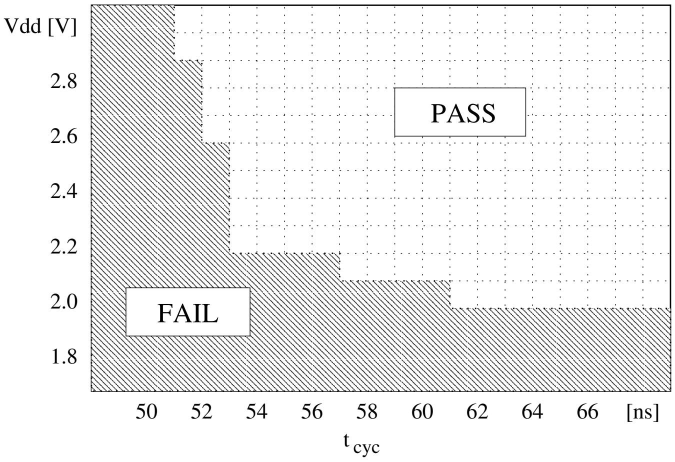


Figure 5.12. Shmoo plot to optimize the cycle time and supply voltage.

Shmoo plotting has the advantage of direct optimization of a pair of STs for a given test on a chip, in case the chip is known to have the targeted defect. Shmoo plotting suffers, however, from the following disadvantages:

1. Depending on the length of the test, generating a Shmoo plot may take large amounts of time since the test has to be repeated for each (x, y) combination in the plot [Hamada93].

2. The tester provides only a restricted controllability and observability of internal parts of the circuit under test [Niggemeyer99].

3. It is not always clear how the externally observed failure of the memory relates to the internal faulty behavior caused by the targeted defect.

4. Since only a limited number of memory devices are investigated, the resulting STs may not be the most optimal for the investigated test and targeted defect.

For a test designer attempting to optimize a given test for a specific defect using Shmoo plots, the above mentioned problems make optimization a rather difficult and challenging task. The simulation-based stress optimization method presented in this section targets these problems and provides more insight into the faulty behavior, an insight that guides a test designer through the process of test optimization, so that test development time can be reduced.

The concept of simulation-based stress optimization will be used in Chapter 9 to optimize test stresses for the elevated strap resistance problem.

Summary

This chapter discussed the simulation-based algorithms used to analyze the faulty behavior of the memory within a practical amount of time. These algorithms solve the two main problems in applying Spice simulation to analyze the faulty behavior of memories: 1. the excessive amount of simulation time needed, and 2. the difficulty to interpret simulation results in terms of faults and tests. The main issues presented in this chapter are the following.

- Discussion of the time complexity of using Spice simulation to precisely analyze the faulty behavior of memories.
- Introduction of the 1D approximate simulation method, to approximate the faulty behavior of the memory, thereby limiting the needed simulation time. The 1D simulation method is only able to analyze defective memories with *one* disconnected (or floating) node, such as the floating cell voltage caused by a cell open.
- Generalization of 1D simulation method to take *two* floating nodes into consideration, such as bit line opens causing a floating cell voltage and a floating bit line voltage.
- Extension of the approximate simulation method to account for DRAM-specific commands. This is done by identifying a minimum set of DRAM-specific command sequences, that could be used to approximate any possible sequence of commands.
- Derivation of a simulation-based stress optimization method, using the concept of the critical resistance. This method can be used to accelerate the Shmoo plot approach, used industrially to optimize memory test stresses.

Contents of this chapter

6.1 Concept of BL coupling
6.2 Simulation of BL coupling
6.3 Impact of BL twisting

6

Effects of bit line coupling

With the shrinking dimensions of manufactured structures on memory chips and the increase in memory size, bit line coupling is becoming ever more influential on the memory behavior. This chapter discusses the effects of BL coupling on the faulty behavior of DRAMs. It starts with an analytical evaluation of coupling effects, followed by a simulation-based fault analysis using a Spice simulation model. Two BL coupling mechanisms are identified (pre-sense and post-sense coupling), which are found to have a partly opposing effect on the faulty behavior. In addition, the influence of BL twisting on the faulty behavior of the memory is analyzed.

This chapter begins with a general discussion of the concept of BL coupling in Section 6.1, where a theoretical analysis of BL coupling is given. Section 6.2 validates the theoretical analysis of BL coupling using a simulation-based fault analysis approach. Finally, Section 6.3 discusses the impact of BL twisting on coupling effects, and on the faulty behavior of defective memories.

6.1 Concept of BL coupling

The long, narrow BL structures running in parallel on the surface of a memory chip are particularly prone to relatively large amounts of *capacitive coupling* (or *crosstalk*) noise from adjacent BLs. As the integration density of memory devices increases, the problems associated with BL coupling noise become more significant because of the weak cell signals that must be sensed reliably on these lines [Redeker02]. This section discusses the concept of BL coupling in general, and the way it influences the faulty behavior of defective memory devices [Al-Ars04b].

6.1.1 Modeling BL coupling

The simulation model used in this chapter is based on a DRAM design validation model from Infineon Technologies, used to simulate the behavior of their memory products. In order to limit the needed simulation time, the model has been reduced in complexity, while electrically compensating removed components. The simulation model contains three folded BL pairs, one of which is shown in Figure 6.1. This model also contains a 2 × 2 cell array with NMOS access transistors, in addition to a sense amplifier and precharge devices. The removed memory cells are compensated for by load cells and parasitic components of different values distributed along the BLs. External to the BL pair, the simulation model contains one data output buffer needed to examine data on the output, and a write driver needed to perform write operations. The memory model employs Spice BSIM3v3 device parameters for the simulations. Two different sets of simulation model parameters are used in Sections 6.2 and 6.3, one corresponding to a 0.20 μm technology and the other corresponding to a 0.14 μm technology, respectively. The sets of model parameters are chosen to be different in order to get two independent sets of data to test the theory. A similar simulation model is also used later in Chapters 7 and 9 to simulate the faulty behavior of the memory.

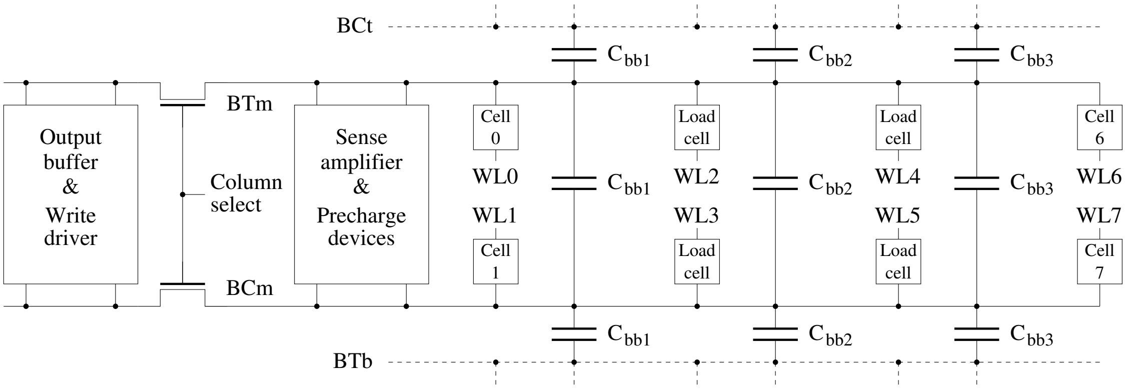


Figure 6.1. Closeup block diagram of one BL pair of the three used for simulation.

The model contains three BL pairs, denoted as BLt for top, BLm for middle, and BLb for bottom, as shown in Figure 6.2. Figure 6.1 shows both the true (BTm) and complementary (BCm) bit lines of BLm, the complementary (BCt) bit line of BLt only, and the true (BTb) bit line of BLb only.

The different BLs influence each other by a number of distributed coupling capacitances ($C_{bb1} + C_{bb2} + C_{bb3} = C_{bb}$). Note that BL coupling capacitances are the same whether coupling takes place within a given BL pair or between different BL pairs. This is true since all BLs on a chip are manufactured in the same way, using the same materials, having the same dimensions, and at the same distances from each other. Identifying a given BL as BT or BC depends solely on the way that BL is connected to the sense amplifier and not on the way it is manufactured.

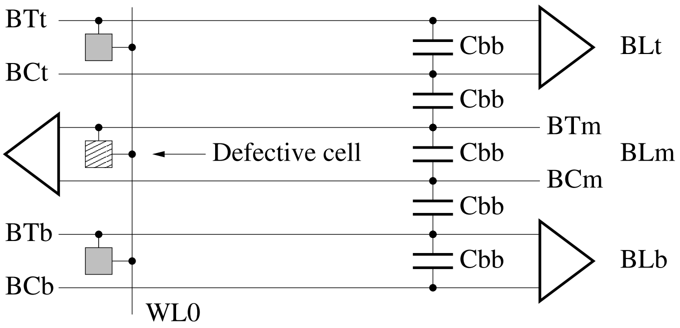


Figure 6.2. Three BL pairs implemented in the simulation model.

The total BL capacitance (C_b) is made up of the sum of the BL coupling capacitance (C_{bb}), and the remaining capacitance (C_{br}) not related to BL coupling, but to word line (WL) coupling, substrate coupling, etc. In the simulation model of Figure 6.1, these capacitances are related as follows:

$$C_b = C_{bb} + C_{br} = 10 \times C_{bb} \tag{6.1}$$

These are typical cell array capacitances in megabit DRAMs with a folded BL pair arrangement [Konishi89, Itoh01]. As a result of C_{bb}, two different kinds of coupling effects may take place: *pre-sense coupling* and *post-sense coupling* [Aoki88].

Pre-sense coupling (ΔV_1) is generated after the WL is activated and cells are accessed, but before the sense amplifier is activated. The noise on a given floating BL results from coupling to two BLs, above and below the victim BL. The amount of worst-case ΔV_1 developing on the floating BL relative to the full voltage V_1 developing on neighboring BLs can be approximated as [Hidaka89]:

$$\frac{\Delta V_1}{V_1} \approx \frac{1}{2 + (C_{br}/C_{bb})} \tag{6.2}$$

This relation indicates that the amount of pre-sense coupling noise increases with increasing C_{bb} from 0 V for $C_{bb} = 0$ to $\frac{1}{2}$ as C_{bb} approaches ∞. For the used simulation parameters in (6.1), the worst-case $\frac{\Delta V_1}{V_1} \approx \frac{1}{11}$.

Post-sense coupling (ΔV_2) is generated after the sense amplifier is activated and the BLs are pulled either to 0 or 1 according to the logic value sensed by the sense amplifier. The main reason for this type of noise is the time difference between sense amplifier activation and the instant the sense amplifier decides to sense a 0 or 1 (Δt). The amount of ΔV_2 can be approximated according to the following relation [Aoki88]:

$$\Delta V_2 \approx \alpha \frac{C_{bb}}{C_b^2} (\Delta t)^3 \tag{6.3}$$

where α is a constant that depends on a number of sense amplifier related parameters and has a value in the order of $10^{12} \sim 10^{13} \frac{\mathrm{FV}}{\mathrm{s}^3}$. The relation shows the strong dependence of ΔV_2 on the time delay until the sense amplifier pulls the BLs either up or down. This means that even small delays in the sense amplifier operation can cause a relatively large amount of post-sense coupling noise.

The total amount of BL coupling noise ΔV is equal to the sum of pre-sense and post-sense coupling ($\Delta V_1 + \Delta V_2$). Whether ΔV_1 or ΔV_2 constitutes the dominant factor in ΔV depends heavily on design specific parameters that generally cannot be evaluated analytically, which leaves circuit simulation as the only analysis option [Itoh01].

6.1.2 Effects of coupling

BL coupling results in developing small coupling voltages on adjacent BLs, which influences proper sense amplifier operation. From a testing point of view, it is important to understand how a specific initialization of a neighborhood of cells affects the sensing of a given victim, so that the worst-case values can be written in the neighboring cells.

The model considered here [see Figure 6.2] consists of 3 BL pairs, each with 2×2 cells, which means that the defective cell (Cell 0 on BLm) has a neighborhood of $3 \times 2 \times 2 = 12$ cells with a possible influence on the behavior. But since the precharge operation functions properly (assuming that the defective cell suffers from an open within the cell, which does not influence the precharge voltage on the BLs), the history of operations performed on any cell other than the defective cell does not influence the faulty behavior of the memory[1]. Therefore, the only cells able to influence the faulty behavior are those sharing the same WL with the defective cell. This means that the neighborhood consists of two cells, each containing either 0 or 1, which results in $2^2 = 4$ different data backgrounds.

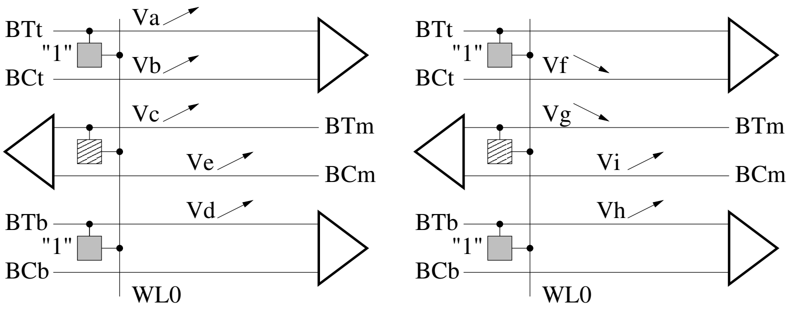
The effects of BL coupling on the faulty behavior can be divided into pre-sensing effects, and post-sensing effects. Figure 6.3 gives graphical representations for both cases, when Cell 0 on BTt contains a logic 1 and Cell 0 on BTb contains a logic 1.

Pre-sensing effects. As soon as WL0 is accessed, Cell 0 on BTt starts to pull the voltage on BTt by an amount of V_a to a higher level; this is indicated by the up-arrow next to V_a in the figure. As a result of C_{bb}, the voltage on BCt is also pulled by an amount of V_b to higher level; this is indicated by the up-arrow next to V_b in the figure. Finally, as a result of C_{bb} between BCt and BTm, the voltage on BTm is pulled higher by an amount of V_c, which promotes sensing a logic 1 in the victim; this is indicated by the up-arrow next to V_c in the figure[2]. From Equations 6.1 and 6.2, the amount V_c is related to V_a by the relation $\frac{V_c}{V_a} = \frac{V_b}{V_a}\frac{V_c}{V_b} \approx \frac{1}{11^2}$. In the same way, as soon as WL0 is accessed, Cell 0 on BTb starts to pull the voltage on BTb

---

[1] On the other hand, a defective precharge circuitry would mean that the read/write history affects the faulty behavior and should be simulated

[2] The increase in the voltage on BTm further results in an increase in the voltage on BCm, but this effect is an order of magnitude less and is therefore negligible



(a) Pre–sensing (b) Post–sensing

Figure 6.3. Effects of (a) pre-sense and (b) post-sense coupling.

by an amount of V_d to a higher level, which in turn pulls the voltage on BCm by an amount of V_e higher. This increase in the voltage on BCm promotes sensing a *logic 0* in the victim cell. The values of V_d and V_e are related by $\frac{V_e}{V_d} \approx \frac{1}{11}$, which means that the cell on BTb has a much higher influence on the faulty behavior than the cell on BTt. In conclusion:

1. The worst-case pre-sensing BG is $\bar{x}_{a_t} x_v x_{a_b}$ (i.e., Cell 0 on BTt contains $\bar{x}$ and Cell 0 on BTb contains x).
2. Cell 0 on BTb has a much higher pre-sensing influence (first-order effect) on the faulty behavior than Cell 0 on BTt (second-order effect).

Post-sensing effects. Once the sense amplifier is activated, and since Cell 0 on BTt contains 1, the sense amplifier pulls the voltage on BTt high while the voltage on BCt is pulled low by an amount of V_f [see Figure 6.3(b)]. As a result of C_{bb}, the voltage on BTm is pulled low by an amount of V_g, which promotes sensing a logic 0 in the victim cell. In a similar way, once the sense amplifier is activated, and since Cell 0 on BTb contains a 1, the sense amplifier pulls the voltage on BTb high by an amount of V_h as indicated in Figure 6.3. As a result of C_{bb}, the voltage on BCm is also pulled high by an amount of V_i, which promotes sensing a logic 0 in the victim cell. Both neighboring cells have a first-order effect on the victim. In conclusion:

1. The worst-case post-sensing BG is $x_{a_t} x_v x_{a_b}$ (i.e., Cell 0 on BTt contains x and Cell 0 on BTb contains x).
2. Both cells have a comparable first-order effect on the faulty behavior.

Comparing the two results of pre and post-sensing, we find that each requires a different BG to ensure the worst-case sensing condition. It is possible to use a memory test that covers both BGs to ensure covering the worst-case condition. But to reduce test time, a single worst-case BG is needed, and therefore we should identify whether pre-sensing or post-sensing is more dominant.

6.2 Simulation of BL coupling

This section uses Spice simulation to evaluate the impact of BL coupling on the faulty behavior of DRAMs. We start with a discussion of the simulation-based fault analysis method, then we use it to evaluate the behavior with the presence of coupling.

6.2.1 Fault analysis method

The fault analysis method described here is the approximate simulation method (or one dimensional analysis) described in much more detail in Section 5.2. The analysis performed here corresponds to BLs with *no* coupling. This means that the coupling capacitances shown in Figure 6.1 are all set to zero: $C_{bb1} = C_{bb2} = C_{bb3} = 0$.

Consider the defective DRAM cell shown in Figure 6.4, where a resistive open (R_{op}) between BT (true bit line) and the access transistor limits the ability to control and observe the voltage across the cell capacitor (V_c). The open is injected into Cell 0 and simulated as part of the reduced memory model shown in Figure 6.1. The reasons for choosing this specific cell defect to analyze BL coupling include the following.

1. This defect models a strap connection between the drain of the pass transistor and the cell capacitor that is difficult to manufacture and may have resistive values that are higher than normal [Adler95].

2. Gradually, increasing the resistive value of this defect results in the gradual reduction of the differential BL signal needed for proper sensing. Therefore, this defect is ideal for analyzing the impact of BL coupling on the faulty behavior.

3. The relative simplicity of the defect model and the required fault analysis.

The analysis takes a range of possible open resistances (10 kΩ $\leq R_{op} \leq$ 10 MΩ) and a range of possible cell voltages (GND $\leq V_c \leq V_{dd}$) into consideration.

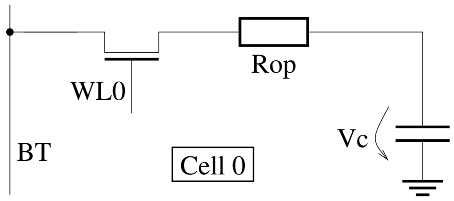
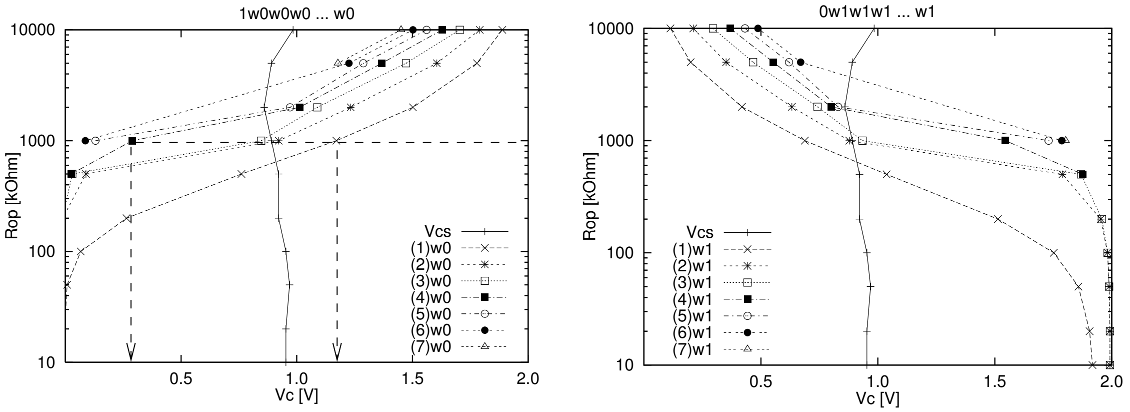


Figure 6.4. Open injected into Cell 0.

Two different (V_c, R_{op}) result planes are generated, one for the $w0$ operation on a cell initialized to one ($1w0$) and one for the $w1$ operation on a cell initialized to 0

($0w1$). These result planes describe the impact of successive $w0$ and successive $w1$ operations on V_c (denoted as $(n)w0$ and $(n)w1$, respectively), for a given value of R_{op}. Write operations described here refer to single-cycle operations, where a cell is accessed, written, then disconnected, and followed by a memory precharge. Figure 6.5(a) shows an automatically generated result plane corresponding to $(n)w0$ operations, while Figure 6.5(b) shows the result plane corresponding to $(n)w1$ operations, for the open R_{op} shown in Figure 6.4.



(a) $w0$ plane

(b) $w1$ plane

Figure 6.5. Simulation results in the (V_c, R_{op}) plane for (a) $(n)w0$, and (b) $(n)w1$.

Plane of $w0$: This result plane is shown in Figure 6.5(a). To generate this figure, the floating cell voltage V_c is initialized to V_{dd} (because a $w0$ operation is performed) and then the operation sequence $1w0w0...w0$ is applied to the cell (i.e., a sequence of $w0$ operations to a cell initialized to 1). The net result of this sequence is the gradual decrease (depending on the value of R_{op}) of V_c towards GND. The voltage level after each $w0$ operation is recorded on the result plane, resulting in a number of curves. The curves are numbered as $(n)w0$, where n is the number of $w0$ operations needed to get to the curve. For example, the arrows in the figure indicate that, for $R_{op} = 1000$ kΩ, a single $w0$ operation represented by $(1)w0$ pulls V_c from V_{dd} to about 1.2 V, while four $w0$ operations represented by $(4)w0$ pull V_c to about 0.3 V. We stop performing the $w0$ sequence when the voltage change ΔV_c, as a result of $w0$ operations, becomes $\Delta V_c \leq 0.05$ V, which results in identifying up to 7 different $w0$ curves in the plane. Initially, an arbitrary small value for ΔV_c is selected, which can be reduced afterwards if it turns out that more than 7 $w0$ operations are needed to describe the faulty behavior. The *sense threshold cell voltage (V_{cs})*, shown as a solid line that runs across the center of the figure, is the cell voltage *above* which the sense amplifier reads a 1, and *below* which the sense amplifier reads a 0. This curve is generated by performing a read operation for a number of V_c values and iteratively identifying the V_c border that distinguishes a

1 and a 0 on the output. V_{cs} is almost independent of R_{op} here because there is no BL coupling considered in this simulation since all BL coupling capacitances are set to zero ($C_{bb1} = C_{bb2} = C_{bb3} = 0$). The small deviation V_{cs} has from the center of the figure is due to sense amplifier imbalance and other types of coupling, such as WL-BL coupling.

Plane of $w1$: This result plane is shown in Figure 6.5(b). The plane is generated in the same way as the result plane of $w0$. First, V_c is initialized to GND and then the operation sequence $0w1w1...w1$ is applied to the cell. The result is a gradual increase of V_c towards V_{dd}. The voltage level after each $w1$ operation is recorded on the result plane, which gives a number of curves in the plane. We stop the $w1$ sequence when ΔV_c becomes small enough (0.05 V in this example). V_{cs} is also shown in the figure as a solid line.

As discussed in Section 5.2, it is possible to use the result planes to analyze a number of important aspects of the faulty behavior such as: the resistive value of the defect where the memory starts to fail, and the test needed to detect the faulty behavior resulting from the open defect [Al-Ars02b].

6.2.2 Simulation results

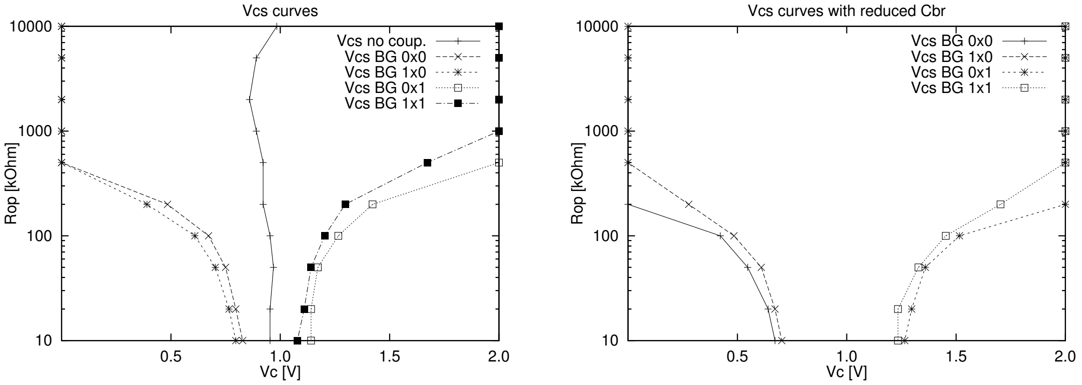
This section presents the simulation results of the effects of BL coupling on the faulty behavior of the memory model shown in Figure 6.1, having the cell open shown in Figure 6.4. The device parameters used in the simulation model correspond to a memory manufactured in 0.20 μm technology. The analysis method used here is the same as that outlined in Section 6.2.1. Four different simulations are performed, one for each data background (BG): BG $0x0$ (both aggressors on BTt and BTb are 0, while the stored voltage of the victim is floating), BG $0x1$ (cell on BTt is 0 and on BTb is 1), BG $1x0$ (cell on BTt is 1 and on BTb is 0) and BG $1x1$ (both cells are 1).

The analysis results show that the write curves are very similar in all BGs, and are also similar to the $(n)w0$ curves shown in Figure 6.5(a) and the $(n)w1$ curves shown in Figure 6.5(b). Therefore, they are not significantly influenced by BL coupling, and are not discussed further.

The effects of BL coupling on the faulty behavior are evident in the way the V_{cs} curve is influenced. This is expected since the V_{cs} curve is closely associated with the amount of differential voltage developing on a given BL pair. Figure 6.6(a) shows four different V_{cs} curves for the simulated BGs, plus the one resulting in the case of zero coupling (no coup.) [see Figure 6.5(a)].

The figure shows that for a victim with a stored 0, the worst-case coupling is generated with BG $1x0$, then with BG $0x0$, followed by the case with no coupling, then BG $1x1$ and finally BG $0x1$. For a victim with a stored 1, the worst-case coupling is generated with BG $0x1$, then BG $1x1$, no coupling, BG $0x0$, and finally BG $1x0$. This means that the worst-case condition for a victim containing value x

corresponds to BG $\bar{x}xx$. Comparing these results with the theoretical analysis of Section 6.1.2 indicates that the dominant BL coupling effect in the simulations of Figure 6.6(a) is pre-sense coupling.



(a) BL coupling

(b) Modified BL coupling

Figure 6.6. V_{cs} curves showing effects of (a) regular BL coupling, and (b) modified BL coupling.

The figure also shows that the most influential cell on the behavior is the one on BTb, which is expected since it generates cooperating pre-sense and post-sense coupling effects. The cell on BTt has a limited impact on the behavior since it generates opposing pre-sense and post-sense coupling effects. It is interesting to note that, for a given victim value, some BGs actually *help* the faulty behavior to produce the correct read output, which indicates the importance of selecting the worst-case background values.

In order to check the correct correspondence between simulation results and the theoretical analysis of Section 6.1.2, one could ask the question: Is it possible to modify the simulation model in such a way that would make post-sensing more dominant than pre-sensing?

Referring to Equations 6.2 and 6.3, ΔV_1 and ΔV_2 are related as follows:

$$\frac{\Delta V_1}{\Delta V_2} \sim \frac{(C_{br} + C_{bb})^2}{C_{br} + 2C_{bb}} \tag{6.4}$$

where we consider Δt as a constant. This relationship indicates that in order to increase the relative impact of ΔV_2 compared to ΔV_1, the ratio in Equation 6.4 should be reduced, which in turn can be done by reducing C_{br}. Figure 6.6(b) shows four new V_{cs} curves corresponding to a modified simulation model with $C_{br2} = \frac{C_{br}}{2}$. The figure shows that for a victim containing a 0, the worst-case condition is ensured by BG $0x0$, then BG $1x0$, BG $1x1$, and finally BG $0x1$. On the other hand, the worst-case condition for a cells containing a 1 is ensured with BG $0x1$, then BG $1x1$, BG $1x0$, and finally BG $0x0$. This represents a mixed behavior where the post-sensing effects are dominant when sensing a 0 in the victim, while pre-sensing effects are dominant when sensing a 1 in the victim [see Section 6.1.2]. Reducing C_{br} further does not change this mixed behavior.

It is worth noting that post-sensing effects are also able to dominate both sensing a 0 and a 1 in the victim. This situation is shown in the simulation results of the analysis in Section 6.3.2, where a memory simulation model with different device parameters is used.

In conclusion, depending on the specific memory design and fabrication technology, either pre-sensing or post-sensing effects (or both) may dominate the resulting faulty behavior. This, in turn, means that unless an analysis is done to identify the exact coupling effects for a specific memory, then all possible worst-case data backgrounds have to be considered during testing.

6.3 Impact of BL twisting

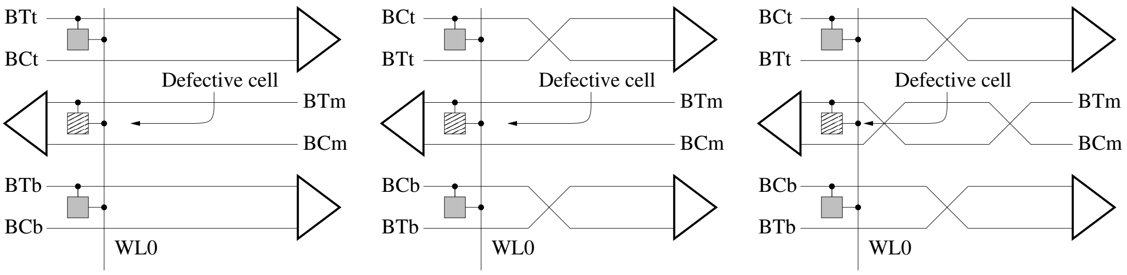
Research in crosstalk reduction on BLs in memory devices indicates the effectiveness of BL twisting techniques in eliminating BL noise in current and future fabrication technologies [Redeker02]. In addition, there is some published qualitative analysis of the possible impact of BL twisting on the faulty behavior of memories [Schanstra03]. This section quantitatively investigates the influence of BL twisting on the faulty behavior of DRAMs, both theoretically and using an electrical simulation model to evaluate the behavior. A number of BL twisting techniques are evaluated and the way a neighborhood of cells influences the behavior is shown [Al-Ars04c].

6.3.1 Theoretical evaluation of twisting

BL twisting is used to reduce the influence of BL coupling on the behavior of the memory, by shielding parts of a BL from neighboring BLs. There are many types of BL twisting schemes used in the industry. Figure 6.7 compares the untwisted BL scheme with other important twisted BL organizations: a. the solid BLs (no twist), b. the single BL twist, and c. the triple BL twist [Aoki88]. These BL organizations are known to be effective at reducing crosstalk between adjacent BLs and they have all been used in commercially produced memory components [Redeker02]. Note how BL twisting modifies the type of BLs simultaneously accessed, resulting in cells on both BT and BC being accessed at the same time with a given WL rather than cells connected only to BT or BC. In Section 6.1.2 above, we analyzed the impact of solid BLs on the behavior, and in this section, we discuss the impact of single and triple twisting on the behavior.

Single twist

Close consideration of the impact of the single twist on memory behavior shows that the single twist fails in completely eliminating pre-sense BL coupling but succeeds in eliminating post-sense coupling. The net effect is that the single twist results in

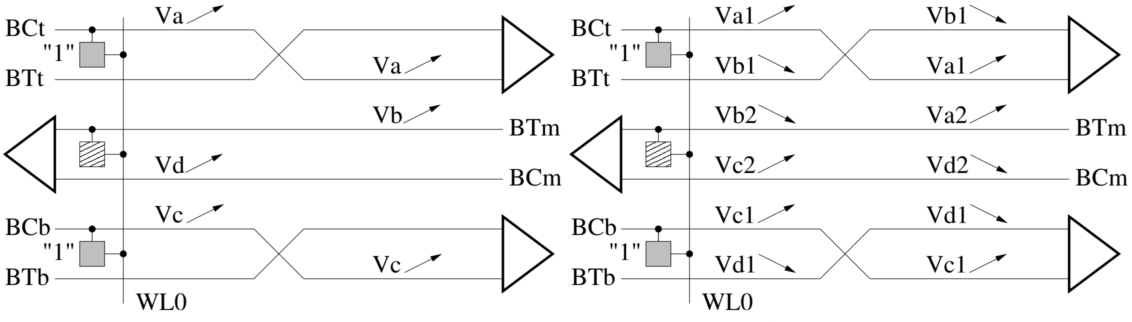


(a) Solid BLs (b) Single BL twist (c) Triple BL twist

Figure 6.7. Analyzed BL organizations: (a) solid BLs, (b) single twist, and (c) triple twist.

making pre-sense coupling more significant than post-sense coupling on the faulty behavior of the memory. In the following, a more detailed discussion of this point is given.

For pre-sense coupling, Figure 6.8(a) shows the pre-sense voltage development on BTm and BCm as a result of a logic 1 stored in both neighboring cells. Since the accessed cell on BCt has a logic 1, a small upward voltage differential of V_a develops on BCt during pre-sensing, which in turn results in pulling BTm up by V_b. As a result of BL twisting, the amount of V_b developing on BTm is almost one half of that in the case of solid BLs[3]. The same situation takes place with BCb, where an up voltage differential of V_c induces an up voltage differential of V_d on BCm. In conclusion, the single twist cuts the amount of pre-sense BL coupling by almost one half, where the remaining coupling effect requires a worst-case BG of $\bar{x}_{a_t}x_vx_{a_b}$ (i.e., Cell 0 on BCt contains $\bar{x}$ and Cell 0 on BCb contains x).



(a) Pre-sense (b) Post-sense

Figure 6.8. Voltage development for a single twist during (a) the pre-sense, and (b) the post-sense stages.

For post-sense coupling, Figure 6.8(b) shows the voltage development on BTm and BCm as a result of a logic 1 stored in both neighboring cells. During post-sensing, and since the accessed cell on BCt has a logic 1, the top sense amplifier

---

[3] It is actually slightly higher than one half as a result of second-order coupling which is not considered in this discussion

pulls the voltage on BCt up by an amount of V_{a1}, which induces an up differential voltage of V_{a2} on BTm. At the same time, the top sense amplifier pulls the voltage on BTt down by an amount of V_{b1} resulting in a down voltage of V_{b2} on BTm. Since V_{a2} and V_{b2} are equal and opposite to each other, they nullify each other leaving a zero net coupling voltage on BTm. The same situation takes place with the bottom sense amplifier, which pulls BCb down by V_{c1} inducing V_{c2} on BCm, and pulls BTb up by V_{d1} inducing V_{d2} on BCm. V_{c2} and V_{d2} nullify each other leaving a zero net coupling voltage on BCm. In conclusion, the single BL twist totally eliminates post-sense coupling effects.

Triple twist

Close consideration of the impact of the triple twist on the memory behavior reveals that the triple twist succeeds in completely eliminating the influence of BL coupling, both pre-sense and post-sense coupling. This means that one cannot evaluate the impact of the triple twist on the faulty behavior of the memory by theoretically analyzing the effect of BL coupling, and therefore electrical simulation here becomes necessary to evaluate the second-order coupling effects on the faulty behavior.

This can be understood by noticing that the triple twist splits the BL into four equal parts, such that one half of any coupling effect is induced onto BTm while the other half is induced onto BCm. Since only a voltage *differential* between BTm and BCm is able to influence the behavior of the memory, then a common change (whether increasing or decreasing) in the voltage of BTm and BCm has no impact on the behavior. In other words, the triple twist transforms the differential mode noise into common mode noise for which the sense amplifier is insensitive.

As an example, Figure 6.9 shows the pre-sense coupling voltage development on BTm and BCm as a result of a logic 1 stored in both neighboring cells. During pre-sensing, and since the accessed cell on BCt has a logic 1, an up voltage of V_a develops on BCt, which in turn results in pulling BCm up by V_{b1} and in pulling BTm up by V_{b2} at the same time. Both V_{b1} and V_{b2} are equal and therefore do not result in a differential voltage developing between BTm and BCm. The same situation takes place with BCb, where an up voltage V_c induces an up voltage of V_{d1} on BCm and an up voltage of V_{d2} on BTm. Since both V_{d1} and V_{d2} are equal, they do not result in a differential voltage developing between BTm and BCm.

6.3.2 Simulation and analysis results

This section presents the results of the simulation analysis of the three different BL organizations shown in Figure 6.7 and discusses their impact of the faulty behavior. The memory simulation model employed for the analysis is the same as the one shown in Figure 6.1, while the device parameters used in the simulation model correspond to a memory manufactured in 0.14 μm technology. These device parameters are chosen to be different from those used in Section 6.2.2, in order to get a new and independent set of results to test the theory. The simulation is based

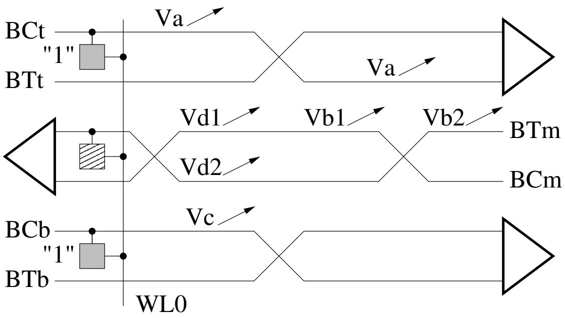


Figure 6.9. Pre-sense voltage development for a triple twist.

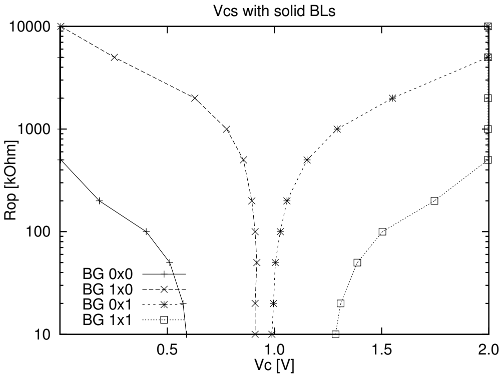
on the concepts of the result planes and the cell sense voltage (V_{cs}) curves, used to evaluate the faulty behavior in Section 6.2.2.

Consider the defective DRAM cell shown in Figure 6.4, where a resistive open (R_{op}) between BT (true bit line) and the pass transistor limits the ability to control and observe the voltage across the cell capacitor (V_c). The open is injected into Cell 0 and simulated as part of the reduced memory model shown in Figure 6.1.

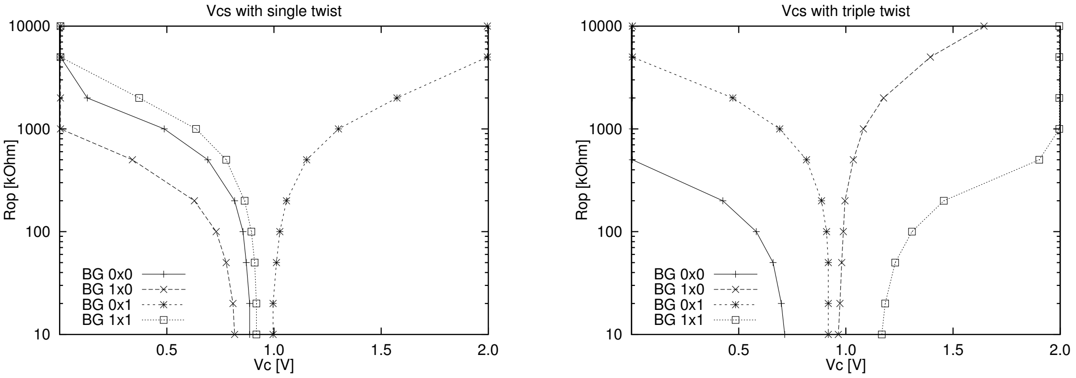
Figure 6.10 shows three result planes for the three BL organizations discussed above: a. solid BLs, b. single twist and c. triple twist. The x-axis of the result plane represents the value of the voltage within the cell (V_c), while the y-axis represents the value of the defect resistance (R_{op}). Each result plane shows 4 different V_{cs} curves, one for each of the four possible BGs: $0x0$, $1x0$, $0x1$ and $1x1$. The V_{cs} curve is the cell sense threshold voltage, which is the cell voltage at which the sense amplifier distinguishes a 0 from a 1. This means that if a read operation is performed when $V_c > V_{cs}$ then the sense amplifier detects a logic 1, while $V_c < V_{cs}$ results in sensing a logic 0. Therefore, the leftmost V_{cs} curve in any of the result planes is associated with the worst-case BG for detecting a 0, while the rightmost V_{cs} curve is associated with the worst-case BG for detecting a 1.

Figure 6.10(a) presents the 4 BGs associated with the solid BL organization, where no twisting is used. The figure shows that the worst-case BG for detecting a 0 in the defective cell is $0x0$, while the worst-case BG for detecting a 1 in the cell is $1x1$. In other words, a worst-case BG of xxx is needed, which means that the post-sense coupling effect is prevalent for the simulated memory model according to Section 6.1.2.

Figure 6.10(b) shows the 4 BGs associated with the single twist BL organization, which according to the analytical evaluation presented in Section 6.3.1 should only be affected by pre-sense BL coupling. The figure shows that the worst-case BG for detecting a 0 in the cell is $1x0$, while the worst-case BG for detecting a 1 in the cell is $0x1$. In other words, a worst-case BG of $\bar{x}xx$ is needed, which indeed matches that of a pre-sense coupling effect. This means that by introducing the single twist into the model, post-sense coupling effects (prevalent before introducing the twist, as indicated by Figure 6.10(a)) have been neutralized, which makes pre-sense



(a) Solid BLs



(b) Single twist

(c) Triple twist

Figure 6.10. V_{cs} curves with different BGs for the (a) solid BL organization, (b) single twist, and (c) triple twist.

coupling effects become the more prevalent sort of BL coupling.

Figure 6.10(c) shows the 4 BGs associated with the triple twist BL organization, which according to the analytical evaluation presented in Section 6.3.1 should totally eliminate pre-sense as well as post-sense BL coupling effects. The figure shows that the worst-case BG for detecting a 0 in the cell is $0x0$, while the worst-case BG for detecting a 1 is $1x1$, which indicates an existing post-sense coupling effect on the defective cell. Although this may seem to be contradictory with the analytical evaluation of Section 6.3.1, it can actually be explained by the fact that Section 6.3.1 only takes into consideration first-order BL coupling effects caused by BL to BL coupling capacitances. It is important to note, however, that second-order BL coupling exists in the form of BL to "line" to BL coupling (BL to WL to BL, for example), which is negligible when compared to the direct first-order BL to BL coupling. But in the absence of first-order coupling, second-order coupling becomes the major factor in the faulty behavior. This is also indicated by the fact that the V_{cs} curves of Figure 6.10(c) are located closer together than those of Figure 6.10(a), which means that the post-sense coupling effect with the triple twist represents a fraction of the full post-sense coupling effect without any twists.

The analysis also shows that the single twist v_{cs} curves in Figure 6.10(b) are closer together than the triple twist v_{cs} curves in Figure 6.10(c). This can be thought to indicate that the single twist is more effective in eliminating bit line coupling effects than the triple twist, which contradicts the theoretical analysis in Section 6.3.1. This can, however, also be explained by assuming the effect of post-sense coupling to be much larger than the effects of pre-sense coupling, thereby making that post-sense effect always dominant if it is not eliminated from the behavior. This can be mathematically illustrated as follows.

$$\begin{aligned} V_{0t} &= \Delta V' + \Delta V'' && (6.5)\\ &= \Delta V' + f \cdot \Delta V' && (6.6)\\ &= \Delta V_2 - \Delta V_1 + f \cdot \Delta V_2 - f \cdot \Delta V_1 && (6.7) \end{aligned}$$

where V_{0t} represents the coupling voltage in the case of solid BLs ($0t$ stands for 0 twists), V' and V'' stand for first-order and second-order coupling effects, respectively, and f represents the *factor* that relates V' to V''. Since second-order effects are supposed to be a fraction of first-order effects, f should satisfy the relation $0 < f < 1$. V_1 and V_2 stand for pre and post-sense coupling voltages [see Equations 6.2 and 6.3].

The coupling voltages in the case of the single BL twist (V_{1t}) and the triple BL twist (V_{3t}) can be represented as:

$$\begin{aligned} V_{1t} &= \frac{1}{2}\Delta V_1 + f \cdot \Delta V_1 - f \cdot \Delta V_2 && (6.8)\\ V_{3t} &= f \cdot \Delta V_2 - f \cdot \Delta V_1 && (6.9) \end{aligned}$$

The fact that the impact of the triple twist coupling voltage seems to be bigger than the impact of the single twist coupling voltage can be represented as:

$$\begin{aligned} V_{3t} > V_{1t} &\implies f \cdot \Delta V_2 - f \cdot \Delta V_1 > \frac{1}{2}\Delta V_1 + f \cdot \Delta V_1 - f \cdot \Delta V_2 && (6.10)\\ &\implies f > \frac{\Delta V_1}{4(\Delta V_2 - \Delta V_1)} && (6.11) \end{aligned}$$

Applying the condition that $0 < f < 1$ to Equation 6.11, results in the following condition that relates pre and post-sense coupling voltages:

$$\Delta V_2 > \frac{5}{4}\Delta V_1 \qquad (6.12)$$

Condition 6.12 states that, even though second-order effects are a fraction f of first-order effects, it is possible for V_{3t} to be bigger than V_{1t} as long as post-sense effects (ΔV_2) remain large enough (relative to ΔV_1).

In the case of the triple twist, analytical evaluation of second-order BL coupling effects on the behavior is overly complex, and electrical simulation can provide considerable insight into the faulty behavior of a defective memory.

Summary

This chapter discussed the impact of BL coupling on the faulty behavior of DRAM devices. For some defects, BL coupling has a significant influence on the faulty behavior, that should be taken into consideration when testing the memory for possible faults. The analysis identified two main causes of BL coupling: 1. pre-sense coupling effects, and 2. post-sense coupling effects. Both of them are analyzed and validated using a simulation-based fault analysis approach. In the coming chapters, these principles are used to take BL coupling into consideration when realistic tests are derived for the memory. The main issues presented in this chapter are the following.

- Introduction of the principle of BL coupling and its importance for current and future memory devices.
- Theoretically discussing the causes and consequences of BL coupling on the faulty behavior of the memory.
- Validation of the theoretical discussion of BL coupling using a simulation-based fault analysis approach of a defective memory, where the simulated faulty behavior is shown to support the theoretical analysis.
- Evaluation of the effectiveness of two industrial BL twisting techniques (single twist and triple twist) in eliminating the effects of BL coupling.
- Identification of the way these BL twisting techniques modify the faulty behavior of the memory.
- Validation of the theoretical discussion of BL twisting using a simulation-based fault analysis approach of a defective memory.

Contents of this chapter

7.1 Memory simulation model
7.2 Classification of defects
7.3 Application of 1D analysis
7.4 Application of 2D analysis

7

Application of the approximation method

In order to demonstrate the effectiveness and practicality of the approximate simulation fault analysis methods presented in Chapter 5, these methods have been applied in an industrial setting at Infineon Technologies on real memory simulation models manufactured in a number of different technologies, ranging from 0.35 μm to 0.11 μm feature sizes. This chapter applies the approximate simulation methods to analyze the faulty behavior of a large number of realistic defects that may take place in a DRAM. The analysis employs the simulation model of a real DRAM, manufactured by Infineon in a 0.2 μm technology, and injects resistive defects into the model. Both approximate algorithms (the one dimensional and the two dimensional simulation methods) discussed in previous chapters are applied here, and the results of the analysis indicate the effectiveness of the approximate simulation methods to analyze the faulty behavior of the memory.

Section 7.1 starts the chapter by presenting the simulation model used for simulating the memory behavior, and discusses the different ways employed to reduce the size of the simulation model. Then, Section 7.2 enumerates and classifies possible defects that can take place in the memory. The classification divides the set of all possible defects into opens, shorts and bridges, some of which result in *one* floating node, while others result in *two* floating nodes. Section 7.3 discusses the fault analysis results of applying the one dimensional (1D) approximate method to those defects causing a single floating node, while Section 7.4 discusses the fault analysis results of applying the two dimensional (2D) approximate method to those defects causing two floating nodes.

7.1 Memory simulation model

In this section, we present the simulation model to be used for defect injection and analysis of the faulty behavior of the memory. The ideal situation would be to inject a defect at all possible locations in the memory, and then to inspect the memory for possible faulty behavior by performing a simulation of the full memory model. The problem with this approach is the huge amount of possible defect combinations, and the long simulation time required. To get an impression of the time needed for such a task, it is enough to note that performing a full memory model simulation of a single memory operation that is 40 ns long, may take more than 5 hours of simulation time. Therefore, it is important to reduce the memory simulation model, so that only relevant circuits are included, leaving out other circuits unnecessary for the simulation [Al-Ars01e, Naik93, Nagi96]. In the following, the circuits used in the simulation are discussed first, followed by the methods used to reduce the size of the simulation model.

7.1.1 Simulated DRAM circuits

As discussed in Section 3.2, memory circuits can be divided into three signal paths: the data path, the address path, and the control path. The data path is the part of the memory that processes and regulates the data flow within the memory, while the other two paths provide the timing and control needed for the data path to function properly. The data path is able to provide the full basic functionality of memory, from storing input data during a write operation, to reproducing this data during a read operation. Therefore, it is possible to restrict the simulation model of the memory to the data path, without any loss of simulated memory functionality.

Figure 7.1 shows the different parts of the data path used for the simulation model of the memory [compare with Figure 3.5]. The model has the following components:

- Three BL pairs, each containing 2 memory cells
- Three sense amplifiers (SAs)
- Three precharge circuits
- Access devices
- One read buffer
- One write buffer

The model contains three BL pairs, one at the top (BTt and BCt), one in the middle (BTm and BCm), and one at the bottom (BTb and BCb). One reason three BL pairs are included in the model is the fact that we would like to evaluate the impact of different *backgrounds patterns* (*BGs*) on the simulated faulty behavior.

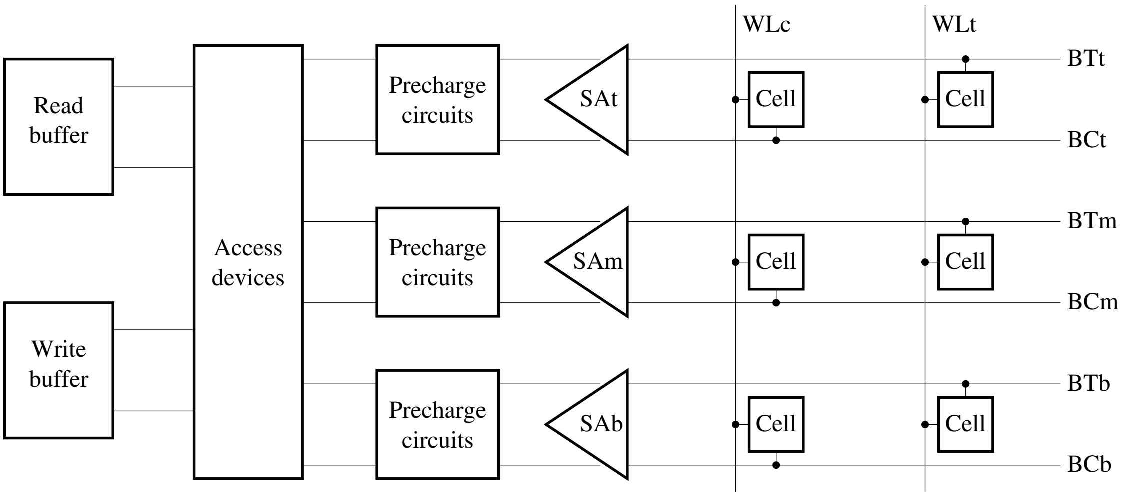


Figure 7.1. Block diagram of the memory simulation model used for fault analysis.

Each BL pair has a couple of cells, one connected to the true BL (BT), while the other is connected to the complement BL (BC). This gives a total of 6 memory cells, three of which are connected to a true BL and controlled by the word line WLt, while the other three are connected to a complement BL and controlled by WLc.

Each BL pair is connected to a sense amplifier and has its own precharge devices. Furthermore, the three BL pairs are connected through the access devices to a read buffer and a write buffer. The electrical schematics and behavior of each of these components is discussed in detail in Section 3.2.

Before performing the simulation of the cell array column, we need to isolate it by simplifying the circuits connected to it along the data path and in the control and decoding paths. Isolation is important for our purposes, since simulating the whole circuit takes an excessive amount of time. Isolation is done by removing electrical components connected to the array column, and replacing them by other simpler components in such a way that the behavior of the array column is affected minimally. In addition to simplification, the column should be initialized at the beginning of each simulation by specifying the initial voltages across all capacitors and the initial state of all latches.

7.1.2 Model reduction techniques

In order to ensure the correctness of the simulations performed on a reduced simulation model, it is important to compensate for the removed circuits, such that the reduced simulation model can reflect the behavior of the full model. A number of memory characteristics, related to its design, structure and operation, make memory simulation models particularly suitable for model reduction, without significant loss in model accuracy [Al-Ars99, Al-Ars00].

- **Memory design**—Memory circuits are commonly divided into three different signal paths (data, address and control paths) that are electrically independent from one another to a large extent. Therefore, it is possible, with minor modifications to the model, to eliminate unneeded signal paths and to simulate only those paths required for analyzing the faulty behavior. For example, it is possible to simulate the behavior of the data path by eliminating the whole address path, and replacing the address decoders with parasitic components in combination with voltage sources to drive the decoded waveform on different WLs.

- **Memory structure**—Memory circuits are constructed in a very modular way, by connecting a large number of identical components in parallel to build large, highly repetitive memories. Therefore, a large number of components can be lumped together into a single representative component, with scaled electrical parameters. For example, the circuit models of a large number of idle memory cells can be lumped together into a single memory cell with electrical characteristics that represent all the idle cells it replaces.

- **Memory operation**—Memory circuits operate primarily based on the principle of precharging and isolated sensing of voltage levels, and not on direct transfer of current from one circuit to the other. This means that memory circuits operate by being first set into a known initial precharge state, and then by isolating them from the rest of the memory, until they stabilize at a specific final state. Therefore, it is possible to correctly simulate the memory components under investigation by eliminating those circuits that are completely isolated from the desired parts of the memory. For example, it is possible to correctly simulate the sensing behavior of a single BL pair, without the need to include a write or a read buffer in the model, since these are always isolated from the BL pair throughout the sensing process.

As shown in Figure 7.1, the simulation model used here is only concerned with the simulation of the data path. Therefore, most of the circuits on the control and the address paths can be simplified or left out. Moreover, circuits on the data path that remain idle throughout the simulation can also be simplified or cut off. Still, the circuits eliminated from the simulation model need to be compensated by parasitic resistances and capacitances to correctly model the load on the simulated parts of the memory. In the following, a number of techniques are given that can be used to simplify the DRAM electrical circuits.

1. **Signals driving transistor gates**—These signals are relatively easy to simplify, since there is no *direct current* (*DC*) transfer to or from the gate of a MOS transistor. The interaction between the gate and other nodes of a transistor is limited to small signal, or *alternating current* (*AC*), effects. Two different simplifications are possible here.

- Any voltage signal, generated by the address or control paths and is driving a transistor gate in the data path, may be replaced by a voltage source generating the same voltage signal. This simplification method has been applied to the data path for signals like those on the word lines and the column select lines.
- Most signals on the address and control paths are digital (i.e., are either high or low), but some signals can assume three states (so-called **tri-state** signals), and in addition to being high or low, they can also be left floating. The high and low voltages of a tri-state signal can be reproduced using a voltage source, while the floating state of the signal can be reproduced using a transistor that disconnects the voltage source whenever the signal is supposed to be floating. This situation is represented in Figure 7.2.

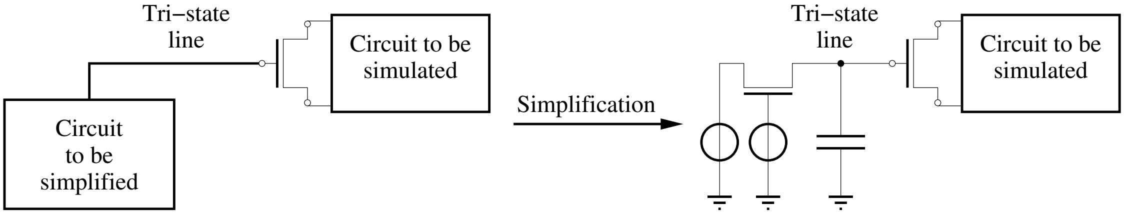


Figure 7.2. Simplification of circuits with tri-state signals driving a transistor gate.

2. **Signals connected to transistor drain/source**—In case a signal line is connected to a drain or source of a transistor, simplification becomes more involved, because DC current can flow from drain to source when the transistor is conducting. One possible situation of this type that applies to the memory cell array is when the signal line to be simplified is kept floating at anytime it is connected to the circuit to be simulated. In this special case, the signal line and the circuit controlling it may be simplified by a transistor, a voltage source and an equivalent capacitor, configured in the way shown in Figure 7.3. For example, this type of simplification can be carried out for the write drivers connected to the bit lines along the data path.

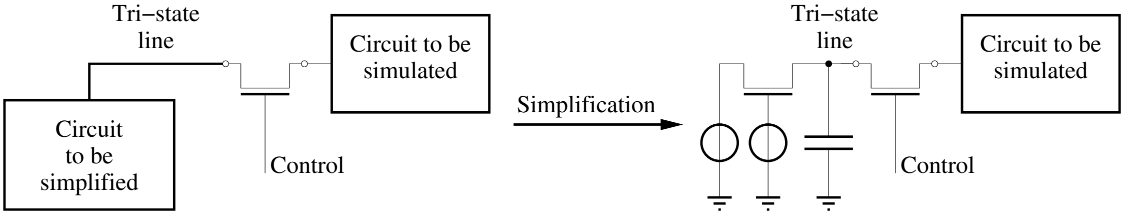


Figure 7.3. Simplification of circuits connected to the drain/source of a transistor.

3. **General simplification methods**—In case none of the methods above work, one of the following two general simplification methods can be applied.
   - It might be possible to trace the signal line backwards, thus including more and more devices into the simulated circuit, in the hope that a cut-set of lines is eventually found that would conform to one of the simplification methods mentioned above.
   - In case back tracing is not possible, then more aggressive simplification techniques are necessary. One such technique is to define a so called *behavioral model* or *simulation macro*, that describe the external behavior of a collection of components rather than the components themselves. This way, the quantity and quality of the models the simulator needs to evaluate is significantly reduced, which results in a shorter simulation time. This simplification technique has not been applied on our circuits.

7.2 Classification of defects

In this thesis, we will mainly be concerned with analyzing defects that take place in the memory cell array, since it takes the largest amount of surface area in the memory and is the part most prone to defects [Al-Ars99, Hamdioui04a]. The defects to be considered are modeled at the electrical level by parasitic resistances. Depending on the signal lines the injected defects are connected to, the defects may be classified into the following three categories [see Figure 7.4]:

- **Open**—Opens represent unwanted resistances on a signal line that is supposed to conduct perfectly.

- **Short**—Shorts are undesired resistive paths between a signal line and power supply (V_{dd} or GND).

- **Bridge**—Bridges are unwanted resistive paths between two signal lines.

It is also possible to model defects as a capacitance coupling between nodes [Al-Ars03a], but these types of defects do not commonly take place in practice [Henderson91], and they usually do not result in a significant impact on the faulty behavior of the memory [Al-Ars99]. The resistive value of opens, shorts and bridges is given by R_{op}, R_{sh} and R_{br}, respectively; each of which may take any value in the range $0 \leq R \leq \infty\ \Omega$. In the following, all memory cell array defects to be analyzed are presented. To this end, this section starts by defining a number of relationships between the injected defects, used to limit the number of simulated defects. This is followed by listing the locations of injected opens, shorts and bridges.

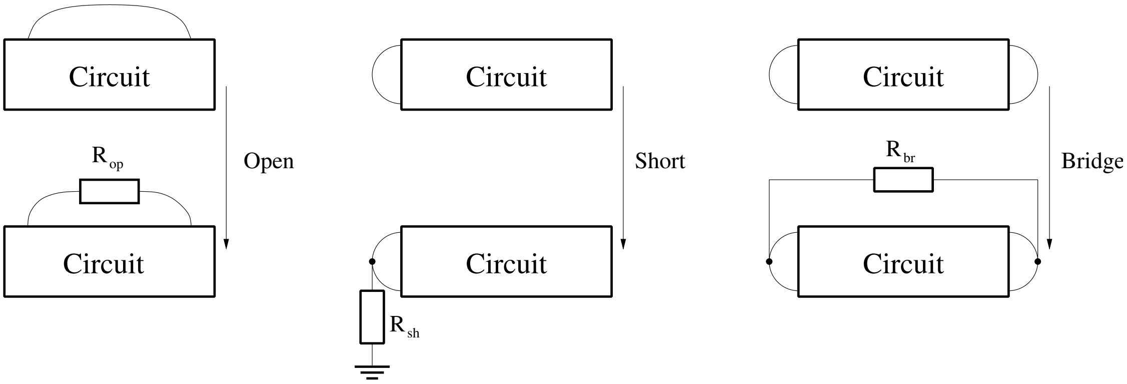


Figure 7.4. Classification of the defects to be simulated and analyzed.

7.2.1 Relations between injected defects

The cell array of the memory is designed with a high degree of symmetry, which results in a corresponding symmetry in the defects injected into the cell array. This symmetry in the injected defects results in a symmetry in the faulty behavior induced by symmetrical defects. Therefore, it is sufficient to analyze the faulty behavior of only one defect in a group of symmetrical defects, and then deduce the faulty behavior of other symmetrically related defects without the need for new simulation and analysis. For this purpose, we provide the following symmetry definitions [Simonse98]:

Table 7.1. Static single-cell fault models and their complementary counterparts.

Fault model			Complement	
SF_0	<0/1/−>	$\Longleftrightarrow$	SF_1	<1/0/−>
IRF_0	<0r0/0/1>	$\Longleftrightarrow$	IRF_1	<1r1/1/0>
$DRDF_0$	<0r0/1/0>	$\Longleftrightarrow$	$DRDF_1$	<1r1/0/1>
RDF_0	<0r0/1/1>	$\Longleftrightarrow$	RDF_1	<1r1/0/0>
WDF_0	<0w0/1/−>	$\Longleftrightarrow$	WDF_1	<1w1/0/−>
TF_0	<1w0/1/−>	$\Longleftrightarrow$	TF_1	<0w1/0/−>

- A defect D1 at a given position shows the **complementary faulty behavior** of a defect D2 at another position, if the faulty behavior of D1 is the same as that of D2, with the only difference that all 1s are replaced by 0s, and vice versa. For example, the fault $<xwy/z/->$ is complementary to the fault $<\overline{x}w\overline{y}/\overline{z}/->$. Table 7.1 lists all static single-cell fault models and their complementary counterparts. The table does not include two-cell faults since it is easy to generalize the table to include them.

- A defect D1 shows the **interchanged faulty behavior** of a defect D2, if the faulty behavior of D1 and D2 contain two-cell faults, and if these two-cell faults are the same with the exception that the aggressor and victim cells are interchanged. In general, if a two-cell fault has the following notation $<S/F/R>_{x,y}$, then the interchanged fault is given by the notation $<S/F/R>_{y,x}$. For example, if a defect D1 causes the two-cell fault $<1; 0w0/1/->_{x,y}$, then the interchanged defect D2 causes the same fault $<1; 0w0/1/->_{y,x}$, but with aggressor and victim interchanged.

- A defect D1 shows the **single-sided complementary behavior** of a defect D2, if the faulty behavior of D1 and D2 contain two-cell faults, and if these two-cell faults are the same with the exception that all 1s are replaced by 0s, and vice versa, in either the aggressor or the victim cells (not both). If the victim sides of two faults are the complement of each other, then these two faults are called *victim-sided complementary*. If the aggressor sides of two faults are the complement of each other, then these two faults are called *aggressor-sided complementary*. For example, suppose that D1, D2 and D3 affect cells x and y, and that D1 forces a $0w0$ operation to cause an up transition in y if cell x is in state 1, then this faulty behavior of D1 is denoted by $<1; 0w0/1/->_{x,y}$. The aggressor-sided complementary defect D2 should force a $0w0$ operation to cause an up transition in y if cell x is in state 0, which is the fault denoted by $<0; 0w0/1/->_{x,y}$. On the other hand, The victim-sided complementary defect D3 should force a $1w1$ operation to cause a down transition in y if cell x is in state 1, which is the fault denoted by $<1; 1w1/0/->_{x,y}$.

In Section 7.2.2, defect names are given attributes to point out the symmetrical relationship a given defect shares with others, based on the symmetry definitions given above (complementary, interchanged and single-sided complementary). Table 7.2 lists all symmetry attributes and describes their meaning. As a result of these relationships, it is sufficient to simulate the behavior of only one representative of a group of symmetrically related defects. This one defect we choose to simulate is called the *simulated defect* and is given the attribute “s”.

In some situations, defects can be classified according to more than one of the symmetrical relationships given in Table 7.2. In this case, the defect group contains more than one letter selected from the table. For example, bridges between two cells, which are classified as interchanged and as complementary at the same time, are given the attributes “i” for interchanged and “c” for complementary, resulting in the combined attribute “ic”.

7.2.2 Definition of defects

Here, all targeted defects in the memory cell array are explicitly defined, and every instance of these defects is given a label to refer to it later in the thesis. Figure 7.5

Table 7.2. Symmetry attributes given to defect names and their meaning.

Attribute	Meaning
s	Defect to be simulated and analyzed
c	Defect with the complementary behavior to the simulated one
i	Defect with the interchanged behavior to the simulated one
a	Defect with an aggressor-sided complementary behavior
v	Defect with a victim-sided complementary behavior

shows an overview of the locations of defects to be simulated. The defects are classified into opens, shorts and bridges. The opens and shorts are divided into defects in memory cells, on BLs or on WLs, while bridges are divided into defects involving no cells, one cell and two different cells. In the following, we start with defining the locations of opens, then shorts and finally bridges.

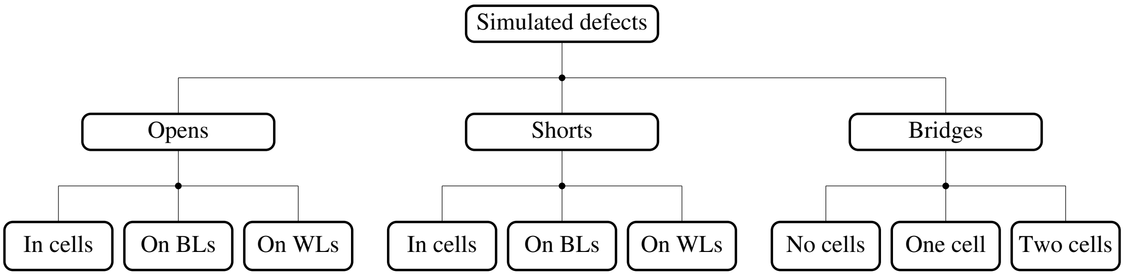
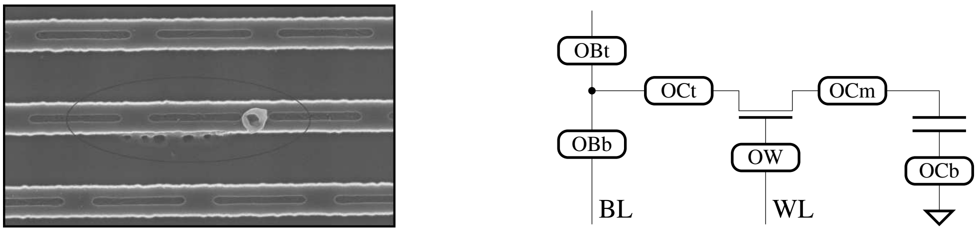


Figure 7.5. Overview of simulated defect locations.

Location of opens

At the layout level, opens are usually caused by broken lines, or particle contamination that results in increasing line resistivity at the open position. Figure 7.6(a) shows a layout example of a BL open caused by particle contamination, resulting in an increase in the BL resistance and inducing some kind of faulty behavior in the memory.

Opens in the memory cell array can be either opens within cells (OC), opens on BLs (OB) or opens on WLs (OW). Figure 7.6(b) shows these three different locations of opens. There are three types of OCs, on top (t), in the middle (m) and at the bottom (b), all of which (partially) disconnect the cell from the BL and limit the ability of the memory to control and observe the voltage within the cell. There are two types of OBs, disconnecting the cell from circuitry on the top of the BL (t), and disconnecting the cell from circuitry at the bottom of a the BL (b).



(a) BL open (b) Location of opens

Figure 7.6. (a) Layout level example of a BL open. (b) Positions of opens in the cell array.

There is only one word line open called OW, which limits the ability of the memory to properly access the cell.

BL in the figure can either be the true bit line (BT), or the complementary bit line (BC). The faulty behavior of defects related to BT is complementary to the faulty behavior of defects related to BC, which means that it is enough to simulate the faulty behavior on BT. Table 7.3 summarizes the possible open defects, where the (s) and (c) subscripts in the name of the defect refer to the simulated defect and the one with the complementary behavior, respectively. The table also indicates whether a 1D or 2D analysis is needed to evaluate the faulty behavior of the memory with the presence of the defect.

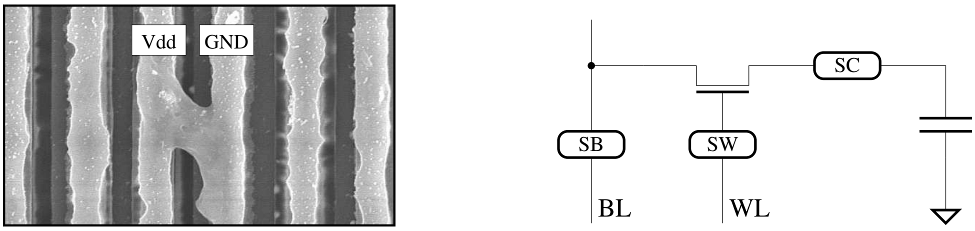
Table 7.3. Analyzed open defects in the memory cell array.

#	BT open	BC open	Analysis	Description
1	OCt_s	OCt_c	1D	Pass transistor connection to the bit line broken
2	OCm_s	OCm_c	1D	Pass transistor connection to the capacitor broken
3	OCb_s	OCb_c	1D	Cell connection to the capacitor plate broken
4	OBt_s	OBt_c	2D	Bit line disconnected from the cell at the top
5	OBb_s	OBb_c	2D	Bit line disconnected from the cell at the bottom
6	OW_s	OW_c	1D	Word line disconnected from memory cell

Location of shorts

At the layout level, shorts can be caused by a number of physical failures such as extra metal deposition or isolation breakdown, that result in faulty connections between power supply lines and other signal lines in the memory. Figure 7.7(a) shows a layout example of a power supply short caused by extra metal deposition, resulting in a faulty new connection being formed between two otherwise disconnected lines.

Shorts, similar to opens, can be either within cells (SC), on bit lines (SB), or on word lines (SW). Figure 7.7(b) shows the short positions of these three different



(a) Power line shorts (b) Location of shorts

Figure 7.7. (a) Physical example of power supply shorts. (b) Positions of shorts in the cell array.

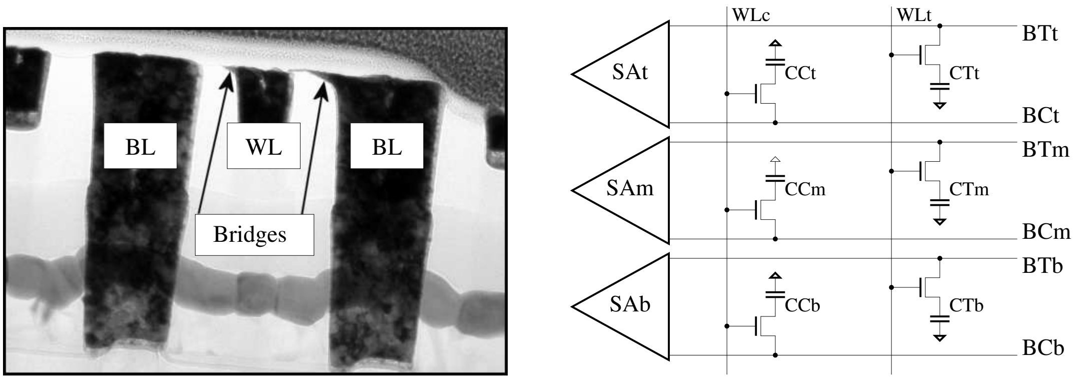
types of shorts. At each position indicated in the figure, a short may connect the node either to V_{dd} or GND. Shorts to V_{dd} are indicated by the letter (v) as in SCv and SBv, while shorts to GND are indicated by the letter (g) as in SCg and SBg. BL in the figure can either be the true bit line (BT), or the complementary bit line (BC). The faulty behavior of shorts related to BT is complementary to the faulty behavior of defects related to BC, which means that it is enough to analyze the fault behavior on either BT or BC. Table 7.4 summarizes the possible short defects, where the (s) subscript in the name stands for a short that is simulated and analyzed, while (c) in the name of the defect refers to a short with the complementary behavior.

Table 7.4. Analyzed shorts in the memory cell array.

#	BT short	BC short	Analysis	Description
1	SCv_s	SCv_c	1D	Cell storage capacitor shorted to V_{dd}
2	SCg_s	SCg_c	1D	Cell storage capacitor shorted to GND
3	SBv_s	SBv_c	2D	Bit line connected to victim cell shorted to V_{dd}
4	SBg_s	SBg_c	2D	Bit line connected to victim cell shorted to GND
5	SWv_s	SWv_c	1D	Word line connected to victim cell shorted to V_{dd}
6	SWg_s	SWg_c	1D	Word line connected to victim shorted to GND

Location of bridges

At the layout level, bridges can be the result of extra metal deposition or isolation layer misalignment, that cause faulty connections between different lines in the memory. Figure 7.8(a) shows a layout example of a bridge between the WL and the BL contacts caused by a misalignment in the isolation layer. The small black structure in the middle of the figure represents the WL contact, which is sandwiched between two relatively large BL contacts. The WL and BL contacts are separated from each other by a thin isolation layer, that is difficult to manufacture. The figure shows that the isolation layer is not created properly, leaving a small space on top through which the BL contact stretches and connects to the WL.



(a) BL-WL bridge

(b) Nodes for bridges

Figure 7.8. (a) Physical example of a BL-WL bridge. (b) Nodes where bridges can take place.

At the electrical level, bridges are resistive connections between two nodes in the memory. In order to take all cell array bridges into consideration, we need to consider bridges between any two nodes in the cell array. Since the cell array has a repetitive structure, it is possible to consider only the region surrounding a single cell in the array as a representative of the whole. Such a representative part is shown in Figure 7.8(b), where all nodes are given names for later reference.

The figure shows three BL pairs, at the top (BTt and BCt), in the middle (BTm and BCm), and at the bottom (BTb and BCb). Each BL pair has two memory cells connected to it, giving a total of six cells, three on BT (with nodes CTt, CTm and CTb) and three on BC (with nodes CCt, CCm and CCb). The cells connected to BT are controlled by the true WL (WLt), while the cells connected to BC are controlled by the complement WL (WLc).

Therefore, the figure identifies 6 BLs (BTt, BCt, BTm, BCm, BTb and BCb), 6 cell nodes (CTt, CCt, CTm, CCm, CTb and CCb), and 2 WLs (WLt and WLc), resulting in a total of 14 different nodes. This in turn means that there is a total of $14 \times 13/2 = 91$ bridges theoretically possible. In reality, however, bridges cannot take place between any two arbitrary nodes in the memory, since only nodes in close proximity of one another can be bridged. Furthermore, many of these bridges result in exactly the same faulty behavior, such as bridges between a given cell to any idle (not accessed) cell. In the following, we choose a total of 24 different bridges to analyze, that give a good representation of the behavior of all 91 possible bridges.

We can classify bridges into those involving zero cells (BZ), one cell (BO) and two cells (BW). BO defects are the easiest to discuss, and therefore we list them first in Table 7.5. For these defects, the defective cell is the one accessed and inspected for possible faulty behavior, since we assume that the faulty behavior would appear there most prominently. The table lists 6 bridges identified as simulated (s), and 6 with a complementary faulty behavior (c).

Table 7.6 lists the bridges that involve two memory cells. During simulation, the

Table 7.5. Possible bridge defects involving one cell (BO) in the memory cell array.

#	Sim. BO		Comp. BO		Ana.	Description
1	BO1$_s$	CTm-WLt	BO1$_c$	CCm-WLc	1D	Accessed cell bridged to own WL
2	BO2$_s$	CTm-WLc	BO2$_c$	CCm-WLt	1D	Accessed cell bridged to another idle WL
3	BO3$_s$	CTm-BTm	BO3$_c$	CCm-BCm	2D	Accessed cell bridged to own BL
4	BO4$_s$	CTm-BCm	BO4$_c$	CCm-BTm	2D	Accessed cell bridged to own comp. BL
5	BO5$_s$	CTm-BCt	BO5$_c$	CCm-BTb	2D	Accessed cell bridged to another comp. BL

cell containing the node CTm is accessed and inspected for faulty behavior. The table lists two defects in the first column, identified as simulated bridges (s), followed by their complementary (c) counterparts in the second column. For example, the bridge BW1$_s$ connects the cell on BTm to the cell on BCm. This complementary couterpart of this defect is identical to the defect itself and therefore need not be mentioned.

Table 7.6. Possible bridges between two memory cells (BW) in the memory cell array.

#	Sim. BW		Comp. BW		Analysis	Description
1	BW1$_s$	CTm-CCm	—	—	2D	Accessed cell bridged to idle cell
2	BW2$_s$	CTm-CTb	BW2$_c$	CCm-CCb	2D	Accessed cell bridged to active cell

Table 7.7 lists the bridges that do not involve memory cells (bridges with zero cells or BZ), but connect nodes outside memory cells. The CTm cell is the one accessed and inspected for faulty behavior for the bridges listed in the simulated BZ column, while the CCm cell is the one inspected for bridges in the complementary BZ columns. Note that the bridge BZ6$_s$, there is no complementary bridge entry in the table, since it is identical to the simulated bridge itself.

Table 7.7. Possible bridges not involving memory cells (BZ) in the memory cell array.

#	Sim. BZ		Comp. BZ		Ana.	Description
1	BZ1$_s$	BTm-BCm	BZ1$_c$	BCm-BTm	1D	Accessed true bit line to accessed complementary bit line
2	BZ2$_s$	BTm-BCt	BZ2$_c$	BCm-BTb	2D	Accessed true bit line to another complementary bit line
3	BZ3$_s$	BCm-BTb	BZ3$_c$	BTm-BCt	2D	Accessed complementary bit line to another true bit line
4	BZ4$_s$	BTm-WLt	BZ4$_c$	BCm-WLc	2D	Accessed true bit line to accessed word line
5	BZ5$_s$	BCm-WLt	BZ5$_c$	BTm-WLc	2D	Accessed complementary bit line to accessed word line
6	BZ6$_s$	WLt-WLc	—	—	2D	Accessed word line to idle word line

Summary of analyzed defects

In total, there are 48 bridge defects that cover most realistic defective connections that may take place in the memory cell array. Only 25 of those need be simulated, while the behavior of the others can be subsequently derived according to the symmetrical equivalence rules described in Section 7.2.1.

Table 7.8 gives a summary of all the defects discussed above, the faulty behavior of which is simulated and analyzed in detail in the coming sections. The table first lists the simulated defects and their complementary counterparts, divided into opens, shorts and bridges. Then, the type of analysis needed to simulate the faulty behavior of each defect is listed. The 1D entry refers to the fact that the corresponding defect causes one floating node and requires the application of the one dimensional analysis, while the 2D entry means that the corresponding defect causes two floating nodes and requires the application of the two dimensional analysis. Finally, the table gives a short description of the defect.

7.3 Application of 1D analysis

In this section, we apply the 1D approximate simulation method to analyze the faulty behavior of those cell array defects that cause a single floating node, and therefore require the 1D analysis [Al-Ars02b, Al-Ars02c]. The used simulation model is shown in Figure 7.1. It includes 3 BL pairs coupled with each other using coupling capacitances, which makes it possible to simulate BL coupling with different **background patterns** (**BGs**). The different BGs and their relationship to the simulation model is discussed in Section 7.1.1. More detailed discussion of BGs and the impact of BL coupling on the faulty behavior can be found in the literature [Al-Ars04b, Al-Ars04c]. Assuming that cell CTm in Figure 7.1 is the one under investigation, then the BG for this cell is chosen to be the combination of the data stored in cell CTt and cell CTb. Such a BG is represented as tmb, where t stands for the data in CTt, while b represents the data in CTb. The value m stands for the initial data stored in CTm, which is often represented by a don't care (x), since the initialization of the cell is mostly done by an operation that is part of the fault. To analyze the impact of BGs, all simulations are repeated for the following four different BGs $0m0$, $0m1$, $1m0$ and $1m1$.

The 1D fault analysis approach (discussed in Section 5.2) is performed for all 1D defects, and the results of this analysis are presented in Table 7.9. In case multiple operations are needed to sensitize a fault (partial faults) the operations are denoted as wx^h or rx^h, where h stands for the number of repetitions of the operation. The results in the table have been derived according to the following criteria.

- If the simulation shows that the defect causes the cell to loose its charge and change its logic value very fast (within the time span of a single memory operation), then the faulty behavior is modeled using a state fault ($<x/\bar{x}/->$).

Table 7.8. All analyzed defects in the memory cell array.

Sim. def.	Comp. def.	Analysis	Description
OCt_s	OCt_c	1D	Pass transistor connection to the bit line broken
OCm_s	OCm_c	1D	Pass transistor connection to the capacitor broken
OCb_s	OCb_c	1D	Cell connection to the capacitor plate broken
OBt_s	OBt_c	2D	Bit line disconnected from the cell at the top
OBb_s	OBb_c	2D	Bit line disconnected from the cell at the bottom
OW_s	OW_c	1D	Word line disconnected from memory cell
SCv_s	SCv_c	1D	Cell storage capacitor shorted to V_{dd}
SCg_s	SCg_c	1D	Cell storage capacitor shorted to GND
SBv_s	SBv_c	2D	Bit line connected to victim cell shorted to V_{dd}
SBg_s	SBg_c	2D	Bit line connected to victim cell shorted to GND
SWv_s	SWv_c	1D	Word line connected to victim cell shorted to V_{dd}
SWg_s	SWg_c	1D	Word line connected to victim shorted to GND
BO1_s	BO1_c	1D	Accessed cell bridged to own WL
BO2_s	BO2_c	1D	Accessed cell bridged to another idle WL
BO3_s	BO3_c	2D	Accessed cell bridged to own BL
BO4_s	BO4_c	2D	Accessed cell bridged to own comp. BL
BO5_s	BO5_c	2D	Accessed cell bridged to another comp. BL
BW1_s	—	2D	Accessed cell bridged to idle cell
BW2_s	BW2_c	2D	Accessed cell bridged to active cell
BZ1_s	BZ1_c	1D	Accessed true bit line to accessed complementary bit line
BZ2_s	BZ2_c	2D	Accessed true bit line to another complementary bit line
BZ3_s	BZ3_c	2D	Accessed complementary bit line to another true bit line
BZ4_s	BZ4_c	2D	Accessed true bit line to accessed word line
BZ5_s	BZ5_c	2D	Accessed complementary bit line to accessed word line
BZ6_s	—	2D	Accessed word line to idle word line

- If the simulation shows that the defect causes write operations to store a weaker voltage than what is needed by a read operation to detect a correct value in the cell, then the faulty behavior is modeled using a faulty write operation ($<wx/\bar{x}/->$).

- If the simulation shows that the defect causes multiple write operations to have a cumulative voltage effect in the cell capacitance (i.e., multiple write operations cause a gradual increase or decrease in cell voltage), then the faulty behavior is modeled as a partial fault $<wx^h\ w\bar{x}/x/->$ [see Section 4.3.1].

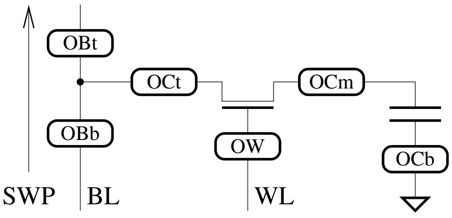
The first column in Table 7.9 lists the name of the simulated defect, followed by the used BG in the second column. The third column uses the FP notation to describe the resulting faulty behavior. The following 12 columns indicate the type of the DRAM-specific fault being modeled by the FP. There are five basic types of faults: hard (h), soft (s), transient (t), partial (p) and dirty (d). Together,

Table 7.9. Simulation results of the 1D fault analysis, where $x \in \{0, 1\}$

Defect	BG	FP	h	ph	dh	pdh	s	ps	ds	pds	t	pt	dt	pdt
OCt$_s$, OCm$_s$ and OCb$_s$	$0x0$	$<w1^3w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x0$	$<w1^3w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$0x1$	$<w0^3w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x1$	$<w0^3w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
OW$_s$	$0x0$	$<w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$1x0$	$<w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$0x1$	$<w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$1x1$	$<w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
SCv$_s$	000	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
	100	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
	001	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
	100	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
SCg$_s$	010	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
	110	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
	011	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
	111	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
SWv$_s$	000	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
	100	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
	001	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
	101	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
SWg$_s$	$0x0$	$<w1^3w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x0$	$<w1^3w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$0x1$	$<w1^3w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x1$	$<w1^3w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
BO1$_s$	010	$<w1^2w0/1/->$	−	+	−	−	−	−	−	−	−	+	−	−
	110	$<w1^2w0/1/->$	−	+	−	−	−	−	−	−	−	+	−	−
	011	$<w1^2w0/1/->$	−	+	−	−	−	−	−	−	−	+	−	−
	111	$<w1^2w0/1/->$	−	+	−	−	−	−	−	−	−	+	−	−
BO2$_s$	010	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
	110	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
	011	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
	111	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
BZ1$_s$	$0x0$	$<w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$1x0$	$<w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$0x1$	$<w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$1x1$	$<w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−

they combine to make up 12 possible fault combinations [see Section 4.3.3]. In the following, the faulty behavior of each defect is discussed in more detail. The description of some defects contain little figures on the right-hand side, to indicate the different positions of the defects. These figures also contain arrows along the BL, that carry the tag “SWP”, which stands for sense amplifier, write driver and precharge circuits. These arrows point toward the direction along which these memory components are connected to the bit line.

Opens at OCt, OCm and OCb

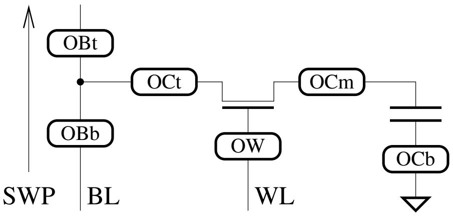


The positions of these opens are indicated in the figure to the right. The arrow with the tag “SWP” points to the direction along which the BL gets connected to the sense amplifier, write driver and precharge circuits. The opens OCt, OCm and OCb take place within the memory cell and limit the ability of the memory to charge or discharge the cell during write operations. As a result, multiple write operations cause a cumulative charge up or discharge of the cell, which means that in order to ensure proper initialization, a sequence of write operations needs to be performed. During read operations, these defects limit the ability of the cell to put a high enough voltage margin on the bit lines, needed to ensure proper sensing of stored data. The low voltage margin on BLs during sensing makes the faulty behavior of these defects very sensitive to voltage disturbances, such as those from background data in neighboring cells, which have a large influence on the read output. According to Table 7.9, the faulty behavior is represented by the FP $<wx^3\ w\bar{x}/x/->$, which requires a sequence of three initializing wx operations, followed by a sensitizing $w\bar{x}$ operation. The three initializing operations are needed to bring the voltage in the cell to a strong enough level, since the open makes it difficult for a single operation to set a full voltage in the cell. The sensitizing $w\bar{x}$ operation fails to properly change the voltage in the cell, leaving a faulty value of x behind.

Table 7.9 indicates that the FP of these defects depends on the BG, which means that the BG plays a significant role in the faulty behavior. This is true since the small voltage margin during the read operation can be easily affected by BL coupling from neighboring cells. The four simulated BGs are $0x0$, $1x0$, $0x1$ and $1x1$, where the initial value of the faulty cell is taken to be x (don’t care) since the initialization of the faulty cell makes part of the FP. The table shows that for BG = $0x0$ and $1x0$, the faulty behavior is represented by $<w1^3\ w0/1/->$, while for BG = $0x1$ and $1x1$, the faulty behavior is represented by $<w0^3\ w1/0/->$.

Since these opens result in FPs that require an initialization consisting of 3 operations, the faults are attributed as partial. They are not dirty, though, since they do not require completing operations to be sensitized. At the same time, these opens do not force leakage current into the cell toward a specific direction, and therefore leakage can cause hard, soft, as well as transient faults.

Open at OW

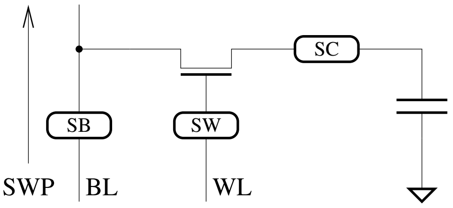


This open takes place along the WL of the faulty cell, which limits the ability of the memory to control the broken WL part connected to the cell. For very high open resistances, the broken WL part remains floating at a given voltage which can leave the cell either constantly connected to the bit line, or constantly disconnected from it. In both cases, the voltage in the cell has no impact on the read operation, and therefore the read output is controlled by the background data (through BL coupling) rather than the data in the faulty cell itself. This way, if the BG causes the read output to be x, then we need to write $\bar{x}$ into the cell in order to sensitize the fault. This faulty behavior can be represented by the FP $<wx/\bar{x}/->$.

The BG plays a significant role in the faulty behavior of this defect, since the small voltage margin during the read operation can easily be influenced by BL coupling from neighboring cells. The table shows that for BG $= 0x0$ and $1x0$, the faulty behavior is represented by $<w0/1/->$, while for BG $= 0x1$ and $1x1$, the faulty behavior is represented by $<w1/0/->$.

Since this open does not force leakage currents into the cell toward a specific direction, leakage currents can lead to hard faults, soft faults as well as transient faults. However, this open does not require multiple initializing operations, nor completing operations, which means that it is neither partial nor dirty.

Shorts SCv and SCg



The positions to which these shorts are connected are indicated in the figure to the right. The arrow with the tag "SWP" points to the direction along which the BL gets connected to the sense amplifier, write driver and precharge circuits. The shorts SCv and SCg take place within the memory cell and connect the cell capacitor node either to V_{dd} or to GND. For very low short resistances, the cell voltage gets pulled to V_{dd} or GND very fast and causes a fault. The resulting faulty behavior can be represented in the form of a state fault ($<x/\bar{x}/->$), which means that a stored value in the cell would be spontaneously modified from x to $\bar{x}$ without performing any operations on the defective cell.

The BG has no major impact on the faulty behavior of these defects, since the voltage in the cell is mainly controlled by the short defect, resulting in the same type of FP for all BGs. The short to V_{dd} always cases the state fault $<0/1/->$, while the short to GND always causes the state fault $<1/0/->$.

Since the shorts result in a leakage current that pulls the cell voltage in the same direction as the resulting fault, the faulty behavior is attributed as a soft fault, and can therefore be represented as $<x_T/\bar{x}/->$, where T stands for a time delay needed for the fault to get sensitized. When the short has a very low resistance, the leakage becomes very high and T becomes very short. When T becomes smaller than the cycle time of a single memory operation, the fault is modeled as a hard fault.

Short SWv

This short connects the WL to V_{dd} and forces the cell to be connected to the BL at all time, resulting in a leakage like behavior to the precharge voltage of the BL ($V_{dd}/2$). Furthermore, as a result of internal biases in the sense amplifier toward sensing a 1, a voltage of $V_{dd}/2$ in the cell gets detected as a 1 by a read operation. The resulting faulty behavior is that of a typical data retention fault represented by $<0_T/1/->$, which means that a stored value 0 gradually depletes toward a faulty 1 as a result of leakage.

The BG has a limited impact on the faulty behavior, since shorting the WL influences *all* cells along the same row, which depletes the voltages in the cells in the background as well. Therefore, as R_{sh} decreases, the effect of the BG decreases as well, thereby keeping it secondary to other effects in memory operation.

Since the short results in a leakage current that pulls the cell voltage in the same direction as the resulting fault, the faulty behavior is attributed as a soft fault, and can therefore be represented as $<0_T/1/->$, where T stands for a time delay needed for the fault to become sensitized. When the short has a very low resistance, the leakage becomes very high and the fault is modeled as a hard fault.

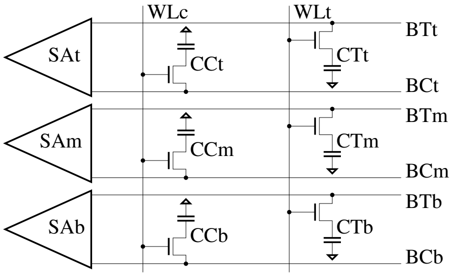
Short SWg

This short connects the WL of the faulty cell to GND, which restricts access to the cell and reduces the ability of the memory to write and read proper memory cell voltages. This type of faulty behavior is very similar to that exhibited by opens within the cell, which also restrict access to the cell. Just like the behavior of cell open, this defect caused multiple write operations to have a cumulative effect on the cell voltage. Therefore, in order to initialize the faulty behavior, multiple wx operations should be performed. This initialization should be followed by the operation $w\bar{x}$ to sensitize the fault in the cell. In conclusion, this faulty behavior can represented by the fault $<w1^3\ w0/1/->$, which starts with three initializing $w1$ operations, followed by a sensitizing $w0$ operation that fails.

Unlike the behavior of cell opens, the faults caused by this defect are independent from background data. The reason for this is that a WL shorted to GND impacts *all* cells in the row connected to the same defective WL, which means that not only the victim cell but also the background cells cannot be accessed. Therefore, as R_{sh} decreases, the effect of the BG decreases as well, thereby keeping it secondary to other effects in memory operation.

Since this short does not force leakage current into the cell toward a specific direction, leakage currents can lead to hard faults, soft faults as well as transient faults in the cell. At the same time, the sequence of three initializing operations indicate that the faulty behavior can also be attributed as partial.

Bridge BO1 between CTm (CCm) and WLt (WLc)



The nodes to which this bridge is connected are indicated in the figure to the right. The defect BO1 is a bridge that connects the cell storage capacitance to its WL, thereby forcing the stored cell voltage to follow the voltage present on the WL during memory operation. With a very low bridge resistance, the voltage in the cell strongly follows the voltage on the WL, which means that once the cell is accessed by pulling the WL high, the cell voltage is pulled high as well. As a result, any write 0 operation fails, while any read operation results in a 1 on the output. As the bridge resistance increases, the cell voltage becomes slower in tracking the WL voltage, which causes multiple operations to have a cumulative effect on the cell voltage. In conclusion, the faulty behavior can be represented by the fault $<w1^2\ w0/1/->$, which starts with two initializing $w1$ operations, followed by a sensitizing $w0$ that fails, leaving a faulty 1 behind in the cell.

The faulty behavior of this defect is not affected by the BG, since the bridge to the strong voltage of the WL is much more influential than coupling voltages from the cells in the background. Therefore all BGs result in exactly the same type of faulty behavior in the memory.

Since the idle voltage of the WL is GND, which is opposite to the active voltage of the WL, the bridge results in a leakage current that pulls the cell voltage in the opposite direction to the resulting fault. Therefore, the faulty behavior is attributed as a transient fault, and can therefore be represented as $<w1\ \underline{w1}\ \underline{w0}/1_L/->$, where the underlined operation should follow the previous one instantaneously to prevent the elimination of the fault as a result of leakage. At the same time, since multiple operations are needed to initialize the fault, it can be attributed as a partial fault.

Bridge BO2 between CTm (CCm) and WLc (WLt)

This defect is a bridge between the cell storage capacitance and a WL connected to a cell on BC, a WL that is idle all the time and carries the voltage of GND. The faulty behavior resulting from this bridge is identical to the faulty behavior resulting from the short SCg which connects the cell to GND. These two defects share the same FP, the same impact of BG on the faulty behavior, and the same DRAM-specific attributes associated to the fault.

Bridge BZ1 between BTm and BCm

This defect forms a bridge between the true bit line and the complement bit line that results in pulling the voltages on BT and BC closer together. For very low bridge resistances, the voltages on both BT and BC are pulled very close together which makes it difficult for the sense amplifier to detect the correct voltage in the cell. The read output in this case depends on both the stored voltage in the cell and on the BG. As a result, when the read output is always forced to be x, we need to perform a $w\bar{x}$ operation on the cell in order to sensitize the fault. This faulty behavior can be represented by the FP $<wx/\bar{x}/->$.

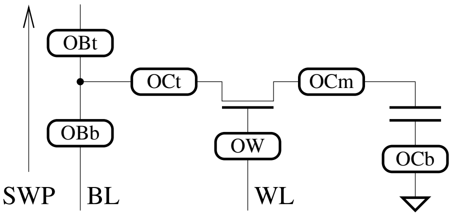
The faulty behavior is influenced by BG in such a way, that $0x0$ causes the FP $<w0/1/->$ while the other BGs cause the FP $<w1/0/->$. Since this bridge does not force leakage current into the cell toward a specific direction, leakage currents can lead to hard faults, soft faults and transient faults. However, this bridge does not require multiple initializing operations, nor completing operations, which means that it is neither partial nor dirty.

7.4 Application of 2D analysis

This section presents the 2D analysis results [see Section 5.3] of a simulation-based study of the defects listed in Table 7.8 that require the 2D analysis method [Al-Ars03d]. In the same way outlined in Section 7.3, the analysis takes BL coupling into consideration, and simulates the faulty behavior with different BGs. All simulations are repeated for the four different BGs $0x0$, $0x1$, $1x0$ and $1x1$. A summary of the results of this analysis is presented in Table 7.10. In case multiple operations are needed to sensitize a fault (partial faults) the operations are denoted as wx^h or rx^h, where h stands for the number of repetitions of the operation. The criteria used to derive the results in the table are the same as those used to derive the results in Table 7.9, as discussed in Section 7.3.

The first column in the table lists the name of the simulated defect, followed by the used BG in the second column. The third column uses the FP notation to describe the resulting faulty behavior. The following 12 columns indicate the type of the DRAM-specific fault being modeled by the FP. There are five basic types of faults: hard (h), soft (s), transient (t), partial (p) and dirty (d). Together, they combine to make up 12 possible fault combinations [see Section 4.3.3].

Open at OBt



The position of this open is indicated in the figure to the right. The arrow with the tag “SWP” points to the direction along which the BL gets connected to the sense amplifier, write driver and precharge circuits. This open takes place on the BL between the precharge, write and read

Table 7.10. Simulation results of the 2D fault analysis, where $t, x, b \in \{0, 1\}$

Defect	BG	FP	h	ph	dh	pdh	s	ps	ds	pds	t	pt	dt	pdt
OBt$_s$	$0x0$	$<w0^2w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x0$	$<w0^2w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$0x1$	$<w0^2w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x1$	$<w0^2w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
OBb$_s$	txb	No fail	−	−	−	−	−	−	−	−	−	−	−	−
SBv$_s$	$0x0$	$<w1^2w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x0$	$<w1^2w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$0x1$	$<w1^2w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1x1$	$<w1^2w0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
SBg$_s$	000	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	100	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	001	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	101	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
BO3$_s$,	000	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
BO4$_s$	100	$<0/1/->$	+	−	−	−	+	−	−	−	−	−	−	−
and	011	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
BO5$_s$	111	$<1/0/->$	+	−	−	−	+	−	−	−	−	−	−	−
BW1$_s$	000	$<w1^2;0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	100	$<w1^2;0/1/->$	−	+	−	−	−	+	−	−	−	+	−	−
	011	$<w0^2;1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
	111	$<w0^2;1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
BW2$_s$	$0xb$	$<w0^2w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
	$1xb$	$<w0^2w1/0/->$	−	+	−	−	−	+	−	−	−	+	−	−
BZ2$_s$	$t00$	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$t01$	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
BZ3$_s$	$00b$	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	$10b$	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
BZ4$_s$	010	$<1w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
	110	$<1w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
	011	$<1w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
	111	$<1w0/1/->$	+	−	−	−	+	−	−	−	+	−	−	−
BZ5$_s$	000	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	100	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	001	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	101	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
BZ6$_s$	000	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	100	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	001	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−
	101	$<0w1/0/->$	+	−	−	−	+	−	−	−	+	−	−	−

circuits and memory cell, which limits the ability of the memory to write a specific voltage into the cell and the ability to read stored data from the cell. The open requires a 2D analysis since it results in two floating nodes: 1. the faulty cell under analysis, and 2. the part of the BL that is disconnected from the sense amplifier and write driver.

The results of the 2D fault analysis show that multiple write operations cause a cumulative charge up or discharge of the cell, which means that in order to ensure proper initialization, a sequence of write operations needs to be performed. Therefore, the faulty behavior is represented by the FP $<wx^2\ w\bar{x}/x/->$, which requires a sequence of two initializing wx operations, followed by a sensitizing $w\bar{x}$ operation. The two initializing operations are needed to bring the voltage in the cell to a strong enough level, since the open makes it difficult for a single operation to set a full voltage in the cell. The sensitizing $w\bar{x}$ operation fails to properly change the voltage in the cell, leaving a faulty value of x behind. Note that as the value of R_{op} increases, the faulty behavior becomes easier to sensitize and detect, since the read operation becomes more difficult to perform. This is due to the fact that the open prevents setting up a high enough voltage across the sense amplifier to perform a correct read operation. As a result, an increase in the open resistance results in an easily detectable faulty behavior, which means that the fault requires less initializing wx operations in the FP.

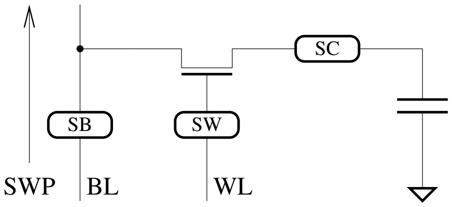
The BG plays a very limited role in the faulty behavior of this defect, since the BL open reduces the length of the BL segment connected to the sense amplifier, thereby reducing the amount of BL coupling affecting the behavior of the memory. All four simulated BGs ($0x0$, $1x0$, $0x1$ and $1x1$) result in the same faulty behavior, $<w0^2\ w1/0/->$.

Since this open results in FPs that require an initialization consisting of 2 operations, the faults are attributed as partial. At the same time, this open does not force leakage current into the cell toward any specific direction, which means that all leakage related faults (hard, soft and transient) are possible.

Open at OBb

This open takes place on the BL, disconnecting the memory cell from a part of the BL and all other cells connected to it. According to the specific design of the simulated memory, such an open does not disconnect the memory cell from any critical write and read circuits, but merely reduces the capacitive influence the other cells have on the inspected cell. This open requires a 2D analysis since it results in two floating nodes: 1. the node of the inspected cell, and 2. the part of BL that is disconnected from the sense amplifier and write driver. The fault analysis results show that there is no faulty behavior associated with this defect, because the cell remains connected to the rest of memory components (sense amplifier, write driver and precharge circuits). It should be noted, however, that other memory designs may indeed result in faults in the inspected cell, in case the sense amplifier, write drivers of precharge circuits become disconnected from the cell by the open.

Short at SBv



The position to which this short is connected is indicated in the figure to the right. The arrow with the tag "SWP" points to the direction along which the BL gets connected to the sense amplifier, write driver and precharge circuits. This short takes place on the BL and connects it to the power supply node V_{dd}, which forces the BL to be charged up toward V_{dd} resulting in a bias against writing and reading a correct 0 in the inspected cell. The two floating voltages resulting from this defect are the stored cell voltage and the voltage on the defective BL. The resulting faulty behavior can be represented as $<wx^2\ w\bar{x}/x/->$, which requires a sequence of two initializing wx operations needed to bring the voltage in the cell to a strong enough voltage, followed by a sensitizing $w\bar{x}$ operation that fails to properly change the voltage in the cell, leaving a faulty value of x behind.

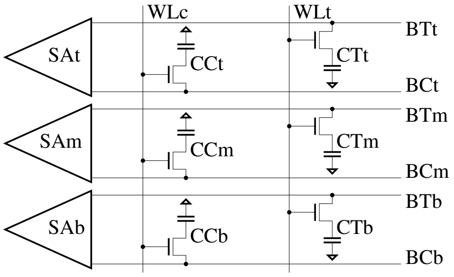
The BG has no major impact on the faulty behavior of this defect, since the voltage on the BL is mainly influenced by the short itself and not by the sense voltage during reading, which in turn causes the same type of FP for all BGs. Since this defect results in FPs that require an initialization consisting of 2 operations, the faults are attributed as partial. At the same time, this defect does not force leakage current into the cell toward a specific direction, and therefore leakage can result in hard, soft, as well as transient faults.

Short at SBg

This short creates a connection between the BL and GND, which forces the BL to be discharged toward 0 V and resulting in a bias against writing and reading a correct 1 in the inspected cell. The two floating voltages resulting from this defect are the stored cell voltage and the voltage on the defective BL. The resulting faulty behavior can be represented as $<xw\bar{x}/x/->$, which requires initializing the cell to x, followed by a sensitizing $w\bar{x}$ operation that fails to properly change the voltage in the cell, leaving a faulty value of x behind.

The BG has no major impact on the faulty behavior of this defect, since the voltage on the BL is mainly influenced by the short itself and not by the sense voltage during reading. Since this defect results in FPs that do not require initializing operations nor completing operations, the faults are not attributed as partial nor dirty. At the same time, this defect does not force leakage current into the cell toward a specific direction, and therefore leakage can cause hard, soft as well as transient faults.

Bridge BO3, BO4 and BO5 between cell and a BL

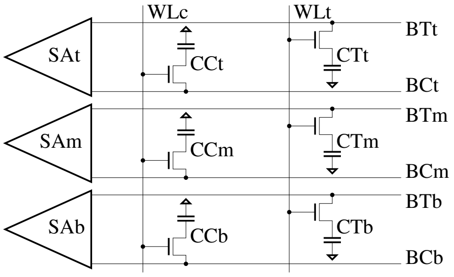


The nodes to which these bridges are connected are indicated in the figure to the right. These defects connect the cell storage capacitance to a BL (either to its own, to a complement, or to an unrelated BL), thereby forcing the stored cell voltage to follow the voltage present on the BL during memory operation. The two floating voltages caused by these defects are the voltage within the defective cell and the voltage on the defective BL. As a result of the defects, the cell leaks its charge gradually to the BL, and exhibits a typical state fault where the cell spontaneously looses its charge after short amount of time. This defect also causes read operations to fail for low defect resistances, as a result of the decreased cell charge. The state fault is represented by the FP $<x/\bar{x}/->$. Bridges with higher R_{br} values result in a weaker leakage current into the cell, thereby forcing the state fault to become a soft state fault.

The BG has a significant influence on the FP of these defects, since sensing takes place for a depleted cell charge and a low voltage margin, making BL coupling rather significant in controlling the sensed result. According to Table 7.10, the simulated defect causes the FP $<0/1/->$ for BGs 000 and 100, while it causes the FP $<1/0/->$ for BGs 011 and 111.

Since the defect forces the leakage current into the cell to flow toward $V_{dd}/2$, the fault can either be a hard fault (in the case of strong leakage) or a soft fault (in the case of weak leakage). However, the fault cannot be attributed as transient, since the passing of time can only strengthen the fault effect.

Bridge BW1 between two cells on the same BL pair



The nodes to which this bridge is connected are indicated in the figure to the right. This bridge connects two cells on the same BL pair to each other, which results in both cells to share their charges together and in writing both cells when one of them is written. The two floating voltages caused by this defect are the voltages within the two bridged cells. The faulty behavior resulting from this defect can be described by the FP $<wx^2;\bar{x}/x/->_{a,v}$, which is a coupling fault where writing the aggressor with the opposite value of that in the victim results in flipping the value of the victim.

The BG has a big impact on the faulty behavior. For the simulated defect, Table 7.10 indicates that the defect causes the FP $<w1^2;0/1/->$ for BGs 000 and 100, while it causes the FP $<w0^2;1/0/->$ for BGs 011 and 111. Since the fault requires multiple sensitizing operations, the faulty behavior is attributed as partial.

At the same time, since the defect does not force the leakage current into the cell toward a specific direction, the fault can be attributed as hard, soft as well as transient.

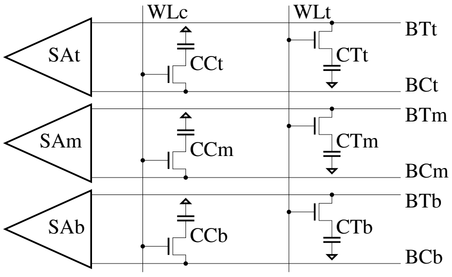
Bridge BW2 between two cells on adjacent BL pairs

This bridge connects two cells connected to two adjacent BL pairs, but share the same WL, which makes them share their charge and influence each other during write operations. The two floating voltages caused by this defect are the voltages within the two bridged cells. Note that in this case, the aggressor cell is the same as the bottom cell of the BG, which means that the BG here consists of only the top neighbor of the victim cell. The general faulty behavior of this $<wx^2\ w\bar{x}/x/->$, where two wx operations are used to initialize the cell to a specific voltage followed by a sensitizing $w\bar{x}$ which leaves x behind in the cell.

Although this defect is very similar to the bridge BW1 which connects two cells on the same BL pair, the resulting faulty behavior is very different. The reason for the difference is that, for BW2, both bridged cells are accessed during a write or read operation, whereas only one cell is accessed in the case of BW1.

Since the defect causes FPs that require multiple initializing write operations, the resulting faulty behavior can be attributed as partial. At the same time, since the defect does not force leakage into the cell toward a specific direction, the fault can be hard, soft and transient.

Bridge BZ2 between BTm (BCm) and BCt (BTb)



The nodes to which this bridge is connected are indicated in the figure to the right. This bridge connects two BLs on adjacent BL pairs together, resulting in influencing both read and write operations on all cells that belong to these two bridged BLs. For the bridge $BZ2_s$, the two floating nodes are chosen to be CTm and CTt, while for the bridge $BZ2_c$, the two floating nodes are chosen to be CCm and CCb. Note that the second floating node is the same as the neighboring cell (either top or bottom) used for the BG. Therefore, the BG for this defect contains only one cell.

The faulty behavior resulting from this defect can be represented as $<xw\bar{x}/x/->$. The defect does not force leakage into the cell toward a specific direction, and therefore the resulting fault can be hard, soft and transient.

Bridge BZ3 between BCm (BTm) and BTb (BCt)

This bridge connects two BLs on adjacent BL pairs together, resulting in influencing both read and write operations on all cells that belong to these two bridged BLs.

This bridge is similar to BZ2, with the exception that the bridge is connected to the complementary BL of the victim cell. The two floating voltages caused by this defect are chosen to be the voltage within the victim cell, and the voltage within the neighboring aggressor cell. The second floating voltage is the same as the neighboring cell used for the BG and, therefore, the BG for this defect contains only one cell.

The faulty behavior resulting from this defect can be represented as $<xw\bar{x}/x/->$. The defect does not force leakage into the cell toward a specific direction, and therefore the resulting fault can be hard, soft and transient.

Bridges BZ4 and BZ5 (between BLs and WLs)

These defects bridge the BL to the WL of the victim cell, causing the voltage on the BL to be influenced by the strong voltage drivers that set the voltage on the WL. The faulty behavior resulting from these defects can be represented as $<xw\bar{x}/x/->$. The BG has limited impact on this faulty behavior, since the BL is mainly influenced by the strong WL voltage during sensing rather than being influenced by the small coupling voltages. These defects do not force call leakage current toward a specific direction, which results in hard, soft as well as transient faults.

Bridge BZ6 between WTr (WCr) and WTl (WCl)

This defect bridges the active WL connected to the victim with an idle WL, which prevents setting the proper voltage on the WL at the right time. The general faulty behavior resulting from this defect can be represented as $<xw\bar{x}/x/->$. Since WL is driven by strong voltage drivers, this fault does not take place unless the bridge has very high bridge resistances. This bridge does not force cell leakage toward a specific direction, which means that the sensitized fault can be hard, soft and transient.

Summary

This chapter presented the application results of the simulation-based fault analysis algorithms, meant to evaluate the faulty behavior of a defective memory. The results indicate the effectiveness of the proposed fault analysis algorithms, and their ability to describe any DRAM-specific faulty behavior, within a reasonable amount of simulation time. The main issues presented in this chapter are as follows.

- Introduction of the reduced Spice simulation model of the memory, used to perform the fault analysis in this chapter. Depending on the simulation time of the model and the available computing power, a reduced memory model is important to limit the amount of needed simulation time to perform the simulation-based fault analysis.

- Discussion of Spice model reduction techniques, used to reduce the size of a memory simulation model, while keeping the high accuracy of the simulation. These techniques are particularly suited for memory devices, based on a number of memory-specific characteristics related to its design, structure and operation.

- Presentation of the different defects that may take place in the memory and the way they are represented using an electrical Spice model. These defects are also classified into three different classes (opens, shorts and bridges), and their possible positions in the memory are shown.

- Evaluation of the faulty behavior of those defects that result in only one floating node, which require the application of the 1D fault analysis, as well as those defect that result in two floating nodes and, therefore, require the application of the 2D fault analysis method.

- The conclusion that, with the exception of dirty faults, most DRAM-specific faults as described in Chapter 4 are very common and do take place in practice. Dirty faults, on the other hand, take place for special DRAM designs only and, therefore, they are not observed in the analysis performed in this chapter.

Contents of this chapter

8.1 Taxonomy of DRAM faults
8.2 DRAM-specific tests
8.3 Customizing march tests

8

Space of DRAM tests

Based on a full analysis of possible fault models in a given memory device, we can derive the full space of possible tests for that memory. The full space of DRAM faults has been analyzed in Section 4.3.3 of this thesis. This chapter uses the space of faults to derive the full space of DRAM tests. In addition, results of the simulations performed in Chapter 7 are used to identify those tests needed to realistically test a given defective memory with a known internal structure.

Section 8.1 starts with a taxonomy of the space of DRAM faults, followed by the derivation of the space of DRAM tests in Section 8.2. Section 8.3 discusses a number of practical aspects needed to be taken into consideration when tests are to be applied on a memory in practice.

8.1 Taxonomy of DRAM faults

First, we discuss the full space of DRAM-specific faults, based on all theoretically possible combinations of faults. Then, we discuss those faults that are practically expected to take place in the memory.

8.1.1 Theoretical fault space

Any generic memory FP, described in Section 4.2, can represent a DRAM-specific fault by adding a DRAM-specific fault attribute to it [see Section 4.3.3]. For example, it is possible to construct a number of DRAM-specific versions of the transition fault, such as the partial transition fault, the dirty transition fault, the soft transition fault, and so on. There are five DRAM-specific attributes, classified into two different classes. The first one is the class of voltage dependent faults, which consists of the following two fault attributes: partial faults (p), and dirty faults (d). The second one is the class of time dependent faults, which consists of the following three fault attributes: hard (h), soft (s) and transient (t) faults. It is important to note here that there are two different types of partial faults, one is the initialization related partial fault (p_i), while the other is the activation (or sensitization) related partial fault (p_a).

In addition to these individual attributes, it is possible to have multiple attributes at the same time associated with a given generic fault model. As a result, it is possible to establish the whole space of DRAM faults by considering the possibility that multiple attributes apply to a given fault at the same time. In order to do this, it is important to consider whether different DRAM-specific attributes are compatible with each other. Below, we consider the following three possibilities:

- Compatibility of voltage dependent attributes with time dependent attributes
- Compatibility of voltage dependent attributes among themselves
- Compatibility of time dependent attributes among themselves

Voltage and time attributes compatibility— Voltage dependent attributes are based on the inability of the memory to write a full voltage into the memory cell, while time dependent attributes are caused by naturally occurring leakage currents depleting the stored cell voltage. Therefore, voltage and time dependent attributes are based on two physically independent root causes, which results in faults that are independent as well. This means that each generic fault can be associated with a voltage dependent attribute in combination with a time dependent attribute in the following way:

$$\text{Fault} = \left\{ \begin{array}{c} \text{Voltage dependent} \\ \text{attribute} \end{array} \right\} \left\{ \begin{array}{c} \text{Time dependent} \\ \text{attribute} \end{array} \right\} \text{FP} \tag{8.1}$$

Voltage attributes compatibility— The different attributes of voltage dependent faults are compatible, which means that they can be combined with each other. The reason for this is that voltage dependent faults influence the behavior of the memory in different parts of the sensitizing operation sequence (S) of the FP [see Section 4.3]. Table 4.3 shows that partial faults influence the behavior during the initialization (I) and activation (A), while dirty faults influence the behavior between I and A, and after A. More precisely, the set of voltage dependent faults is equal to {-, p, d, pd}, where - stands for no attribute, while pd stands for the combined attribute “partial dirty”. Here, it is important to note here that there are three different combinations of partial faults (p): the initialization related partial faults (p_i), the activation related partial faults (p_a), and the initialization and activation related partial fault (p_{ia}).

Time attributes compatibility— In contrast to voltage dependent attributes, the different attributes of time dependent faults (h, s and t), are not compatible with each other (i.e., they cannot be combined with each other). The reason for this is that each specific attribute of time dependent faults is based on the direction and strength of the leakage current [see Section 4.3.2]. A leakage current that falls within the specs of the memory leads to a hard fault (h), a leakage current exceeding the specs and supporting the applied operation leads to a transient fault (t), while a leakage current exceeding the specs and opposing the applied operation leads to a soft fault (s). As a result, only one time dependent behavior can take place at a given time, and for a given defect. More precisely, the set of time dependent faults is equal to {h, s, t}. It is worth noting here that the hard fault attribute does not modify a generic FP in any way, which means that it is identical to the absence of an attribute (symbolized by -).

In conclusion, Expression 8.1 can be expanded to describe all possible DRAM-specific faults as follows:

$$\text{Fault} = \left\{ \begin{array}{c} \text{-} \\ \text{p} \\ \text{d} \\ \text{pd} \end{array} \right\} \left\{ \begin{array}{c} \text{h} \\ \text{s} \\ \text{t} \end{array} \right\} \text{FP} \tag{8.2}$$

Expression 8.2 indicates that any generic fault model can be either regular (-), partial (p), dirty (d) or partial dirty (pd), while being hard (h or -), soft (s) or transient (t) at the same time. In total, this gives a space of $4 \times 3 = 12$ different attributes for DRAM-specific faults. Table 8.1 lists all these different attributes, and shows an example of how they can be attributed to the down transition fault (TF_0).

Fault #1 in the table is the hTF_0, which is identical to the generic TF_0. Fault #2 is the partial hard TF_0, which is denoted by the FP $<w1^i\ w0^a/1/->$. The i in the sequence $w1^i$ is caused by the initialization related partial fault (p_i), where it stands for the number of times the initializing $w1$ operation should be performed ($i \geq 0$). The a in the sequence $w0^a$ is caused by the activation related partial fault (p_a), where it stands for the number of times the activating $w0$ operation should be performed ($a \geq 1$). Fault #3 is the dirty hard TF_0, which is obtained by adding a completing sequence of operations ($[C]$) to the sensitizing operation sequence (S). Fault #4 in the table is the partial dirty hard TF_0, which is denoted by the FP $<w1^i\ w0^a\ [C]/1/->$. This fault contains i initializing $w1$ operations, it contains a activating $w0$ operations, in addition to the completing operation sequence $[C]$. The four types of hard transition faults can be represented by the general hard TF_0 (ghTF_0), denoted as $<w1^i\ w0^a\ [C^d]/1/->$, where $i \geq 0$, $a \geq 1$ and $d \in \{0, 1\}$. The ghTF_0 can be reduced to any type of hard TF_0 by properly setting its parameters i, a and d. For example, it can be reduced to the hTF_0 by using $i = 0$, $a = 1$ and $d = 0$; it can be reduced to the $\text{p}_\text{ia}\text{hTF}_0$ by using $d = 0$; and it can be reduced to the dhTF_0 by using $i = 0$, $a = 1$ and $d = 1$.

Fault #5 in the table is the soft TF_0, since it adds a delay time (T) to the

Table 8.1. Space of DRAM-specific faults for the down transition fault (TF$_0$).

#	Fault	FP	Name
1	hTF$_0$	$<1w0/1/->$	hard transition 0 fault
2	p$_{\text{ia}}$hTF$_0$	$<w1^i w0^a/1/->$	partial hard transition 0 fault
3	dhTF$_0$	$<1w0[C]/1/->$	dirty hard transition 0 fault
4	p$_{\text{ia}}$dhTF$_0$	$<w1^i w0^a[C]/1/->$	partial dirty hard transition 0 fault
	ghTF$_0$	$<w1^i w0^a[C^d]/1/->$	general hard transition 0 fault
5	sTF$_0$	$<1w0_T/1/->$	soft transition 0 fault
6	p$_{\text{ia}}$sTF$_0$	$<w1^i w0^a_T/1/->$	partial soft transition 0 fault
7	dsTF$_0$	$<1w0[C]_T/1/->$	dirty soft transition 0 fault
8	p$_{\text{ia}}$dsTF$_0$	$<w1^i w0^a[C]_T/1/->$	partial dirty soft transition 0 fault
	gsTF$_0$	$<w1^i w0^a[C^d]_T/1/->$	general soft transition 0 fault
9	tTF$_0$	$<1\underline{w0}/1_L/->$	transient transition 0 fault
10	p$_{\text{ia}}$tTF$_0$	$<\underline{w1}^i \underline{w0}^a/1_L/->$	partial transient transition 0 fault
11	dtTF$_0$	$<1\underline{w0}[\underline{C}]/1_L/->$	dirty transient transition 0 fault
12	p$_{\text{ia}}$dtTF$_0$	$<\underline{w1}^i \underline{w0}^a[\underline{C}]/1_L/->$	partial dirty transient transition 0 fault
	gtTF$_0$	$<\underline{w1}^i \underline{w0}^a[\underline{C}^d]/1_L/->$	general transient transition 0 fault

sensitizing operation of the generic transition fault. In the same way, a soft version of the partial, dirty, and partial dirty TF$_0$ is obtained by adding a T to their respective sensitizing sequences. And finally, all these soft TF$_0$ can be represented by the general soft TF$_0$ (gsTF$_0$), denoted as $<w1^i\ w0^a\ [C^d]_T/1/->$, where $i \geq 0$, $a \geq 1$ and $d \in \{0, 1\}$. The gsTF$_0$ can be reduced to any type of soft TF$_0$ by properly setting its three parameters i, a and d. It can be reduced to the sTF$_0$ by using $i = 0$, $a = 1$ and $d = 0$, it can be reduced to the p$_{\text{ia}}$sTF$_0$ by using $d = 0$, and it can be reduced to the dsTF$_0$ by using $i = 0$, $a = 1$ and $d = 1$.

Faults #9 through #12 in the table represent the four different types of transient TF$_0$, since they add a life time L to the fault effect R. In addition, an underscore is added to each operation in S, which means that these operations must be performed in *back-to-back* mode (i.e., instantaneously after each other and without delay)[1]. The general transient TF$_0$ (gtTF$_0$) is represented by the FP $<\underline{w1}^i\ \underline{w0}^a\ [\underline{C}^d]/1_L/->$, where $i \geq 0$, $a \geq 1$ and $d \in \{0, 1\}$. The gtTF$_0$ can be to any type of transient TF$_0$ by properly setting its three parameters i, a and d.

---

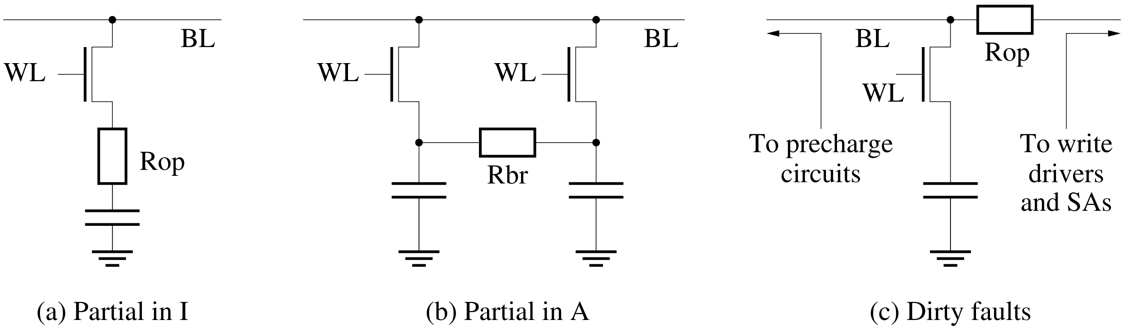
[1] In terms of detection conditions, an underscore below operations in a transient fault means that the operations have to be performed after each other within one march element. For example, if $S = w1\underline{w0}$ then the detection condition should be $\Updownarrow$(..., $w1$, $w0$, ...).

8.1.2 Realistic fault space

In this section, we describe the DRAM-specific fault attributes that may take place in practice when a given defect affects the behavior of the memory. All possible memory cell array opens, shorts and bridges have been classified and simulated in Chapter 7. Each defect in Table 7.8, when injected into the memory, causes its own type of faulty behavior, which in turn can be modeled by a fault from the space of fault primitives (such as those listed in Table 4.1), in combination with some of the 12 DRAM-specific fault attributes listed in Table 8.1.

The results of the simulation-based fault analysis in this thesis, and others [Al-Ars99], performed to analyze the faulty behavior of DRAMs, indicate the following realistic restrictions on the space of DRAM faults:

1. Restrictions for single-cell faults.
    (a) State faults may not be partial. Although this is theoretically possible, partial state faults have not been observed in practice, because they are commonly caused by strong defects that fail the memory easily, without the need for a special initialization and/or activation.
    (b) Single-cell faults can suffer from initialization partial faults (p_i), but not activation partial faults (p_a). This is true since single-cell faults are sensitized by applying an operation on the victim [see Figure 8.1(a)]. The activation of this fault has to fail in order to cause a fault effect. Hence, repeating this operation reinforces the proper behavior rather than the faulty behavior.



(a) Partial in I (b) Partial in A (c) Dirty faults

Figure 8.1. Defects causing (a) partial faults in I, (b) in A, and (c) causing dirty faults.

2. Restrictions for two-cell faults.
    (a) State coupling faults may not be partial, as these have not been observed in practice. For example, $<0;1/0/->$ may not have the form $<w0^h;1/0/->$ [see note about state faults in the restrictions for single-cell faults above].

   (b) Coupling faults may not suffer from sensitization related partial faults ($\mathrm{p_a}$), in case the sensitizing operation is performed on the victim. This due to the same reasons outlined for single-cell faults. For example, the fault $<1; 1w0/1/->$ may not have the from $<1; 1w0^h/1/->$.

   (c) Only the aggressor in a coupling fault may suffer from initialization related partial faults ($\mathrm{p_i}$), since the aggressor is the cell that causes the fault. Figure 8.1(b) shows a typical bridge that causes coupling faults, where it is clear that the fault effect can be strengthened by performing multiple operations on one cell, but not on both.

   (d) Coupling faults may not be dirty. Although theoretically possible, these have not been observed in simulations.

3. General restrictions. The completing operation $[C]$ of dirty faults (d) can either be a write or a read operation performed on a cell (a) along the same bit line of the faulty cell (v), but with the opposite data to the sensitizing operation on v. Figure 8.1(c) shows an example of a defect on BL that causes dirty faults.

As an example of the realistic restrictions on the space of DRAM faults, Table 8.2 shows the realistic DRAM fault space of the down transition fault ($\mathrm{TF_0}$). Based on the restrictions listed above, the full space of DRAM $\mathrm{TF_0}$ faults shown in Table 8.1 is reduced to that shown in Table 8.2. The only difference between the two tables is that the partial fault is only applicable for the initialization part of the fault ($\mathrm{p_i}$), and not for the activation part ($\mathrm{p_a}$). More precisely, the only restriction on the full space of DRAM transition faults is that $a = 1$.

In summary, the difference between realistic single and two-cell DRAM-specific faults can be represented in the following two expressions.

$$\text{Single-cell fault} = \left\{ \begin{array}{c} \text{-} \\ \mathrm{p_i} \\ \mathrm{d} \\ \mathrm{pd} \end{array} \right\} \left\{ \begin{array}{c} \mathrm{h} \\ \mathrm{s} \\ \mathrm{t} \end{array} \right\} \mathrm{FP}, \text{Two-cell fault} = \left\{ \begin{array}{c} \text{-} \\ \mathrm{p} \end{array} \right\} \left\{ \begin{array}{c} \mathrm{h} \\ \mathrm{s} \\ \mathrm{t} \end{array} \right\} \mathrm{FP} \qquad (8.3)$$

These expressions indicate that there are fewer realistic restrictions on single-cell faults than there are on two-cell faults. Single-cell faults can be attributed as partial, dirty and partial dirty, while two-cell faults can only be attributed as partial.

8.2 DRAM-specific tests

In this section, we use the space of DRAM-specific faults derived in the previous section to identify the space of DRAM-specific tests, that would detect any possible DRAM fault. We base our analysis on single-cell and two-cell static faults,

Table 8.2. Realistic space of DRAM-specific faults for the down transition fault (TF$_0$).

#	Fault	FP	Name
1	hTF$_0$	$<1w0/1/->$	hard transition 0 fault
2	p$_\mathrm{i}$hTF$_0$	$<w1^iw0/1/->$	partial hard transition 0 fault
3	dhTF$_0$	$<1w0[C]/1/->$	dirty hard transition 0 fault
4	p$_\mathrm{i}$dhTF$_0$	$<w1^iw0[C]/1/->$	partial dirty hard transition 0 fault
	ghTF$_0$	$<w1^iw0[C^d]/1/->$	general hard transition 0 fault
5	sTF$_0$	$<1w0_T/1/->$	soft transition 0 fault
6	p$_\mathrm{i}$sTF$_0$	$<w1^iw0_T/1/->$	partial soft transition 0 fault
7	dsTF$_0$	$<1w0[C]_T/1/->$	dirty soft transition 0 fault
8	p$_\mathrm{i}$dsTF$_0$	$<w1^iw0[C]_T/1/->$	partial dirty soft transition 0 fault
	gsTF$_0$	$<w1^iw0[C^d]_T/1/->$	general soft transition 0 fault
9	tTF$_0$	$<1\underline{w0}/1_L/->$	transient transition 0 fault
10	p$_\mathrm{i}$tTF$_0$	$<\underline{w1}^i\underline{w0}/1_L/->$	partial transient transition 0 fault
11	dtTF$_0$	$<1\underline{w0}[\underline{C}]/1_L/->$	dirty transient transition 0 fault
12	p$_\mathrm{i}$dtTF$_0$	$<\underline{w1}^i\underline{w0}[\underline{C}]/1_L/->$	partial dirty transient transition 0 fault
	gtTF$_0$	$<\underline{w1}^i\underline{w0}[\underline{C}^d]/1_L/->$	general transient transition 0 fault

described in Section 4.2, and start with tests for hard, then transient, and finally for soft faults.

8.2.1 Detecting hard faults

Hard faults are time-independent faults that get sensitized once their sensitizing operation sequence is performed, and they remain sensitized afterwards until they are detected. In the following, we discuss the detection conditions needed to detect each of the hard faults first, and then we list the tests to detect these faults.

Detection conditions for hard faults

According to Section 8.1.2, a single-cell hard fault can either be partial with respect to initialization (p$_\mathrm{i}$h), dirty (dh), or both (p$_\mathrm{i}$dh). The fault p$_\mathrm{i}$h is modeled by multiple initialization operations, while the fault dh is modeled by performing a write or read operation on a cell along the same BL as the faulty cell, but with opposite data to the sensitization.

Table 8.3 lists all single-cell hard faults, along with the detection conditions needed to detect them [compare with Table 4.1]. The table considers the general form of single-cell hard faults, where both partial, as well as dirty faults take place. The detection conditions are designed to detect both faults as well. For example, the (partial, dirty and hard) write-0 destructive fault (p$_\mathrm{i}$dh WDF$_0$), must first be initialized a multiple number of times ($w0_v^h$). Then, a completing operation with data 1 (a value opposite to that of the sensitizing value) must be applied to a different cell along the same BL ($[O1_a]$), before the sensitizing write 0 operation can be performed ($w0_v$). The only requirement the completing write operation has to fulfill is to change the state of the BLs connected to the victim cell. The exact address of the aggressor is therefore not important, only the fact that it lies along the same BL. The detection condition starts with multiple $w0$ operations to initialize the cell to 0. The operation $O1$ ensures that the opposite data is present in a cell along the same BL just before any cell (with address higher than the current cell) is sensitized by the $w0$ operation. Then, the read operation ensures the detection of the fault.

Table 8.3. List of single-cell, hard FPs and their detection conditions. The completing operation Ox_a is performed with a value (x) opposite to that in the sensitizing operation and to a different cell (a) along the same BL.

#	Fault	$<S/F/R>$, $O \in \{w, r\}$	Detection condition, $O \in \{w, r\}$
1	dh SF$_0$	$<0_v[O1_a]/1/->_{a\in BL(v)}$	$\Updownarrow(...w0, ...O1a, ...r0, ...)$
2	dh SF$_1$	$<1_v[O0_a]/0/->_{a\in BL(v)}$	$\Updownarrow(...w1, ...O0a, ...r1, ...)$
3	p$_\mathrm{i}$dh WDF$_0$	$<w0_v^h[O1_a]w0_v/1/->_{a\in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...w0, ...r0, ...)$
4	p$_\mathrm{i}$dh WDF$_1$	$<w1_v^h[O0_a]w1_v/0/->_{a\in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...w1, ...r1, ...)$
5	p$_\mathrm{i}$dh TF$_1$	$<w0_v^h[O0_a]w1_v/0/->_{a\in BL(v)}$	$\Updownarrow(...w0^h, ...O0_a, ...w1, ...r1, ...)$
6	p$_\mathrm{i}$dh TF$_0$	$<w1_v^h[O1_a]w0_v/1/->_{a\in BL(v)}$	$\Updownarrow(...w1^h, ...O1_a, ...w0, ...r0, ...)$
7	p$_\mathrm{i}$dh IRF$_0$	$<w0_v^h[O1_a]r0_v/0/1>_{a\in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...r0, ...)$
8	p$_\mathrm{i}$dh IRF$_1$	$<w1_v^h[O0_a]r1_v/1/0>_{a\in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...r1, ...)$
9	p$_\mathrm{i}$dh DRDF$_0$	$<w0_v^h[O1_a]r0_v/1/0>_{a\in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...r0, ...r0, ...)$
10	p$_\mathrm{i}$dh DRDF$_1$	$<w1_v^h[O0_a]r1_v/0/1>_{a\in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...r1, ...r1, ...)$
11	p$_\mathrm{i}$dh RDF$_0$	$<w0_v^h[O1_a]r0_v/1/1>_{a\in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...r0, ...)$
12	p$_\mathrm{i}$dh RDF$_1$	$<w1_v^h[O0_a]r1_v/0/0>_{a\in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...r1, ...)$

Table 8.4 lists all two-cell hard faults, along with the detection conditions needed to detect them [compare with Table 4.2]. The table considers the general form of two-cell hard faults, where both partial, as well as dirty faults take place. The detection conditions are designed to detect both faults. When operations need to be performed on both the aggressor *and* the victim to sensitize the corresponding fault, then we assume that these operations can be performed in any order, starting with the aggressor then the victim, or vice versa. Some faults have two detection conditions: one is needed to sensitize the fault when the aggressor has a lower memory address than the victim ($a < v$), while the other is needed when $a > v$. Some faults, on the other hand, require only one detection condition (such as FP #1 and #25). This is possible when the initialization, sensitization and detection

Table 8.4. List of two-cell, hard FPs and their detection conditions.

# Fault	$<S_a; S_v/F/R>$	Detection condition	
		$a < v$	$a > v$
1 h $\text{SF}_{0;0}$	$<0; 0/1/->$	$\Updownarrow(...0); \Updownarrow(..., r0, ...)$	
2 h $\text{SF}_{0;1}$	$<0; 1/0/->$	$\Uparrow(r1, ...w0, ...)$	$\Downarrow(r1, ...w0, ...)$
3 h $\text{SF}_{1;0}$	$<1; 0/1/->$	$\Uparrow(r0, ...w1, ...)$	$\Downarrow(r0, ...w1, ...)$
4 h $\text{SF}_{1;1}$	$<1; 1/0/->$	$\Updownarrow(...1); \Updownarrow(..., r1, ...)$	
5 p_ah $\text{CFds}_{0w0;0}$	$<0w0^h; 0/1/->$	$\Downarrow(...w0^h); \Updownarrow(r0...)$	$\Uparrow(...w1^h); \Updownarrow(r1...)$
6 p_ah $\text{CFds}_{0w0;1}$	$<0w0^h; 1/0/->$	$\Uparrow(r1, ..., w0^h, ...)$	$\Downarrow(r1, ..., w0^h, ...)$
7 p_ah $\text{CFds}_{1w1;0}$	$<1w1^h; 0/1/->$	$\Uparrow(r0, ..., w1^h, ...)$	$\Downarrow(r0, ..., w1^h, ...)$
8 p_ah $\text{CFds}_{1w1;1}$	$<1w1^h; 1/0/->$	$\Downarrow(...w1^h); \Updownarrow(r1...)$	$\Uparrow(...w1^h); \Updownarrow(r1...)$
9 p_ah $\text{CFds}_{0w1;0}$	$<0w1^h; 0/1/->$	$\Uparrow(r0, ..., w1^h, ...)$	$\Downarrow(r0, ..., w1^h, ...)$
10 p_ah $\text{CFds}_{0w1;1}$	$<0w1^h; 1/0/->$	$\Downarrow(...0, w1^h); \Updownarrow(r1...)$	$\Uparrow(...0, w1^h); \Updownarrow(r1...)$
11 p_ah $\text{CFds}_{1w0;0}$	$<1w0^h; 0/1/->$	$\Downarrow(...1, w0^h); \Updownarrow(r0...)$	$\Uparrow(...1, w0^h); \Updownarrow(r0...)$
12 p_ah $\text{CFds}_{1w0;1}$	$<1w0^h; 1/0/->$	$\Uparrow(r1, ..., w0^h, ...)$	$\Downarrow(r1, ..., w0^h, ...)$
13 p_ah $\text{CFds}_{0r0;0}$	$<0r0^h; 0/1/->$	$\Uparrow(r0^h, ...)$	$\Downarrow(r0^h, ...)$
14 p_ah $\text{CFds}_{0r0;1}$	$<0r0^h; 1/0/->$	$\Downarrow(r0^h, ...1); \Updownarrow(r1...)$	$\Uparrow(r0^h, ...1); \Updownarrow(r1...)$
15 p_ah $\text{CFds}_{1r1;0}$	$<1r1^h; 0/1/->$	$\Downarrow(r1^h, ...0); \Updownarrow(r0...)$	$\Uparrow(r1^h, ...0); \Updownarrow(r0...)$
16 p_ah $\text{CFds}_{1r1;1}$	$<1r1^h; 1/0/->$	$\Uparrow(r1^h, ...)$	$\Downarrow(r1^h, ...)$
17 p_ih $\text{CFwd}_{0;0}$	$<w0^h; 0w0/1/->$	$\Uparrow(...w0^h); \Updownarrow(r0...)$	$\Downarrow(...w0^h); \Updownarrow(r0...)$
18 p_ih $\text{CFwd}_{1;0}$	$<w1^h; 0w0/1/->$	$\Updownarrow(...w1^h); \Downarrow(...0, w0); \Updownarrow(r0...)$	$\Updownarrow(...w1^h); \Uparrow(...0, w0); \Updownarrow(r0...)$
19 p_ih $\text{CFwd}_{0;1}$	$<w0^h; 1w1/0/->$	$\Updownarrow(...w0^h); \Downarrow(...1, w1); \Updownarrow(r1...)$	$\Updownarrow(...w0^h); \Uparrow(...1, w1); \Updownarrow(r1...)$
20 p_ih $\text{CFwd}_{1;1}$	$<w1^h; 1w1/0/->$	$\Uparrow(...w1^h); \Updownarrow(r1...)$	$\Downarrow(...w1^h); \Updownarrow(r1...)$
21 p_ih $\text{CFtr}_{0;0}$	$<w0^h; 1w0/1/->$	$\Updownarrow(...1); \Downarrow(...w0^h); \Updownarrow(r0...)$	$\Updownarrow(...1); \Uparrow(...w0^h); \Updownarrow(r0...)$
22 p_ih $\text{CFtr}_{1;0}$	$<w1^h; 1w0/1/->$	$\Updownarrow(...w1^h); \Downarrow(...w0); \Updownarrow(r0...)$	$\Updownarrow(...w1^h); \Uparrow(...w0); \Updownarrow(r0...)$
23 p_ih $\text{CFtr}_{0;1}$	$<w0^h; 0w1/0/->$	$\Updownarrow(...w0^h); \Downarrow(...w1); \Updownarrow(r1...)$	$\Updownarrow(...w0^h); \Uparrow(...w1); \Updownarrow(r1...)$
24 p_ih $\text{CFtr}_{1;1}$	$<w1^h; 0w1/0/->$	$\Updownarrow(...0); \Downarrow(...w1^h); \Updownarrow(r1...)$	$\Updownarrow(...0); \Uparrow(...w1^h); \Updownarrow(r1...)$
25 p_ih $\text{CFir}_{0;0}$	$<w0^h; 0r0/0/1>$	$\Updownarrow(..., w0^h); \Updownarrow(..., r0, ...)$	
26 p_ih $\text{CFir}_{1;0}$	$<w1^h; 0r0/0/1>$	$\Uparrow(r0, ..., w1^h)$	$\Downarrow(r0, ..., w1^h)$
27 p_ih $\text{CFir}_{0;1}$	$<w0^h; 1r1/1/0>$	$\Uparrow(r1, ..., w0^h)$	$\Downarrow(r1, ..., w0^h)$
28 p_ih $\text{CFir}_{1;1}$	$<w1^h; 1r1/1/0>$	$\Updownarrow(..., w1^h); \Updownarrow(..., r1, ...)$	
29 p_ih $\text{CFdr}_{0;0}$	$<w0^h; 0r0/1/0>$	$\Updownarrow(..., w0^h); \Updownarrow(..., r0, r0, ...)$	
30 p_ih $\text{CFdr}_{1;0}$	$<w1^h; 0r0/1/0>$	$\Uparrow(r0, r0, ..., w1^h)$	$\Downarrow(r0, r0, ..., w1^h)$
31 p_ih $\text{CFdr}_{0;1}$	$<w0^h; 1r1/0/1>$	$\Uparrow(r1, r1, ..., w0^h)$	$\Downarrow(r1, r1, ..., w0^h)$
32 p_ih $\text{CFdr}_{1;1}$	$<w1^h; 1r1/0/1>$	$\Updownarrow(..., w1^h); \Updownarrow(..., r1, r1, ...)$	
33 p_ih $\text{CFrd}_{0;0}$	$<w0^h; 0r0/1/1>$	$\Updownarrow(..., w0^h); \Updownarrow(..., r0, ...)$	
34 p_ih $\text{CFrd}_{1;0}$	$<w1^h; 0r0/1/1>$	$\Uparrow(r0, ..., w1^h)$	$\Downarrow(r0, ..., w1^h)$
35 p_ih $\text{CFrd}_{0;1}$	$<w0^h; 1r1/0/0>$	$\Uparrow(r1, ..., w0^h)$	$\Downarrow(r1, ..., w0^h)$
36 p_ih $\text{CFrd}_{1;1}$	$<w1^h; 1r1/0/0>$	$\Updownarrow(..., w1^h); \Updownarrow(..., r1, ...)$	

of the fault requires exactly the same data, *and* when it is possible to interchange the order of the operations performed on the aggressor and the victim.

As an example of the faults in Table 8.4, consider the (partial, hard) $0w0$ disturb 0 coupling fault ($\mathrm{p_a h\ CFds}_{0w0;0}$). When $a < v$, this fault is sensitized by performing a write 0 a multiple number of times w^h in a decreasing address order. This way, all victim cells are first initialized to 0, and the fault is then sensitized by the hammer $w0$ sequence on all potential a cells with an address less than that of v. The fault is then detected by a read 0 operation in any address order. When $a > v$, the same sequence of sensitizing write operations should be performed in an increasing address order.

Tests for hard faults

Based on the detection conditions in Tables 8.3 and 8.4, it is possible to derive memory tests that detect all single-cell and two-cell hard faults. March 1CH below detects all single-cell hard faults.

$$\begin{array}{lccc} \text{March 1CH} = \{ & \Updownarrow(w0^h,\, w1_a,\, r0,\, r0); & \Updownarrow(w1^h,\, w0_a,\, r1,\, r1); & \Updownarrow(w0^h,\, w1_a,\, w0,\, r0); \\ & \text{ME0} & \text{ME1} & \text{ME2} \\ & \Updownarrow(w1^h,\, w0_a,\, w1,\, r1); & \Updownarrow(w0^h,\, w1,\, r1); & \Updownarrow(w1^h,\, w0,\, r0)\} \\ & \text{ME3} & \text{ME4} & \text{ME5} \end{array}$$

This march test has six march elements (ME0 through ME5), each of which begins with a hammer write operation and ends with a detecting read operation. Each two consecutive march elements represent the exact complement of each other, as they are generated to target complementary FPs. The test substitutes the dirty operation (O) in the detection conditions of Table 8.3 by a write operation, since this choice reduces the length of the test when the completing operation needs to change the data present in a. The test has a relatively high complexity of $(16 \cdot n + 6 \cdot h \cdot n)$ compared to other single-cell march tests, as a result of the partial and the dirty DRAM-specific faults.

Table 8.5 lists all march elements in March 1CH along with the hard single-cell FPs, and indicates the first memory operation in the test that detects the corresponding FP. For example, ME0 (march element 0) shares an entry #1/3 with dh SF. This entry means that FP #1, which refers to the fault dh SF_0, is first detected in March 1CH by the 3rd operation of ME0. The table shows that each of ME0 and ME1 detect 4 different FPs, while each of ME2, ME3, ME4 and ME5 detect a single FP.

A march test that detects all two-cell hard faults can be represented by March 2CH below.

$$\begin{array}{lccc} \text{March 2CH} = \{ & \Updownarrow(w0^h); & \Uparrow(r0^h, w1^h); & \Uparrow(r1^h, w0^h); \\ & \text{ME0} & \text{ME1} & \text{ME2} \\ & \Downarrow(r0^h, w1^h); & \Downarrow(r1^h, w0^h); & \Updownarrow(r0)\} \\ & \text{ME3} & \text{ME4} & \text{ME5} \end{array}$$

Table 8.5. Detection capabilities of march elements in March 1CH.

Fault	dh SF	$\text{p}_\text{i}$dh WDF	$\text{p}_\text{i}$dh TF	$\text{p}_\text{i}$dh IRF	$\text{p}_\text{i}$dh DRDF	$\text{p}_\text{i}$dh RDF
ME0	#1/3	–	–	#7/3	#9/4	#11/3
ME1	#2/3	–	–	#8/3	#10/4	#12/3
ME2	–	#3/4	–	–	–	–
ME3	–	#4/4	–	–	–	–
ME4	–	–	#5/3	–	–	–
ME5	–	–	#6/3	–	–	–

This march test has 6 march elements (ME0 through ME5), many of which begin with a hammer read operation and end with a hammer write operation. These sequences are characteristic for march tests that aim to detect two-cell faults. The march element ME1 is the exact complementary of ME3, while ME2 is the exact complementary of ME4. This results from the fact that these march elements are constructed to detect complementary FPs. This test has a complexity of $n+9\cdot n\cdot h$.

Table 8.6 lists all march elements in March 2CH along with the hard two-cell FPs, and indicates the first memory operation in the test that detects the corresponding FP. There are two columns for each fault, one for $a < v$ and the other $a > v$. For example, ME1 (march element 1) shares an entry #1 with h SF. This entry means that FP #1, which refers to the fault h $\text{SF}_{0;0}$, is first detected in March 2CH by the read operations in ME1.

8.2.2 Detecting transient faults

The idea of transient FPs implies that, after a fault is sensitized, leakage results in correcting the faulty behavior before it is detected on the output. In the following, we first discuss the detection conditions needed to detect transient faults, and then we introduce the tests that should detect these faults.

Detection conditions for transient faults

An FP has two components to describe a fault: F (the value of the faulty cell) and R (the output on a read operation). It is possible for F to change in such a way that a faulty state would be transformed into a proper state within the cell. However, the value of R cannot be transient, since it gets sensitized *and* detected on the output at the same time.

For example, the FP $<1w1/0/->$ represents a non-transition write 1 operation that sets a faulty 0 into the memory cell. Under the assumption that faults are constant in time, this FP would remain sensitized until a read operation is performed to detect this fault on the output. Under the assumption of transient FPs, this FP

Table 8.6. Detection capabilities of march elements in March 2CH.

ME	h SF		$\mathrm{p_a}$h CFds		$\mathrm{p_i}$h CFwd		$\mathrm{p_i}$h CFtr		$\mathrm{p_i}$h CFir		$\mathrm{p_i}$h CFdr		$\mathrm{p_i}$h CFrd	
	$a < v$	$a > v$	$a < v$	$a > v$	$a < v$	$a > v$	$a < v$	$a > v$	$a < v$	$a > v$	$a < v$	$a > v$	$a < v$	$a > v$
ME0	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ME1	#1		#7	–	–	–	–	–	#26	–	#30	–	#34	–
	#3	–	#9	–	–	–	–	–	–	–	–	–	–	–
	–	–	#13	–	–	–	–	–	–	–	–	–	–	–
ME2	#2	–	#6	#8	#20	#19	–	#23	#27	–	#31	–	#35	–
	#4		#12	#10	–	–	–	#24	#28		#32		#36	
	–	–	#16	#14	–	–	–	–	–	–	–	–	–	–
ME3	–	#3	–	#7	#17	#18	–	#21	#25		#29		#33	
	–	–	–	#9	–	–	–	#22	–	#26	–	#30	–	#34
	–	–	–	#11	–	–	–	–	–	–	–	–	–	–
	–	–	–	#13	–	–	–	–	–	–	–	–	–	–
	–	–	–	#15	–	–	–	–	–	–	–	–	–	–
ME4	–	#2	#8	#6	#19	#20	#23	–	–	#27	–	#31	–	#35
	–	–	#10	#12	–	–	#24	–	–	–	–	–	–	–
	–	–	#14	#16	–	–	–	–	–	–	–	–	–	–
ME5	–	–	#5	–	#18	#17	#21	–	–	–	–	–	–	–
	–	–	#11	–	–	–	#22	–	–	–	–	–	–	–
	–	–	#15	–	–	–	–	–	–	–	–	–	–	–

would only remain sensitized for a limited period of time and then the state of the cell gets automatically corrected to its expected value (i.e., logic 1). As a second example, consider the FP $<1r1/1/0>$ that represents a faulty read operation, resulting in an incorrect value on the output. It cannot be transient since the output of the faulty read operation is sensitized and detected simultaneously.

Table 8.7 lists all single-cell transient faults, along with their detection conditions [compare with Table 8.3]. The table takes both the initialization partial, as well as the dirty transient faults into consideration. For example, the (partial, dirty and transient) write-0 destructive fault ($\mathrm{p_i}$dt $\mathrm{WDF_0}$), must first be initialized a multiple number of times ($w0^h_v$). Then, a completing operation with data 1 must be applied to a different cell along the same BL ($[O1_a]$), before the sensitizing write 0 operation can be performed ($w0_v$). To ensure the detection of this transient fault, all of these operations must be applied directly after each other (back-to-back), and directly followed by a detecting read operation. The detection condition starts with multiple $w0$ operations to initialize the cell to 0, directly followed by $O1_a$, $w0$ and a detecting $r0$. Note that this detection condition is not a regular one, since it requires operations to be performed on two different cells (a and v) within only one march element. The fact that the operations in these detection conditions need to be performed back-to-back is indicated by the underscore below the corresponding operations.

Table 8.8 lists all two-cell transient faults, along with their detection conditions [compare with Table 8.4]. For example, the (partial, transient) $0w0$ disturb 0 coupling fault ($\mathrm{p_a}$t $\mathrm{CF_{0\mathit{w}0;0}}$), must first be initialized to a 0, then sensitized by writing

Table 8.7. List of single-cell, transient FPs and their detection conditions. The underlined operations must be performed back-to-back.

#	Fault	$<S/F_L/R>$, $O \in \{w, r\}$	Detection condition, $O \in \{w, r\}$
1	dt SF_0	$<0_v[\underline{O1}_a]/1_L/->_{a \in BL(v)}$	$\Updownarrow(..., w0, \underline{O1}_a, \underline{r0}, ...)$
2	dt SF_1	$<1_v[\underline{O0}_a]/0_L/->_{a \in BL(v)}$	$\Updownarrow(..., w1, \underline{O0}_a, \underline{r1}, ...)$
3	$\text{p}_\text{i}$dt WDF_0	$<\underline{w0}_v^h[\underline{O1}_a]\underline{w0}_v/1_L/->_{a \in BL(v)}$	$\Updownarrow(..., \underline{w0}^h, \underline{O1}_a, \underline{w0}, \underline{r0}, ...)$
4	$\text{p}_\text{i}$dt WDF_1	$<\underline{w1}_v^h[\underline{O0}_a]\underline{w1}_v/0_L/->_{a \in BL(v)}$	$\Updownarrow(..., \underline{w1}^h, \underline{O0}_a, \underline{w1}, \underline{r1}, ...)$
5	$\text{p}_\text{i}$dt TF_1	$<\underline{w0}_v^h[\underline{O0}_a]\underline{w1}_v/0_L/->_{a \in BL(v)}$	$\Updownarrow(..., \underline{w0}^h, \underline{O0}_a, \underline{w1}, \underline{r1}, ...)$
6	$\text{p}_\text{i}$dt TF_0	$<\underline{w1}_v^h[\underline{O1}_a]\underline{w0}_v/1_L/->_{a \in BL(v)}$	$\Updownarrow(..., \underline{w1}^h, \underline{O1}_a, \underline{w0}, \underline{r0}, ...)$
7	$\text{p}_\text{i}$dt IRF_0	$<\underline{w0}_v^h[\underline{O1}_a]\underline{r0}_v/0_L/1>_{a \in BL(v)}$	$\Updownarrow(..., \underline{w0}^h, \underline{O1}_a, \underline{r0}, ...)$
8	$\text{p}_\text{i}$dt IRF_1	$<\underline{w1}_v^h[\underline{O0}_a]\underline{r1}_v/1_L/0>_{a \in BL(v)}$	$\Updownarrow(..., \underline{w1}^h, \underline{O0}_a, \underline{r1}, ...)$
9	$\text{p}_\text{i}$dt $DRDF_0$	$<\underline{w0}_v^h[\underline{O1}_a]\underline{r0}_v/1_L/0>_{a \in BL(v)}$	$\Updownarrow(..., \underline{w0}^h, \underline{O1}_a, \underline{r0}, \underline{r0}, ...)$
10	$\text{p}_\text{i}$dt $DRDF_1$	$<\underline{w1}_v^h[\underline{O0}_a]\underline{r1}_v/0_L/1>_{a \in BL(v)}$	$\Updownarrow(..., \underline{w1}^h, \underline{O0}_a, \underline{r1}, \underline{r1}, ...)$
11	$\text{p}_\text{i}$dt RDF_0	$<\underline{w0}_v^h[\underline{O1}_a]\underline{r0}_v/1_L/1>_{a \in BL(v)}$	$\Updownarrow(..., \underline{w0}^h, \underline{O1}_a, \underline{r0}, ...)$
12	$\text{p}_\text{i}$dt RDF_1	$<\underline{w1}_v^h[\underline{O0}_a]\underline{r1}_v/0_L/0>_{a \in BL(v)}$	$\Updownarrow(..., \underline{w1}^h, \underline{O0}_a, \underline{r1}, ...)$

0 a multiple number of times ($0w0^h$) to the aggressor. The detection condition uses a so-called **nested march element** to detect the fault, where the indices i and j are used to control the progress of the performed operations. For every victim $v = i$, the march element performs an initializing $w0$ operation on i, then performs a sensitizing hammer $w0$ sequence on all possible aggressors $a = j$ in the memory, and finally detects the fault by performing a $r0$ on i. Needless to say, nested march elements are computationally expensive.

These expensive nested march elements result from a very strict interpretation of transient faults, where all initializing, sensitizing, and detecting operations must be performed directly after each other. It is possible to use a more relaxed interpretation of transient faults for two-cell faults, where the initialization is not required to be back-to-back with sensitization and detection, which results in less complex detection conditions. Here, however, we use the more strict interpretation, since it is capable of detecting all soft two-cell faults.

Tests for transient faults

Based on the detection conditions in Tables 8.7 and 8.8, it is possible to derive memory tests that detect all single-cell and two-cell transient faults. A march test that detects all single-cell transient faults can be represented by March 1CT below.

$$\begin{array}{llll} \text{March 1CT} = \{ & \underset{\text{ME0}}{\Updownarrow(\underline{w0}^h, \underline{w1}_a, \underline{r0}, \underline{r0})}; & \underset{\text{ME1}}{\Updownarrow(\underline{w1}^h, \underline{w0}_a, \underline{r1}, \underline{r1})}; & \underset{\text{ME2}}{\Updownarrow(\underline{w0}^h, \underline{w1}_a, \underline{w0}, \underline{r0})}; \\ & \underset{\text{ME3}}{\Updownarrow(\underline{w1}^h, \underline{w0}_a, \underline{w1}, \underline{r1})}; & \underset{\text{ME4}}{\Updownarrow(\underline{w0}^h, \underline{w1}, \underline{r1})}; & \underset{\text{ME5}}{\Updownarrow(\underline{w1}^h, \underline{w0}, \underline{r0})}\} \end{array}$$

Table 8.8. List of two-cell, transient FPs and their detection conditions.

#	Fault	$<S_a;S_v/F/R>$	Detection condition
1	t SF$_{0;0}$	$<0;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j,\underline{r0}_i,...),...)$
2	t SF$_{0;1}$	$<0;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j,\underline{r1}_i,...),...)$
3	t SF$_{1;0}$	$<1;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j,\underline{r0}_i,...),...)$
4	t SF$_{1;1}$	$<1;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j,\underline{r1}_i,...),...)$
5	p$_\mathrm{a}$t CFds$_{0w0;0}$	$<0\underline{w0}^h;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j^h,\underline{r0}_i,...),...)$
6	p$_\mathrm{a}$t CFds$_{0w0;1}$	$<0\underline{w0}^h;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j^h,\underline{r1}_i,...),...)$
7	p$_\mathrm{a}$t CFds$_{1w1;0}$	$<1\underline{w1}^h;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j^h,\underline{r0}_i,...),...)$
8	p$_\mathrm{a}$t CFds$_{1w1;1}$	$<1\underline{w1}^h;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j^h,\underline{r1}_i,...),...)$
9	p$_\mathrm{a}$t CFds$_{0w1;0}$	$<0\underline{w1}^h;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j,\underline{w1}_j^h,\underline{r0}_i,...),...)$
10	p$_\mathrm{a}$t CFds$_{0w1;1}$	$<0\underline{w1}^h;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j,\underline{w1}_j^h,\underline{r1}_i,...),...)$
11	p$_\mathrm{a}$t CFds$_{1w0;0}$	$<1\underline{w0}^h;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j,\underline{w0}_j^h,\underline{r0}_i,...),...)$
12	p$_\mathrm{a}$t CFds$_{1w0;1}$	$<1\underline{w0}^h;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j,\underline{w0}_j^h,\underline{r1}_i,...),...)$
13	p$_\mathrm{a}$t CFds$_{0r0;0}$	$<0\underline{r0}^h;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j,\underline{r0}_j^h,\underline{r0}_i,...),...)$
14	p$_\mathrm{a}$t CFds$_{0r0;1}$	$<0\underline{r0}^h;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j,\underline{r0}_j^h,\underline{r1}_i,...),...)$
15	p$_\mathrm{a}$t CFds$_{1r1;0}$	$<1\underline{r1}^h;0/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j,\underline{r1}_j^h,\underline{r0}_i,...),...)$
16	p$_\mathrm{a}$t CFds$_{1r1;1}$	$<1\underline{r1}^h;1/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j,\underline{r1}_j^h,\underline{r1}_i,...),...)$
17	p$_\mathrm{i}$t CFwd$_{0;0}$	$<\underline{w0}^h;0\underline{w0}/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j^h,\underline{w0}_i,\underline{r0}_i,...),...)$
18	p$_\mathrm{i}$t CFwd$_{1;0}$	$<\underline{w1}^h;0\underline{w0}/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j^h,\underline{w0}_i,\underline{r0}_i,...),...)$
19	p$_\mathrm{i}$t CFwd$_{0;1}$	$<\underline{w0}^h;1\underline{w1}/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j^h,\underline{w1}_i,\underline{r1}_i,...),...)$
20	p$_\mathrm{i}$t CFwd$_{1;1}$	$<\underline{w1}^h;1\underline{w1}/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j^h,\underline{w1}_i,\underline{r1}_i,...),...)$
21	p$_\mathrm{i}$t CFtr$_{0;1}$	$<\underline{w0}^h;1\underline{w0}/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j^h,\underline{w0}_i,\underline{r0}_i,...),...)$
22	p$_\mathrm{i}$t CFtr$_{1;1}$	$<\underline{w1}^h;1\underline{w0}/1_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j^h,\underline{w0}_i,\underline{r0}_i,...),...)$
23	p$_\mathrm{i}$t CFtr$_{1;0}$	$<\underline{w1}^h;0\underline{w1}/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j^h,\underline{w1}_i,\underline{r1}_i,...),...)$
24	p$_\mathrm{i}$t CFtr$_{0;0}$	$<\underline{w0}^h;0\underline{w1}/0_L/->$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j^h,\underline{w1}_i,\underline{r1}_i,...),...)$
25	p$_\mathrm{i}$t CFir$_{0;0}$	$<\underline{w0}^h;0\underline{r0}/0_L/1>$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j^h,\underline{r0}_i,...),...)$
26	p$_\mathrm{i}$t CFir$_{1;0}$	$<\underline{w1}^h;0\underline{r0}/0_L/1>$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j^h,\underline{r0}_i,...),...)$
27	p$_\mathrm{i}$t CFir$_{0;1}$	$<\underline{w0}^h;1\underline{r1}/1_L/0>$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j^h,\underline{r1}_i,...),...)$
28	p$_\mathrm{i}$t CFir$_{1;1}$	$<\underline{w1}^h;1\underline{r1}/1_L/0>$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j^h,\underline{r1}_i,...),...)$
29	p$_\mathrm{i}$t CFdr$_{0;0}$	$<\underline{w0}^h;0\underline{r0}/1_L/0>$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j^h,\underline{r0}_i,\underline{r0}_i,...),...)$
30	p$_\mathrm{i}$t CFdr$_{1;0}$	$<\underline{w1}^h;0\underline{r0}/1_L/0>$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j^h,\underline{r0}_i,\underline{r0}_i,...),...)$
31	p$_\mathrm{i}$t CFdr$_{0;1}$	$<\underline{w0}^h;1\underline{r1}/0_L/1>$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j^h,\underline{r1}_i,\underline{r1}_i,...),...)$
32	p$_\mathrm{i}$t CFdr$_{1;1}$	$<\underline{w1}^h;1\underline{r1}/0_L/1>$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j^h,\underline{r1}_i,\underline{r1}_i,...),...)$
33	p$_\mathrm{i}$t CFrd$_{0;0}$	$<\underline{w0}^h;0\underline{r0}/1_L/1>$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w0}_j^h,\underline{r0}_i,...),...)$
34	p$_\mathrm{i}$t CFrd$_{1;0}$	$<\underline{w1}^h;0\underline{r0}/1_L/1>$	$\Updownarrow_i(...,\Updownarrow_j(...,w0_i,\underline{w1}_j^h,\underline{r0}_i,...),...)$
35	p$_\mathrm{i}$t CFrd$_{0;1}$	$<\underline{w0}^h;1\underline{r1}/0_L/0>$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w0}_j^h,\underline{r1}_i,...),...)$
36	p$_\mathrm{i}$t CFrd$_{1;1}$	$<\underline{w1}^h;1\underline{r1}/0_L/0>$	$\Updownarrow_i(...,\Updownarrow_j(...,w1_i,\underline{w1}_j^h,\underline{r1}_i,...),...)$

This march test has six march elements (ME0 through ME5), each of which begins with a hammer write operation and ends with a detecting read operation. This test is identical to the test for hard single-cell DRAM faults (March 1CT = March 1CH), and therefore has the same complexity of $16\cdot n+6\cdot h\cdot n$. The reason the two tests are the same is due to the fact that March 1CT requires all operations that belong to a given detection condition to be performed back-to-back (since they are designed to detect transient faults). At the same time, this is also the way March 1CH is constructed, since back-to-back application of the test happens to be faster than other ways to detect the faulty behavior.

Table 8.9 lists all march elements in March 1CT along with the transient single-cell FPs, and indicates the first memory operation in the test that detects the corresponding FP. Again, the detection capabilities of March 1CT are identical to the detection capabilities of March 1CH. For example, ME0 (march element 0) shares an entry #1/3 with dt SF. This entry means that FP #1, which refers to the fault dt SF_0, is first detected in March 1CT by the 3rd operation of ME0. The table shows that each of ME0 and ME1 detect 4 different FPs, while each of ME2, ME3, ME4 and ME5 detect a single FP.

A march test that detects all two-cell, transient faults can be represented by March 2CT below.

$$
\boxed{
\begin{array}{llll}
\text{March 2CT} = \{\Updownarrow_i(& \underset{\text{ME0,0}}{\Updownarrow_j(w0_i,\underline{w0}_j^h,\underline{r0}_i,\underline{r0}_i)}, & \underset{\text{ME0,1}}{\Updownarrow_j(w0_i,\underline{w1}_j^h,\underline{r0}_i,\underline{r0}_i)}, & \underset{\text{ME0,2}}{\Updownarrow_j(w1_i,\underline{w0}_j^h,\underline{r1}_i,r1_i)}, \\
 & \underset{\text{ME0,3}}{\Updownarrow_j(w1_i,\underline{w1}_j^h,\underline{r1}_i,\underline{r1}_i)}, & \underset{\text{ME0,4}}{\Updownarrow_j(w0_i,\underline{w0}_j,\underline{w1}_j^h,\underline{r0}_i)}, & \underset{\text{ME0,5}}{\Updownarrow_j(w1_i,\underline{w0}_j,\underline{w1}_j^h,\underline{r1}_i)}, \\
 & \underset{\text{ME0,6}}{\Updownarrow_j(w0_i,\underline{w1}_j,\underline{w0}_j^h,\underline{r0}_i)}, & \underset{\text{ME0,7}}{\Updownarrow_j(w1_i,\underline{w1}_j,\underline{w0}_j^h,\underline{r1}_i)}, & \underset{\text{ME0,8}}{\Updownarrow_j(w0_i,\underline{w0}_j,\underline{r0}_j^h,\underline{r0}_i)}, \\
 & \underset{\text{ME0,9}}{\Updownarrow_j(w1_i,\underline{w0}_j,\underline{r0}_j^h,\underline{r1}_i)}, & \underset{\text{ME0,10}}{\Updownarrow_j(w0_i,\underline{w1}_j,\underline{r1}_j^h,\underline{r0}_i)}, & \underset{\text{ME0,11}}{\Updownarrow_j(w1_i,\underline{w1}_j,\underline{r1}_j^h,\underline{r1}_i)}, \\
 & \underset{\text{ME0,12}}{\Updownarrow_j(w0_i,\underline{w0}_j^h,\underline{w0}_i,\underline{r0}_i)}, & \underset{\text{ME0,13}}{\Updownarrow_j(w0_i,\underline{w1}_j^h,\underline{w0}_i,\underline{r0}_i)}, & \underset{\text{ME0,14}}{\Updownarrow_j(w1_i,\underline{w0}_j^h,\underline{w1}_i,\underline{r1}_i)}, \\
 & \underset{\text{ME0,15}}{\Updownarrow_j(w1_i,\underline{w1}_j^h,\underline{w1}_i,\underline{r1}_i)}, & \underset{\text{ME0,16}}{\Updownarrow_j(w1_i,\underline{w0}_j^h,\underline{w0}_i,\underline{r0}_i)}, & \underset{\text{ME0,17}}{\Updownarrow_j(w1_i,\underline{w1}_j^h,\underline{w0}_i,\underline{r0}_i)}, \\
 & \underset{\text{ME0,18}}{\Updownarrow_j(w0_i,\underline{w1}_j^h,\underline{w1}_i,\underline{r1}_i)}, & \underset{\text{ME0,19}}{\Updownarrow_j(w0_i,\underline{w0}_j^h,\underline{w1}_i,\underline{r1}_i)} &)\}
\end{array}
}
$$

This test has only one march element (ME0) that contains 20 nested march elements (ME0,0 through ME0,19). This test has a complexity of $60\cdot n^2+20\cdot h\cdot n^2$, which is of the order $O(n^2)$. This is a very complex test indeed, one of the most costly ever proposed. The reason behind the high computational complexity is the fact that it detects *transient* faults, for which the operations to initialize, sensitize and detect the faults have to be performed back-to-back, directly after each other. This makes march tests with linear complexity insufficient to detect the faults, and prevents using the possible parallelism present in the needed detection conditions.

This version of the test assumes that an aggressor can cause a fault in any victim anywhere in the memory. This assumption is, however, not realistic. The

Table 8.9. Detection capabilities of march elements in March 1CT.

Fault	dt SF	$p_i$dt WDF	$p_i$dt TF	$p_i$dt IRF	$p_i$dt DRDF	$p_i$dt RDF
ME0	#1/3	–	–	#7/3	#9/4	#11/3
ME1	#2/3	–	–	#8/3	#10/4	#12/4
ME2	–	#3/4	–	–	–	–
ME3	–	#4/4	–	–	–	–
ME4	–	–	#5/3	–	–	–
ME5	–	–	#6/3	–	–	–

impact of an aggressor is almost always limited to the adjacent neighboring cells. This observation can significantly simplify March 2CT, by limiting the value of j to a limited number of adjacent cells. Such a realistic two-cell march test (March 2CTr) is discussed later in Section 8.3.

Table 8.10 lists all march elements in March 2CT along with the transient two-cell FPs, and indicates the first memory operation in the test that detects the corresponding FP. The correspondence is apparent between the detection conditions in Table 8.8 and the individual nested march elements in March 2CT. With the exception of the first 4 nested march elements (each of which detects 4 different FPs), the elements from ME0,4 till ME0,19 each is dedicated to detect a single FP, and therefore corresponds to a single detection condition in Table 8.8. For example, ME0,0 (nested march element 0,0) shares an entry #1/3 with t SF. This entry means that FP #1, which refers to the fault t $SF_{0;0}$, is first detected in March 2CT by the 3rd operation of ME0,0.

8.2.3 Detecting soft faults

The idea of soft FPs implies that an operation can set a partly correct voltage into the cell, which then can gradually be depleted and cause a detectable fault in the cell after a period of time. In this section, we start by discussing the detection conditions needed to detect all soft faults, and then we generate memory tests for these soft faults.

Detection conditions for soft faults

In general terms, an FP has two components to describe a fault: F (the value of the faulty cell) and R (the output on a read operation). It is possible for F to change in such a way that a weak, proper state would be transformed into a faulty state within the cell. However, the value of R cannot be soft, since it is only important at the instant of reading and not later in time.

For example, the FP $<1w1/0/->$ represents a non-transition write 1 operation

Table 8.10. Detection capabilities of march elements in March 2CT.

ME	t SF	$\mathrm{p_a}$t CFds	$\mathrm{p_i}$t CFwd	$\mathrm{p_i}$t CFtr	$\mathrm{p_i}$t CFir	$\mathrm{p_i}$t CFdr	$\mathrm{p_i}$t CFrd
ME0,0	#1/3	#5/3	–	–	#25/3	#29/4	#33/3
ME0,1	#2/3	#6/3	–	–	#26/3	#30/4	#34/3
ME0,2	#3/3	#7/3	–	–	#27/3	#31/4	#35/3
ME0,3	#4/3	#8/3	–	–	#28/3	#32/4	#36/3
ME0,4	–	#9/4	–	–	–	–	–
ME0,5	–	#10/4	–	–	–	–	–
ME0,6	–	#11/4	–	–	–	–	–
ME0,7	–	#12/4	–	–	–	–	–
ME0,8	–	#13/4	–	–	–	–	–
ME0,9	–	#14/4	–	–	–	–	–
ME0,10	–	#15/4	–	–	–	–	–
ME0,11	–	#16/4	–	–	–	–	–
ME0,12	–	–	#17/4	–	–	–	–
ME0,13	–	–	#18/4	–	–	–	–
ME0,14	–	–	#19/4	–	–	–	–
ME0,15	–	–	#20/4	–	–	–	–
ME0,16	–	–	–	#21/4	–	–	–
ME0,17	–	–	–	#22/4	–	–	–
ME0,18	–	–	–	#23/4	–	–	–
ME0,19	–	–	–	#24/4	–	–	–

that sets a faulty 0 into the memory cell. Under the assumption that faults are constant in time, this FP should be sensitized the instant the write operation is performed. Under the assumption of soft FPs, however, this FP would only be sensitized after a specific amount of time T. This behavior is recorded by subscripting the last sensitizing operation with T, as in $<1w1_T/0/->$. The idle time T needs to have a value that is larger than the refresh time of the cell, which is typically around 64 ms. It should be noted here that this value is not fixed, but depends heavily on the technology of the memory and the operation temperature.

Table 8.11 lists all single-cell soft faults, along with the detection conditions needed to detect them. This table is easy to construct based on the detection conditions in Table 8.3, by introducing a delay time T after every sensitizing operation to allow enough time for the soft fault to get sensitized. Note that the detection conditions for soft IRFs and RDFs do not include the T, since these faults are detected as soon as they get sensitized by the read operations. The table lists both the partial, as well as the dirty types of soft faults. The detection conditions are designed to detect both faults. For example, the (partial, dirty and soft) write-0

Table 8.11. List of single-cell, soft FPs and their detection conditions.

#	Fault	$<S/F/R>$, $O \in \{w, r\}$	Detection condition, $O \in \{w, r\}$
1	ds SF$_0$	$<0_v[O1_a]_T/1/->_{a \in BL(v)}$	$\Updownarrow(...w0, ...O1a, ...T, ...r0, ...)$
2	ds SF$_1$	$<1_v[O0_a]_T/0/->_{a \in BL(v)}$	$\Updownarrow(...w1, ...O0a, ...T, ...r1, ...)$
3	p$_\mathrm{i}$ds WDF$_0$	$<w0^h_v[O1_a]w0_{vT}/1/->_{a \in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...w0, ...T, ...r0, ...)$
4	p$_\mathrm{i}$ds WDF$_1$	$<w1^h_v[O0_a]w1_{vT}/0/->_{a \in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...w1, ...T, ...r1, ...)$
5	p$_\mathrm{i}$ds TF$_1$	$<w0^h_v[O0_a]w1_{vT}/0/->_{a \in BL(v)}$	$\Updownarrow(...w0^h, ...O0_a, ...w1, ...T, ...r1, ...)$
6	p$_\mathrm{i}$ds TF$_0$	$<w1^h_v[O1_a]w0_{vT}/1/->_{a \in BL(v)}$	$\Updownarrow(...w1^h, ...O1_a, ...w0, ...T, ...r0, ...)$
7	p$_\mathrm{i}$ds IRF$_0$	$<w0^h_v[O1_a]r0_{vT}/0/1>_{a \in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...r0, ...)$
8	p$_\mathrm{i}$ds IRF$_1$	$<w1^h_v[O0_a]r1_{vT}/1/0>_{a \in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...r1, ...)$
9	p$_\mathrm{i}$ds DRDF$_0$	$<w0^h_v[O1_a]r0_{vT}/1/0>_{a \in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...r0, ...T, ...r0, ...)$
10	p$_\mathrm{i}$ds DRDF$_1$	$<w1^h_v[O0_a]r1_{vT}/0/1>_{a \in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...r1, ...T, ...r1, ...)$
11	p$_\mathrm{i}$ds RDF$_0$	$<w0^h_v[O1_a]r0_{vT}/1/1>_{a \in BL(v)}$	$\Updownarrow(...w0^h, ...O1_a, ...r0, ...)$
12	p$_\mathrm{i}$ds RDF$_1$	$<w1^h_v[O0_a]r1_{vT}/0/0>_{a \in BL(v)}$	$\Updownarrow(...w1^h, ...O0_a, ...r1, ...)$

destructive fault (p$_\mathrm{i}$ds WDF$_0$), must first be initialized a multiple number of times ($w0^h$). Then, a completing operation with data 1 must be applied to a different cell along the same BL ($[O1_a]$), before the sensitizing write 0 operation can be performed ($w0$). To ensure the detection of this soft fault, a delay time T must be introduced after the sensitizing $w0$ operation to allow for sensitization to take place. The detection condition is identical to that used for the hard WDF$_0$, apart from the introduction of a delay T after the sensitizing $w0$ operation.

Table 8.12 lists all two-cell soft faults, along with the detection conditions needed to detect them. This table is constructed from Table 8.4, by introducing a delay T after every sensitizing operation in each detection condition. Note, though, that the detection conditions for soft CFir and CFrd do not include a delay T, since these faults are detected as soon as they get sensitized by their respective read operations. When operations need to be performed on both the aggressor *and* the victim to sensitize the corresponding fault, then we assume that these operations can be performed in either order, starting with the aggressor then the victim, or vice versa. The table takes both soft and partial faults into consideration, and lists detection conditions designed to detect both faults. For example, the (partial, soft) $0w0$ disturb 0 coupling fault (p$_\mathrm{a}$s CFds$_{0w0;0}$), must first be sensitized by writing 0 a multiple number of times ($0w0^h$) to all cells in the memory, followed by a waiting period of T, and finally detected by a subsequent read operation. The detection condition for this fault is identical to that of the corresponding hard fault (p$_\mathrm{a}$h CFds$_{0w0;0}$), with the exception of the delay T added after the $w0^h$ to ensure sensitizing the soft fault.

Table 8.12. List of two-cell, soft FPs and their detection conditions.

# Fault	$<S_a;S_v/F/R>$	Detection condition	
		$a<v$	$a>v$
1 s SF$_{0;0}$	$<0;0_T/1/->$	$\Updownarrow(\ldots0);T;\Updownarrow(\ldots,r0,\ldots)$	
2 s SF$_{0;1}$	$<0;1_T/0/->$	$\Uparrow(r1,\ldots w0,T,\ldots)$	$\Downarrow(r1,\ldots w0,T,\ldots)$
3 s SF$_{1;0}$	$<1;0_T/1/->$	$\Uparrow(r0,\ldots w1,T,\ldots)$	$\Downarrow(r0,\ldots w1,T,\ldots)$
4 s SF$_{1;1}$	$<1;1_T/0/->$	$\Updownarrow(\ldots1);T;\Updownarrow(\ldots,r1,\ldots)$	
5 p$_\mathrm{a}$s CFds$_{0w0;0}$	$<0w0^h;0_T/1/->$	$\Downarrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$	$\Uparrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$
6 p$_\mathrm{a}$s CFds$_{0w0;1}$	$<0w0^h;1_T/0/->$	$\Uparrow(r1,\ldots,w0^h,T,\ldots)$	$\Downarrow(r1,\ldots,w0^h,T,\ldots)$
7 p$_\mathrm{a}$s CFds$_{1w1;0}$	$<1w1^h;0_T/1/->$	$\Uparrow(r0,\ldots,w1^h,T,\ldots)$	$\Downarrow(r0,\ldots,w1^h,T,\ldots)$
8 p$_\mathrm{a}$s CFds$_{1w1;1}$	$<1w1^h;1_T/0/->$	$\Downarrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$	$\Uparrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$
9 p$_\mathrm{a}$s CFds$_{0w1;0}$	$<0w1^h;0_T/1/->$	$\Uparrow(r0,\ldots,w1^h,T,\ldots)$	$\Downarrow(r0,\ldots,w1^h,\ldots)$
10 p$_\mathrm{a}$s CFds$_{0w1;1}$	$<0w1^h;1_T/0/->$	$\Updownarrow(\ldots0);\Downarrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$	$\Updownarrow(\ldots0);\Uparrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$
11 p$_\mathrm{a}$s CFds$_{1w0;0}$	$<1w0^h;0_T/1/->$	$\Updownarrow(\ldots1);\Downarrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$	$\Updownarrow(\ldots1);\Uparrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$
12 p$_\mathrm{a}$s CFds$_{1w0;1}$	$<1w0^h;1_T/0/->$	$\Uparrow(r1,\ldots,w0^h,T,\ldots)$	$\Downarrow(r1,\ldots,w0^h,T,\ldots)$
13 p$_\mathrm{a}$s CFds$_{0r0;0}$	$<0r0^h;0_T/1/->$	$\Uparrow(r0^h,T,\ldots)$	$\Downarrow(r0^h,T,\ldots)$
14 p$_\mathrm{a}$s CFds$_{0r0;1}$	$<0r0^h;1_T/0/->$	$\Downarrow(r0^h,\ldots1);T;\Updownarrow(r1\ldots)$	$\Uparrow(r0^h,\ldots1);T;\Updownarrow(r1\ldots)$
15 p$_\mathrm{a}$s CFds$_{1r1;0}$	$<1r1^h;0_T/1/->$	$\Downarrow(r1^h,\ldots0);T;\Updownarrow(r0\ldots)$	$\Uparrow(r1^h,\ldots0);T;\Updownarrow(r0\ldots)$
16 p$_\mathrm{a}$s CFds$_{1r1;1}$	$<1r1^h;1_T/0/->$	$\Uparrow(r1^h,T,\ldots)$	$\Downarrow(r1^h,T,\ldots)$
17 p$_\mathrm{i}$s CFwd$_{0;0}$	$<w0^h;0w0_T/1/->$	$\Uparrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$	$\Downarrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$
18 p$_\mathrm{i}$s CFwd$_{1;0}$	$<w1^h;0w0_T/1/->$	$a<v$: $\Updownarrow(\ldots w1^h);\Downarrow(\ldots0,w0);T;\Updownarrow(r0\ldots)$ $a>v$: $\Updownarrow(\ldots w1^h);\Uparrow(\ldots0,w0);T;\Updownarrow(r0\ldots)$	
19 p$_\mathrm{i}$s CFwd$_{0;1}$	$<w0^h;1w1_T/0/->$	$a<v$: $\Updownarrow(\ldots w0^h);\Downarrow(\ldots1,w1);T;\Updownarrow(r1\ldots)$ $a>v$: $\Updownarrow(\ldots w0^h);\Uparrow(\ldots1,w1);T;\Updownarrow(r1\ldots)$	
20 p$_\mathrm{i}$s CFwd$_{1;1}$	$<w1^h;1w1_T/0/->$	$\Uparrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$	$\Downarrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$
21 p$_\mathrm{i}$s CFtr$_{0;0}$	$<w0^h;1w0_T/1/->$	$\Updownarrow(\ldots1);\Downarrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$	$\Updownarrow(\ldots1);\Uparrow(\ldots w0^h);T;\Updownarrow(r0\ldots)$
22 p$_\mathrm{i}$s CFtr$_{1;0}$	$<w1^h;1w0_T/1/->$	$\Updownarrow(\ldots w1^h);\Downarrow(\ldots w0);T;\Updownarrow(r0\ldots)$	$\Updownarrow(\ldots w1^h);\Uparrow(\ldots w0);T;\Updownarrow(r0\ldots)$
23 p$_\mathrm{i}$s CFtr$_{0;1}$	$<w0^h;0w1_T/0/->$	$\Updownarrow(\ldots w0^h);\Downarrow(\ldots w1);T;\Updownarrow(r1\ldots)$	$\Updownarrow(\ldots w0^h);\Uparrow(\ldots w1);T;\Updownarrow(r1\ldots)$
24 p$_\mathrm{i}$s CFtr$_{1;1}$	$<w1^h;0w1_T/0/->$	$\Updownarrow(\ldots0);\Downarrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$	$\Updownarrow(\ldots0);\Uparrow(\ldots w1^h);T;\Updownarrow(r1\ldots)$
25 p$_\mathrm{i}$s CFir$_{0;0}$	$<w0^h;0r0_T/0/1>$	$\Updownarrow(\ldots,w0^h);\Updownarrow(\ldots,r0,\ldots)$	
26 p$_\mathrm{i}$s CFir$_{1;0}$	$<w1^h;0r0_T/0/1>$	$\Uparrow(r0,\ldots,w1^h)$	$\Downarrow(r0,\ldots,w1^h)$
27 p$_\mathrm{i}$s CFir$_{0;1}$	$<w0^h;1r1_T/1/0>$	$\Uparrow(r1,\ldots,w0^h)$	$\Downarrow(r1,\ldots,w0^h)$
28 p$_\mathrm{i}$s CFir$_{1;1}$	$<w1^h;1r1_T/1/0>$	$\Updownarrow(\ldots,w1^h);\Updownarrow(\ldots,r1,\ldots)$	
29 p$_\mathrm{i}$s CFdr$_{0;0}$	$<w0^h;0r0_T/1/0>$	$\Updownarrow(\ldots,w0^h);\Updownarrow(\ldots,r0,T,r0,\ldots)$	
30 p$_\mathrm{i}$s CFdr$_{1;0}$	$<w1^h;0r0_T/1/0>$	$\Uparrow(r0,T,r0,\ldots,w1^h)$	$\Downarrow(r0,T,r0,\ldots,w1^h)$
31 p$_\mathrm{i}$s CFdr$_{0;1}$	$<w0^h;1r1_T/0/1>$	$\Uparrow(r1,T,r1,\ldots,w0^h)$	$\Downarrow(r1,T,r1,\ldots,w0^h)$
32 p$_\mathrm{i}$s CFdr$_{1;1}$	$<w1^h;1r1_T/0/1>$	$\Updownarrow(\ldots,w1^h);\Updownarrow(\ldots,r1,T,r1,\ldots)$	
33 p$_\mathrm{i}$s CFrd$_{0;0}$	$<w0^h;0r0_T/1/1>$	$\Updownarrow(\ldots,w0^h);\Updownarrow(\ldots,r0,\ldots)$	
34 p$_\mathrm{i}$s CFrd$_{1;0}$	$<w1^h;0r0_T/1/1>$	$\Uparrow(r0,\ldots,w1^h)$	$\Downarrow(r0,\ldots,w1^h)$
35 p$_\mathrm{i}$s CFrd$_{0;1}$	$<w0^h;1r1_T/0/0>$	$\Uparrow(r1,\ldots,w0^h)$	$\Downarrow(r1,\ldots,w0^h)$
36 p$_\mathrm{i}$s CFrd$_{1;1}$	$<w1^h;1r1_T/0/0>$	$\Updownarrow(\ldots,w1^h);\Updownarrow(\ldots,r1,\ldots)$	

Tests for soft faults

Based on the detection conditions in Tables 8.11 and 8.12, it is possible to derive memory tests that detect all single-cell and two-cell soft faults. A march test that detects all single-cell soft faults can have the form of March 1CS below.

$$\begin{array}{lccc} \text{March 1CS} = \{ & \Updownarrow(w0^h,\, w1_a,\, r0,\, T,\, r0); & \Updownarrow(w1^h,\, w0_a,\, r1,\, T,\, r1); & \Updownarrow(w0^h,\, w1_a,\, w0,\, T,\, r0); \\ & \text{ME0} & \text{ME1} & \text{ME2} \\ & \Updownarrow(w1^h,\, w0_a,\, w1,\, T,\, r1); & \Updownarrow(w0^h,\, w1,\, T,\, r1); & \Updownarrow(w1^h,\, w0,\, T,\, r0)\} \\ & \text{ME3} & \text{ME4} & \text{ME5} \end{array}$$

This march test has six march elements (ME0 through ME5), each of which begins with a hammer write operation and ends with a detecting read operation. This test is similar to the tests for hard single-cell DRAM faults (March 1CH), which is expected since the space of soft faults is derived from the space of hard faults. This march test substitutes the dirty operation (O) in the detection conditions of Table 8.11 by a write operation, since this choice reduces the length of the test. The test has a complexity of $16 \cdot n + 6 \cdot h \cdot n + 6 \cdot T \cdot n$, which is higher than 1CH by $6 \cdot T \cdot n$. For a typical idle time of $T > 64$ ms, the total idle test time becomes rather long relative to the total length of the test (assuming $h \approx 5$). In order to reduce test time, it is important to implement a number of test time reduction methods, such as *design-for-testability* (*DFT*) techniques or the application of test stresses that force soft faults to become directly detectable hard faults, which in turn do not require any delay time to detect. The theory behind these methods is discussed in Section 4.4.1, while an example of applying these methods to a specific test problem is discussed in Section 9.4.

Table 8.13 lists all march elements in March 1CS along with the soft single-cell FPs, and indicates the first memory operation in the test that detects the corresponding FP. For example, ME0 (march element 0) shares an entry #1/5 with ds SF. This entry means that FP #1, which refers to the fault ds SF_0, is first detected in March 1CS by the 5th operation of ME0. The table shows that each of ME0 and ME1 detect 4 different FPs, while each of ME2, ME3, ME4 and ME5 detect a single FP.

A march test that detects all two-cell soft faults can be represented by March 2CS below.

$$\begin{array}{lcccc} \text{March 2CS} = \{ & \Updownarrow(w0^h); & \Uparrow(r0^h, T, r0, w1^h); & \Uparrow(r1^h, T, r1, w0^h); & \\ & \text{ME0} & \text{ME1} & \text{ME2} & \\ & \Downarrow(r0^h, T, r0, w1^h); & \Downarrow(r1^h, T, r1, w0^h); & T; & \Updownarrow(r0)\} \\ & \text{ME3} & \text{ME4} & \text{ME5} & \text{ME6} \end{array}$$

This march test has 7 march elements (ME0 through ME6). It is based on March 2CH for hard faults, with a number of added delays T to a number of march elements in the test (sometimes separating one read operation from a hammer read

Table 8.13. Detection capabilities of march elements in March 1CS.

Fault	ds SF	$\mathrm{p_i}$ds WDF	$\mathrm{p_i}$ds TF	$\mathrm{p_i}$ds IRF	$\mathrm{p_i}$ds DRDF	$\mathrm{p_i}$ds RDF
ME0	#1/5	–	–	#7/3	#9/5	#11/3
ME1	#2/5	–	–	#8/3	#10/5	#12/3
ME2	–	#3/5	–	–	–	–
ME3	–	#4/5	–	–	–	–
ME4	–	–	#5/4	–	–	–
ME5	–	–	#6/4	–	–	–

sequence). Note that ME5 is simply a single delay T added to sensitize the faults before the final detecting read operations are performed in ME6. This test has a time complexity of $5 \cdot n + 9 \cdot h \cdot n + (4 \cdot n + 1)T$. The delay time introduced by T takes a relatively long period to perform, which makes this test very costly. In order to reduce test time, it is important to implement a number of test time reduction methods, such as test stresses, which force soft faults to become directly detectable hard faults. An example of applying stresses to a specific test problem is discussed in Section 9.4.

Table 8.14 lists all march elements in March 2CS along with the soft two-cell FPs, and indicates the first memory operation in the test that detects the corresponding FP. There are two columns for each fault, one for $a < v$ and the other $a > v$. The table is very similar to Table 8.6 for hard faults. For example, ME1 shares an entry #1/1 with s SF. This entry means that FP #1, which refers to the fault h $\mathrm{SF}_{0;0}$, is first detected in March 2CS by the read operations in ME1.

8.3 Customizing march tests

The march tests listed in Section 8.2 are general memory tests designed to detect the targeted DRAM-specific faults in any DRAM. It is possible, however, to customize these march tests based on the specific design or implementation of the DRAM under test. This customization can be done for a number of reasons, such as reducing the complexity of some tests, or to extend their detection capability to detect more faults associated with special DRAM designs. In this section, we first customize the above tests for specific electrical DRAM designs, then we show how to reduce the complexity of March 2CT based on layout information of the memory.

8.3.1 Memory design considerations

The memory tests discussed in the previous section are generated under the assumption that we have no information about the internal design of the memory. In this section, we suggest a number of different internal memory design alternatives

Table 8.14. Detection capabilities of march elements in March 2CS.

<table>
<tr><th rowspan="2">ME</th><th colspan="2">s SF</th><th colspan="2">p_as CFds</th><th colspan="2">p_is CFwd</th><th colspan="2">p_is CFtr</th><th colspan="2">p_is CFir</th><th colspan="2">p_is CFdr</th><th colspan="2">p_is CFrd</th></tr>
<tr><th>$a < v$</th><th>$a > v$</th><th>$a < v$</th><th>$a > v$</th><th>$a < v$</th><th>$a > v$</th><th>$a < v$</th><th>$a > v$</th><th>$a < v$</th><th>$a > v$</th><th>$a < v$</th><th>$a > v$</th><th>$a < v$</th><th>$a > v$</th></tr>
<tr><td>ME0</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td rowspan="3">ME1</td><td colspan="2">#1</td><td>#7</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>#26</td><td>–</td><td>#30</td><td>–</td><td>#34</td><td>–</td></tr>
<tr><td>#3</td><td>–</td><td>#9</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td>#13</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td rowspan="3">ME2</td><td>#2</td><td>–</td><td>#6</td><td>#8</td><td>#20</td><td>#19</td><td>–</td><td>#23</td><td>#27</td><td>–</td><td>#31</td><td>–</td><td>#35</td><td>–</td></tr>
<tr><td colspan="2">#4</td><td>#12</td><td>#10</td><td>–</td><td>–</td><td>–</td><td>#24</td><td colspan="2">#28</td><td colspan="2">#32</td><td colspan="2">#36</td></tr>
<tr><td>–</td><td>–</td><td>#16</td><td>#14</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td rowspan="5">ME3</td><td>–</td><td>#3</td><td>–</td><td>#7</td><td>#17</td><td>#18</td><td>–</td><td>#21</td><td colspan="2">#25</td><td colspan="2">#29</td><td colspan="2">#33</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>#9</td><td>–</td><td>–</td><td>–</td><td>#22</td><td>–</td><td>#26</td><td>–</td><td>#30</td><td>–</td><td>#34</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>#11</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>#13</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td>–</td><td>#15</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td rowspan="3">ME4</td><td>–</td><td>#2</td><td>#8</td><td>#6</td><td>#19</td><td>#20</td><td>#23</td><td>–</td><td>–</td><td>#27</td><td>–</td><td>#31</td><td>–</td><td>#35</td></tr>
<tr><td>–</td><td>–</td><td>#10</td><td>#12</td><td>–</td><td>–</td><td>#24</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td>#14</td><td>#16</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>ME5</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td rowspan="3">ME6</td><td>–</td><td>–</td><td>#5</td><td>–</td><td>#18</td><td>#17</td><td>#21</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td>#11</td><td>–</td><td>–</td><td>–</td><td>#22</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
<tr><td>–</td><td>–</td><td>#15</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
</table>

that exhibit limited types of the fully possible range of faulty behavior. This reduces the complexity of the memory tests needed to detect the faulty behavior of the memory.

Available memory design alternatives

The discussion in Chapter 7 indicates that the faulty behavior of the memory simulated in that chapter does not exhibit any type of dirty faults. The reason for this is the fact that the write and precharge circuitry is located on the same side of each BL pair [see Figure 7.1]. The examples of dirty faults discussed in Section 8.1.2 above show that in order for partial faults to take place the precharge circuitry should be located on one side of the BL pair, while the write drivers and the sense amplifiers located on the other side [see Figure 8.1(c)].

Table 8.15. Different positions of the SA, WD and PR on either side of a BL pair.

<table>
<tr><th rowspan="2">#</th><th colspan="3">Upside</th><th colspan="3">Downside</th><th rowspan="2">Description</th></tr>
<tr><th>SA</th><th>WD</th><th>PR</th><th>SA</th><th>WD</th><th>PR</th></tr>
<tr><td>1</td><td>t</td><td>t</td><td>t</td><td>b</td><td>b</td><td>b</td><td>SA, WD and PR all on the same side of BL</td></tr>
<tr><td>2</td><td>t</td><td>t</td><td>b</td><td>b</td><td>b</td><td>t</td><td>SA and WD on one side, PR on the other</td></tr>
<tr><td>3</td><td>t</td><td>b</td><td>t</td><td>b</td><td>t</td><td>b</td><td>SA and PR on one side, WD on the other</td></tr>
<tr><td>4</td><td>t</td><td>b</td><td>b</td><td>b</td><td>t</td><td>t</td><td>SA on one side, WD and PR on the other</td></tr>
</table>

Table 8.15 lists the 8 different combinations the sense amplifiers (SA), the write drivers (WD) and the precharge circuits (PR) can be positioned with respect to each other on either side of a BL pair. An entry "t" means that the corresponding circuitry is located at the top of the BL pair, while an entry "b" means that the corresponding circuitry is located at the bottom of the BL pair. Since the top and the bottom of a BL pair are symmetrical, only 4 combinations in Table 8.15 are unique, while the other 4 combinations can be derived by viewing the BL pair from the upside or from the downside.

For example, Figure 8.2 shows one possible position allocation of the SA, WD and PR circuitry. In this configuration, the SA and WD are located on one side of the BL pair and the PR is located on the other side. Considering the upside of the BL pair, the SA and the WD are located at the top while the PR is located at the bottom. Considering the downside of the BL pair, the PR is located at the top while the SA and the WD are located at the bottom. Obviously, both of these BL organizations are identical.

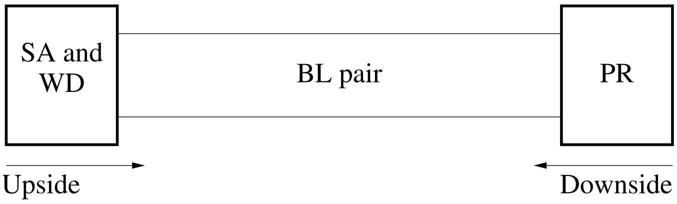


Figure 8.2. Upside and downside symmetry of BL pairs.

Despite the fact that there are 4 different unique combinations of SA, WD and PR listed in Table 8.15, in practice only two of these four combinations are used in real designs. BL pairs are usually designed such that the SA is located in the neighborhood of the WD, since this makes it easier for the WD to set the proper value into the SA during a write operation. Therefore, the only valid practical combinations in Table 8.15 are combination #1 and combination #2, where the SA and WD are located at the same side of the BL pair. The implications of these two combinations on the required testing is discussed next.

Impact of design on faults

We start this discussion with combination #2 in Table 8.15, since it is the easier one to discuss. This combination corresponds to the situation when the SA and WD circuits are located at one end of the BL pair, while the PR circuits are located at the other end of the BL pair. This organization is the one we keep referring to when discussing dirty faults [see Figure 8.1(c) and the associated discussion]. Apart from being responsible for causing dirty faults when some specific defects are present, this organization does not prevent any type of fault from taking place. As a result, this BL pair organization is capable of exhibiting the full space of memory faults as described in Expression 8.3. This means that in order to perform a full test for the faulty behavior of this memory design, all memory tests discussed in the Section 8.2 (1CH, 2CH, 1CT, 2CT, 1CS and 2CS) should be applied to the memory.

The second situation corresponds to combination #1 in Table 8.15, where all three circuits SA, WD and PR are located at the same side of the BL pair. This situation corresponds to the electrical memory model shown in Figure 7.1, which has been simulated and analyzed in detail in Chapter 7. According to the analysis in that chapter, this BL pair organization prevents the memory from exhibiting any types of dirty faults. In other words, this organization reduces the full space of memory faults exhibited as a result of all possible defects from the one described in Expression 8.3 to the following one.

$$\text{Single-cell fault} = \left\{ \begin{array}{c} \text{-} \\ \text{p}_\text{i} \end{array} \right\} \left\{ \begin{array}{c} \text{h} \\ \text{s} \\ \text{t} \end{array} \right\} \text{FP}, \text{Two-cell fault} = \left\{ \begin{array}{c} \text{-} \\ \text{p} \end{array} \right\} \left\{ \begin{array}{c} \text{h} \\ \text{s} \\ \text{t} \end{array} \right\} \text{FP} \qquad (8.4)$$

Note that this reduction in the space of possible faults influences only single-cell faults, since they are the only faults that can be dirty. As a result, the space of two-cell faults remains identical to the general one. Consequently, in order to test for all possible faults caused by this design, all two-cell tests described in Section 8.2 (2CH, 2CT and 2CS) should be applied to the memory. The tests for single-cell faults, however, can be reduced in complexity as dirty faults do not need to be tested for in this organization.

Impact of design on testing hard faults

Table 8.16 lists all single-cell hard faults, along with the detection conditions needed to detect them [compare with Table 8.3]. The table considers the reduced form of single-cell hard faults, where only partial faults take place, but not dirty faults. For example, the (partial, hard, but not dirty) write-0 destructive fault (p_ih WDF_0), must be initialized and sensitized with a $w0$ a multiple number of times ($w0^h$) in order for the fault to be sensitized. The detection condition starts with multiple $w0$ operations to initialize and sensitize the cell to 0, then the read operation ensures the detection of the fault.

A test that detects the single-cell hard faults for the BL pair configuration #1 in Table 8.15 is the following.

$$\boxed{\begin{array}{lll} \text{March 1CH}_{swp} = \{ & \underset{\text{ME0}}{\Updownarrow(w0^h,\, r0,\, r0);} & \underset{\text{ME1}}{\Updownarrow(w1^h,\, r1,\, r1);} \\ & \underset{\text{ME2}}{\Updownarrow(w0^h,\, w1,\, r1);} & \underset{\text{ME3}}{\Updownarrow(w1^h,\, w0,\, r0)\}} \end{array}}$$

Table 8.16. Single-cell, hard (but not dirty) FPs for BL pair configuration #1 in Table 8.15.

#	Fault	$<S/F/R>$	Detection condition
1	h SF$_0$	$<0/1/->$	$\Updownarrow(...r0,...)$
2	h SF$_1$	$<1/0/->$	$\Updownarrow(...r1,...)$
3	p$_\text{i}$h WDF$_0$	$<w0^h/1/->$	$\Updownarrow(...w0^h,...r0,...)$
4	p$_\text{i}$h WDF$_1$	$<w1^h/0/->$	$\Updownarrow(...w1^h,...r1,...)$
5	p$_\text{i}$h TF$_1$	$<w0^h w1/0/->$	$\Updownarrow(...w0^h,...w1,...r1,...)$
6	p$_\text{i}$h TF$_0$	$<w1^h w0/1/->$	$\Updownarrow(...w1^h,...w0,...r0,...)$
7	p$_\text{i}$h IRF$_0$	$<w0^h r0/0/1>$	$\Updownarrow(...w0^h,...r0,...)$
8	p$_\text{i}$h IRF$_1$	$<w1^h r1/1/0>$	$\Updownarrow(...w1^h,...r1,...)$
9	p$_\text{i}$h DRDF$_0$	$<w0^h r0/1/0>$	$\Updownarrow(...w0^h,...r0,...r0,...)$
10	p$_\text{i}$h DRDF$_1$	$<w1^h r1/0/1>$	$\Updownarrow(...w1^h,...r1,...r1,...)$
11	p$_\text{i}$h RDF$_0$	$<w0^h r0/1/1>$	$\Updownarrow(...w0^h,...r0,...)$
12	p$_\text{i}$h RDF$_1$	$<w1^h r1/0/0>$	$\Updownarrow(...w1^h,...r1,...)$

This march test has four march elements (ME0 through ME3), each of which begins with a hammer write operation and ends with a detecting read operation. Each two consecutive march elements represent the exact complement of each other, as they are generated to target complementary FPs. The first two march elements (ME0 and ME1) are designed to detect all single-cell FPs except TF$_0$ and TF$_1$, whereas ME2 is designed to detect TF$_1$ and ME3 is designed to detect TF$_0$. The test has a complexity of $(8 \cdot n + 4 \cdot h \cdot n)$, which is lower than the complexity of 1CH that can also detect dirty single-cell faults.

Impact of design on testing transient faults

Table 8.17 lists all single-cell transient faults, along with their detection conditions [compare with Table 8.16]. For example, the (partial, transient, but not dirty) write-0 destructive fault (p$_\text{i}$t WDF$_0$), must first be initialized and sensitized a multiple number of times ($w0^h$). To ensure the detection of this transient fault, all of these operations must be applied directly after each other (back-to-back), and directly followed by a detecting read operation. The detection condition starts with multiple $w0$ operations to initialize and sensitize the cell to 0, directly followed by a detecting $r0$ operation. The fact that the operations must be performed back-to-back is indicated by the underscores below the corresponding operations.

A march test March 1CT$_{swp}$ can be proposed to detect all the faults described in Table 8.17. Such a test can have the following form.

Table 8.17. Single-cell, transient (but not dirty) FPs for the BL configuration #1 in Table 8.15.

#	Fault	$<S/F_L/R>$	Detection condition
1	t SF$_0$	$<0/1_L/->$	$\Updownarrow(..., w0, \underline{r0}, ...)$
2	t SF$_1$	$<1/0_L/->$	$\Updownarrow(..., w1, \underline{r1}, ...)$
3	p_it WDF$_0$	$<\underline{w0}^h\underline{w0}/1_L/->$	$\Updownarrow(..., \underline{w0}^h, \underline{r0}, ...)$
4	p_it WDF$_1$	$<\underline{w1}^h\underline{w1}/0_L/->$	$\Updownarrow(..., \underline{w1}^h, \underline{r1}, ...)$
5	p_it TF$_1$	$<\underline{w0}^h\underline{w1}/0_L/->$	$\Updownarrow(..., \underline{w0}^h, \underline{w1}, \underline{r1}, ...)$
6	p_it TF$_0$	$<\underline{w1}^h\underline{w0}/1_L/->$	$\Updownarrow(..., \underline{w1}^h, \underline{w0}, \underline{r0}, ...)$
7	p_it IRF$_0$	$<\underline{w0}^h\underline{r0}/0_L/1>$	$\Updownarrow(..., \underline{w0}^h, \underline{r0}, ...)$
8	p_it IRF$_1$	$<\underline{w1}^h\underline{r1}/1_L/0>$	$\Updownarrow(..., \underline{w1}^h, \underline{r1}, ...)$
9	p_it DRDF$_0$	$<\underline{w0}^h\underline{r0}/1_L/0>$	$\Updownarrow(..., \underline{w0}^h, \underline{r0}, \underline{r0}, ...)$
10	p_it DRDF$_1$	$<\underline{w1}^h\underline{r1}/0_L/1>$	$\Updownarrow(..., \underline{w1}^h, \underline{r1}, \underline{r1}, ...)$
11	p_it RDF$_0$	$<\underline{w0}^h\underline{r0}/1_L/1>$	$\Updownarrow(..., \underline{w0}^h, \underline{r0}, ...)$
12	p_it RDF$_1$	$<\underline{w1}^h\underline{r1}/0_L/0>$	$\Updownarrow(..., \underline{w1}^h, \underline{r1}, ...)$

$$\begin{array}{lcc} \text{March 1CT}_{swp} = \text{March 1CH}_{swp} = \{ & \Updownarrow(w0^h,\ r0,\ r0); & \Updownarrow(w1^h,\ r1,\ r1); \\ & \text{ME0} & \text{ME1} \\ & \Updownarrow(w0^h,\ w1,\ r1); & \Updownarrow(w1^h,\ w0,\ r0)\} \\ & \text{ME2} & \text{ME3} \end{array}$$

This test is identical to March 1CH$_{swp}$ described above in this section (1CT$_{swp}$ = 1CH$_{swp}$). This is unsurprising, since the same situation is true for the general tests generated in Section 8.2, where the general test generated to detect all single-cell transient FPs (March 1CT) is identical to the general test generated to detect all single-cell hard FPs (March 1CH).

Impact of design on testing soft faults

Table 8.18 lists all single-cell soft (but not dirty) faults, along with the detection conditions needed to detect them. This table is easy to construct based on the detection conditions in Table 8.16, by introducing a delay time T after every sensitizing operation to allow enough time for the soft fault to get sensitized. Note that the detection conditions for soft IRFs and RDFs do not include the T, since these faults are detected as soon as they get sensitized by the read operations. For example, the (partial, soft but not dirty) write-0 destructive fault (p_is WDF$_0$), must first be initialized and sensitized a multiple number of times ($w0^h$). To ensure the detection of this soft fault, a delay time T must be introduced after the last $w0$ operation to allow for sensitization to take place. The detection condition is identical to that used for the hard WDF_0, apart from the introduction of a delay T after the sensitizing $w0$ operation.

Table 8.18. List of single-cell, soft (but not dirty) FPs for the BL configuration #1 in Table 8.15.

#	Fault	$<S/F/R>$	Detection condition
1	s SF_0	$<0_T/1/->$	$\Updownarrow(...w0, ...T, ...r0, ...)$
2	s SF_1	$<1_T/0/->$	$\Updownarrow(...w1, ...T, ...r1, ...)$
3	$\mathrm{p_i}$s WDF_0	$<w0^h_T/1/->$	$\Updownarrow(...w0^h, ...T, ...r0, ...)$
4	$\mathrm{p_i}$s WDF_1	$<w1^h_T/0/->$	$\Updownarrow(...w1^h, ...T, ...r1, ...)$
5	$\mathrm{p_i}$s TF_1	$<w0^h w1_T/0/->$	$\Updownarrow(...w0^h, ...w1, ...T, ...r1, ...)$
6	$\mathrm{p_i}$s TF_0	$<w1^h w0_T/1/->$	$\Updownarrow(...w1^h, ...w0, ...T, ...r0, ...)$
7	$\mathrm{p_i}$s IRF_0	$<w0^h r0_T/0/1>$	$\Updownarrow(...w0^h, ...r0, ...)$
8	$\mathrm{p_i}$s IRF_1	$<w1^h r1_T/1/0>$	$\Updownarrow(...w1^h, ...r1, ...)$
9	$\mathrm{p_i}$s DRDF_0	$<w0^h r0_T/1/0>$	$\Updownarrow(...w0^h, ...r0, ...T, ...r0, ...)$
10	$\mathrm{p_i}$s DRDF_1	$<w1^h r1_T/0/1>$	$\Updownarrow(...w1^h, ...r1, ...T, ...r1, ...)$
11	$\mathrm{p_i}$s RDF_0	$<w0^h r0_T/1/1>$	$\Updownarrow(...w0^h, ...r0, ...)$
12	$\mathrm{p_i}$s RDF_1	$<w1^h r1_T/0/0>$	$\Updownarrow(...w1^h, ...r1, ...)$

Based on the detection conditions in Table 8.18, it is possible to derive memory tests that detect all single-cell soft faults for this specific memory design. Such a march test can have the form of March $1\mathrm{CS}_{swp}$ below.

$$\boxed{\begin{aligned} \text{March } 1\mathrm{CS}_{swp} = \{ \; & \underbrace{\Updownarrow(w0^h,\, r0,\, T,\, r0)}_{\text{ME0}};\; \underbrace{\Updownarrow(w1^h,\, r1,\, T,\, r1)}_{\text{ME1}}; \\ & \underbrace{\Updownarrow(w0^h,\, w1,\, T,\, r1)}_{\text{ME2}};\; \underbrace{\Updownarrow(w1^h,\, w0,\, T,\, r0)}_{\text{ME3}} \} \end{aligned}}$$

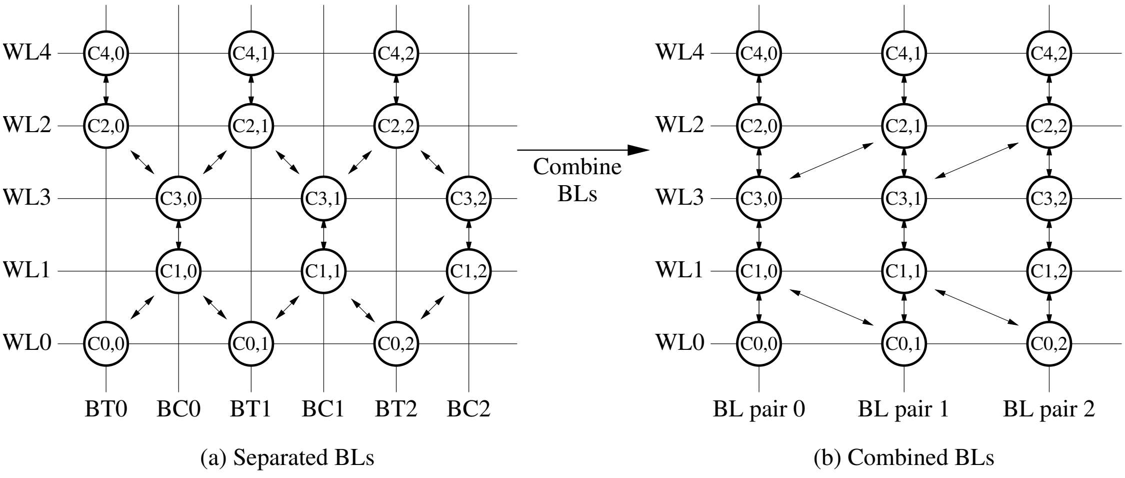
This march test has four march elements (ME0 through ME3), each of which begins with a hammer write operation and ends with a detecting read operation. This test is similar to the test for hard single-cell DRAM faults (March $1\mathrm{CH}_{swp}$), which is expected since the space of soft faults is derived from the space of hard faults. The test has a complexity of $8 \cdot n + 4 \cdot h \cdot n + 4 \cdot T \cdot n$, which is higher than $1\mathrm{CH}_{swp}$ by $4 \cdot T \cdot n$. For a typical idle time of $T > 64$ ms, the total idle test time becomes rather long relative to the total length of the test (assuming $h \approx 5$). In order to reduce test time, it is important to implement a number of test time reduction methods, which force soft faults to become directly detectable hard faults, as they do not require any delay time to detect. The theory behind these methods is discussed in Section 4.4.1, while an example of applying these methods to a specific test problem is discussed in Section 9.4.

8.3.2 Memory layout implications

Assuming that a test designer has no knowledge of the internal layout of the memory under test, then the designer is obliged to perform the most general form of memory tests to be able to detect every possible fault that may take place in the memory. It is possible, however, to reduce the test time of the memory by restricting the complexity of memory tests based on information of the layout, as discussed in this section.

Section 8.2 introduced a very computationally expensive test called March 2CT to test for all two-cell transient faults. This test has a complexity of the order $O(n^2)$ because it assumes that each cell can be coupled to all other cells in the memory. This assumption is unnecessary, since practically a cell can only be coupled to the closest physically neighboring cells on the layout. Once the layout of the memory is known, it is possible to reduce the complexity of this test from quadratic to linear with the number of memory cells.

The most widely used DRAM layout today is shown in Figure 3.16, as discussed in Section 3.3.3. This layout employs a *reflected WL organization* that allows memory cells to be placed closer to each other, which achieves a higher integration density than the regular WL organization. With this WL organization, each cell has three closest physical neighbors on the layout [Muhmenthaler91]. This situation is shown in Figure 8.3(a), where the closest neighbors are highlighted by arrows that connect between them. The reflected WL organization refers to the fact that WL3 follows WL1 instead of WL2.



(a) Separated BLs (b) Combined BLs

Figure 8.3. Physically neighboring cells (a) on the layout, and (b) with combined BLs.

Each memory cell in the figure is indicated by the letter C and a couple of numbers that refer to the WL and BL the cell is connected to. For example, the cell C3,2 is the memory cell connected to WL3 and BL2. All even numbered WLs (such as WL0, WL2, etc.) access cells connected to BT, while all odd numbered

WLs access cells connected to BC. Considering C1,1, for example, the three closest neighboring cells on the layout are C0,1, C0,2 and C3,1.

When march tests are applied to the memory under test, memory cells are accessed in an increasing, or a decreasing logical address order [refer to Section 4.2.4]. In the memory example shown in Figure 8.3(a), an increasing logical cell address corresponds to the following cell sequence C0,0, C1,0, C2,0, C3,0, C4,0, C1,0, C1,1, etc. In this example, a march test accesses a cell on BT first, then a cell on BC, then again on BT, then BC, etc., according to their logical address and not to their physical position. This means that from a march test point of view (i.e., using logical addressing), there is no difference between a cell connected to BT or to BC. Therefore, it is possible to combine each BT and BC of a given BL pair to inspect the way cell neighbors are organized from a march test point of view. This is done in Figure 8.3(b), which combines each BTx and BCx in Figure 8.3(a) into a single BL pair x line.

In order to identify all logically neighboring cells from Figure 8.3(b), it is only necessary to swap WL2 and WL3, so that a logically ordered WL sequence can be obtained. This is done in Figure 8.4. This figure shows clearly each cell and its closest neighbors, derived from the physical cell proximity in Figure 8.3(a).

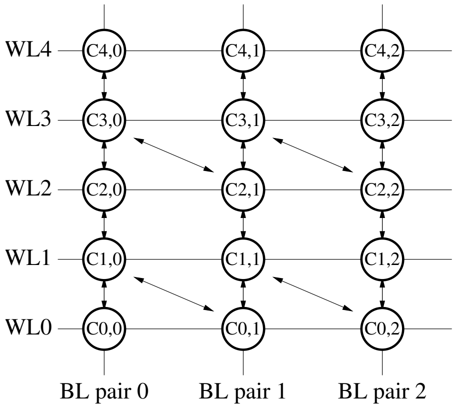


Figure 8.4. Logically neighboring cells.

Based on the information presented in Figure 8.4, it is possible to introduce a localized version of March 2CT (called March 2CTr) that reduces the complexity of the test from quadratic to linear, by limiting the possible cells that could act as aggressors to only the three cells connected by an arrow in Figure 8.4. The localized version of the test is given next.

$$\begin{array}{llll}
\text{March 2CTr} = \{\Updownarrow_i(& \Updownarrow_j(w0_i, w0_j^h, r0_i, r0_i), & \Updownarrow_j(w0_i, w1_j^h, r0_i, r0_i), & \Updownarrow_j(w1_i, w0_j^h, r1_i, r1_i), \\
 & \quad\text{ME0,0} & \quad\text{ME0,1} & \quad\text{ME0,2} \\
 & \Updownarrow_j(w1_i, w1_j^h, r1_i, r1_i), & \Updownarrow_j(w0_i, w0_j, w1_j^h, r0_i), & \Updownarrow_j(w1_i, w0_j, w1_j^h, r1_i), \\
 & \quad\text{ME0,3} & \quad\text{ME0,4} & \quad\text{ME0,5} \\
 & \Updownarrow_j(w0_i, w1_j, w0_j^h, r0_i), & \Updownarrow_j(w1_i, w1_j, w0_j^h, r1_i), & \Updownarrow_j(w0_i, w0_j, r0_j^h, r0_i), \\
 & \quad\text{ME0,6} & \quad\text{ME0,7} & \quad\text{ME0,8} \\
 & \Updownarrow_j(w1_i, w0_j, r0_j^h, r1_i), & \Updownarrow_j(w0_i, w1_j, r1_j^h, r0_i), & \Updownarrow_j(w1_i, w1_j, r1_j^h, r1_i), \\
 & \quad\text{ME0,9} & \quad\text{ME0,10} & \quad\text{ME0,11} \\
 & \Updownarrow_j(w0_i, w0_j^h, w0_i, r0_i), & \Updownarrow_j(w0_i, w1_j^h, w0_i, r0_i), & \Updownarrow_j(w1_i, w0_j^h, w1_i, r1_i), \\
 & \quad\text{ME0,12} & \quad\text{ME0,13} & \quad\text{ME0,14} \\
 & \Updownarrow_j(w1_i, w1_j^h, w1_i, r1_i), & \Updownarrow_j(w1_i, w0_j^h, w0_i, r0_i), & \Updownarrow_j(w1_i, w1_j^h, w0_i, r0_i), \\
 & \quad\text{ME0,15} & \quad\text{ME0,16} & \quad\text{ME0,17} \\
 & \Updownarrow_j(w0_i, w1_j^h, w1_i, r1_i), & \Updownarrow_j(w0_i, w0_j^h, w1_i, r1_i) & \quad)\} \\
 & \quad\text{ME0,18} & \quad\text{ME0,19} & \\
\text{where } \{j\} \text{ only has three elements} & & &
\end{array}$$

This test is identical to March 2CT, with the exception that j in this test refers only to the three cells connected to cell i by an arrow in Figure 8.4. This test has only one march element (ME0) that contains 20 nested march elements (ME0,0 through ME0,19). This test has a complexity of $180 \cdot n + 60 \cdot h \cdot n$, which is of the order $O(n)$.

Summary

This chapter presented the DRAM-specific tests needed to test for all the DRAM-specific fault models introduced in this thesis. Using these tests should give memory test engineers insights into the way fault models should be used to generate memory tests practically and efficiently. The main issues presented in this chapter are the following:

- Detailed discussion of the space of DRAM-specific faults, using five individual fault attributes, which result in 12 different attribute combinations. These 12 attribute combinations modify the behavior of a given generic FP in such a way that describes all DRAM-specific faults.
- Discussion of the two DRAM-specific fault models related to improperly set voltages: partial faults and dirty faults.
- Discussion of the two time dependent fault models in DRAMs related to leakage currents: soft faults and transient faults. Soft faults are characterized by the need to wait for a specific period of time to ensure the sensitization of the fault, while transient faults are characterized by the need to perform all sensitizing operations successively and without waiting to ensure sensitizing the fault.
- Reduction of the general space of DRAM faults to a set of realistic memory faults that have been observed by simulation in the faulty behavior of a DRAM.
- Derivation of DRAM-specific tests for each of the proposed realistic memory faults. In total, six general tests have been proposed to detect (hard, transient and soft) single-cell and two-cell faults.
- Customization of DRAM-specific tests to reduce test complexity and suit specific memory organizations. Two different bases for test reduction have been proposed: reduction on the bases of memory design and reduction on the bases of memory layout. Four different customized tests have been proposed that significantly reduce the complexity of the general tests.

Contents of this chapter

9.1 Definition of problem
9.2 Analysis methodology
9.3 Results and tests
9.4 Optimizing stresses

9

Case study: the strap problem

In order to further clarify the concepts of the simulation-based fault analysis method and the faulty behavior approximation algorithms, this chapter presents a simulation-based analysis of the faulty behavior resulting from a special type of open defect within memory cells. This type of open causes only one floating node in the memory, and it is thus an appropriate example for the one dimensional (1D) analysis [see Section 5.2]. The strap problem is different from a typical open defect within the cell in having a special parabolic dependence on temperature. This chapter runs through the whole course of a practical application of the 1D analysis, starting from problem definition through test generation and ending in stress optimization with respect to timing, temperature and voltage. The analysis used in this section uses the five DRAM-specific commands, rather than the generic memory operations $w0$, $w1$ and r. This chapter is based on a more elaborate internal report written for Infineon, detailing the procedure and ending with the practical application of the derived tests for their memories [Al-Ars02d, Al-Ars02b].

Section 9.1 starts with a definition of the strap problem, along with an electrical model of it to be used for the simulation. This section also discusses the concepts of process corners and process variations, and presents a way to model them at the electrical level. Section 9.2 describes the simulation-based fault analysis methodology used to analyze the faulty behavior of the strap. Section 9.3 takes process variations into consideration, discusses their impact on the faulty behavior, and derives memory tests to detect the faults in all process corners. Finally, Section 9.4 uses the concept of the critical resistance for stresses optimization of the derived memory tests.

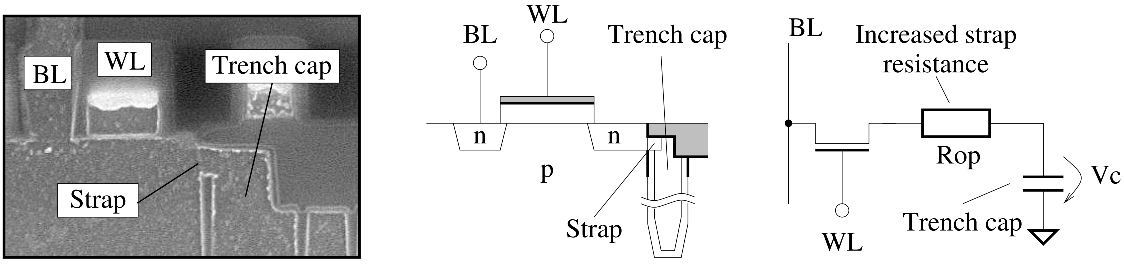
9.1 Definition of strap problem

In this section, we provide some background information regarding the simulation model, the open defect to be analyzed, the concept of process corners, and the simulated sequences.

9.1.1 The defect

The special open defect analyzed in this chapter models an increase in the resistive value of what is called the **strap connection**. The strap connection is a conductive path between the drain region of the pass transistor of the memory cell and the trench capacitor [Adler95]. Figure 9.1(a) shows an SEM (scanning electron microscope) image of the memory cell, where the position of the strap is indicated. Figure 9.1(b) gives a schematic representation of the cell and strap. Due to imperfections in the fabrication process, the strap may take up any resistive value according to the statistical distribution of the fabrication process.

Ideally, the memory is designed such that the strap should be manufactured with a predefined target value (R_{st}), that is integrated as part of the pass transistor model of the memory cell. An *increase* in the strap resistance can be electrically modeled as an added series resistance (R_{op}) along the conductive path between the pass transistor and the trench capacitor in the cell, as shown in Figure 9.1(c). An increase in R_{op} reduces the ability of the memory to control the voltage stored across the cell capacitor, which leaves the stored voltage in the cell (V_c) floating to a certain extent.



(a) (b) (c)

Figure 9.1. Modeling the cell and the strap from (a) silicon, to (b) the layout level, and (c) the electrical level (source: Infineon Technologies).

From a physical point of view, the modeled increase in the strap resistance R_{op} can be attributed to a number of factors, such as a change in the doping concentration of the strap, or a geometrical misalignment in the positioning or sizing of the strap.

9.1.2 Simulation model

The memory Spice model used to perform the simulations for the strap problem is the same as the model discussed in Section 7.1, used to analyze the faulty behavior of the opens, shorts and bridges in Chapter 7. To reduce simulation time, the memory model is reduced to include only those parts of the memory needed to perform the fault analysis. Figure 9.2 shows a block diagram of the memory model. The model has three BL pairs (BTt-BCt, BTm-BCm and BTb-BCb), each containing 2 memory cells, one of which is connected to the true bit line (BT) while the other is connected to the complementary bit line (BC). In addition, the model has 3 sense amplifiers (SAt, SAm and SAb), precharge circuits and access devices. A write buffer is included to enable simulating a write operation, in addition to a read buffer for simulating a read operation.

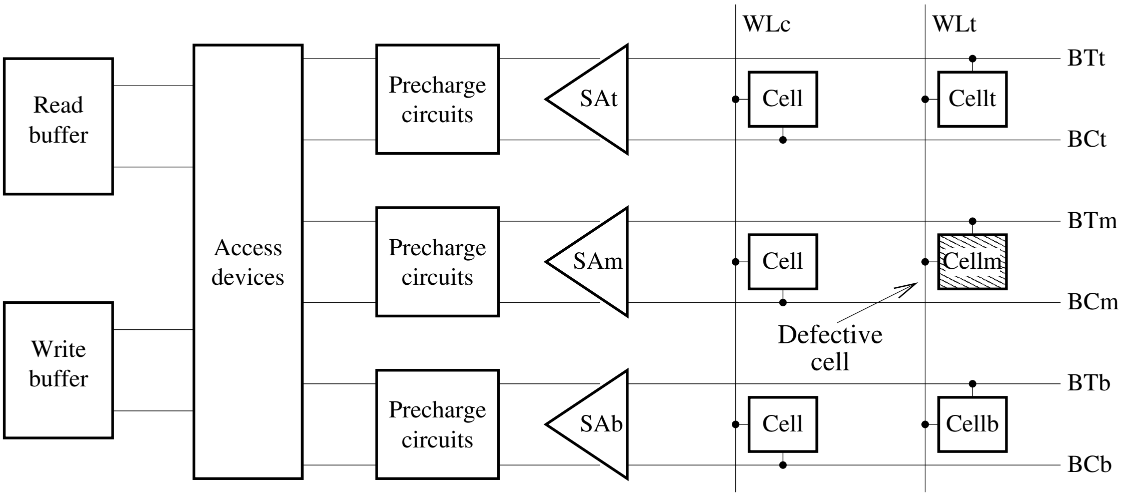


Figure 9.2. Spice simulation model used in the fault analysis.

Each word line in the model is connected to three memory cells, WLt is connected to three cells on BT, while WLc is connected to three cells in BC. The fault analysis described in this section is performed on Cellm (the memory cell connected to WLt and BTm). The behavior of cells connected to BC is the complementary to that of cells connected to BT. The faulty behavior of a cell connected to BC is the same as the faulty behavior of a cell connected to BT, with all 0s in the fault replaced with 1s, and vice versa [see Section 7.2.1].

In order to take the impact of background patterns (BGs) into consideration, we assume that the faulty behavior of R_{op} can only be influenced by the three BL pairs in the simulation model. In the case of the strap open, Cellt and Cellb are influential on the faulty behavior, since all three are connected to the same word line (WLt). Therefore, when Cellm is accessed, Cellt and Cellb are accessed at the same time, thereby influencing the behavior of the operations performed on Cellm. To simulate the impact of different BGs, the simulation analysis is performed for different voltages stored in Cellt and Cellb. There are four different BGs:

1. **BG 00**—This refers to 0s stored in Cellt on BTt and Cellb on BTb.
2. **BG 10**—This refers to a 1 stored in Cellt on BTt and a 0 stored in Cellb on BTb.
3. **BG 11**—This refers to 1s stored in Cellt and Cellb.
4. **BG 01**—This refers to a 0 stored in Cellt and a 1 stored in Cellb.

9.1.3 Process variations

During the fabrication process of memory components, the characteristics of the produced components vary in a given range around their target specifications. These variations result in some components operating slower or faster than the targeted speed of operation. It is important to evaluate the impact of process variations on the faulty behavior of the memory, since a test should be able to detect the faulty behavior for *any* produced memory component.

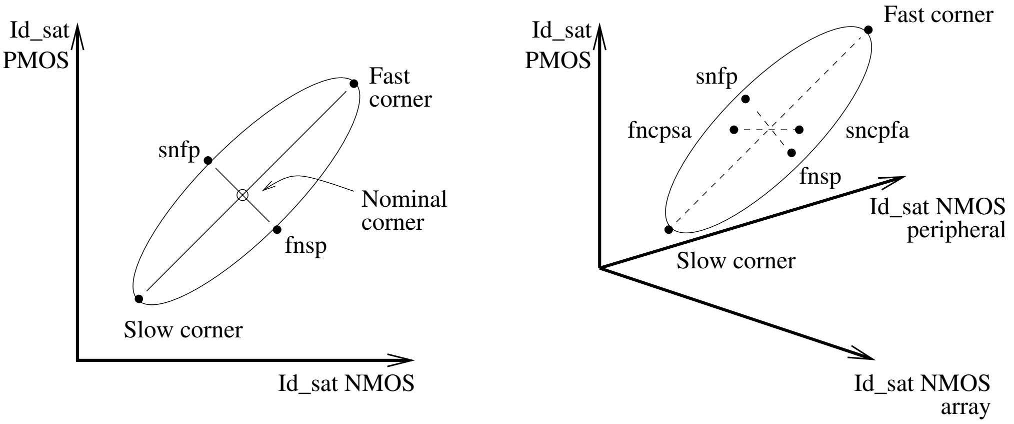
The fact that produced components operate with different speeds implies that transistors on the chip operate with different speeds. Therefore, process variations can be modeled and simulated as variations in the transistor parameters used in simulation [Foty97]. Since each type of transistor on chip (there are a number of PMOS and NMOS transistors with different doping, oxide thickness, etc.) is produced with its own sequence of process steps, there should be a separate set of parameter variations for each transistor type. Since such an approach results in a large number of possible variations, transistors are usually divided into *sets* that are supposed to have a correlated behavior. Here, we discuss two ways to classify transistors:

- **2D variations**—These refer to dividing all transistors into two sets, PMOS and NMOS, each with independently varying parameters. From a design point of view, it is usually considered enough to consider variations in PMOS and NMOS transistors separately.
- **3D variations**—From a fault analysis point of view, in addition to the independent PMOS variations, variations in array NMOS transistors (or pass transistors) should be considered independently from variations in other NMOS transistors (or peripheral transistors), since array NMOS transistors are manufactured in a special way different from other transistors.

2D variations

Figure 9.3(a) shows how to model process variations considering PMOS vs NMOS transistor variations. The x-axis represents the measured saturation current of

NMOS transistors (Id_sat NMOS), while the y-axis represents the measured saturation current of PMOS transistors (Id_sat pFET). The ellipse represents the *process spread* around the center. Each point in the ellipse represents one possible combination of Id_sat for NMOS vs PMOS. Transistors with Id_sat at the center of the ellipse (the crossing of the two principal axes) are said to belong to the *nominal corner.* These are modeled using the nominal transistor parameters. For every other point in the ellipse, a set of variations in the transistor parameters (Δparameter) are added to the nominal parameter set.



(a) Correlated array NMOS

(b) Independent array NMOS

Figure 9.3. Modeling process variations using variations in transistor behavior: PMOS vs (a) a single set of correlated NMOS transistors, and vs (b) array NMOS taken to vary independently from peripheral NMOS transistors.

A **process corner** is a term that refers to any point in the ellipse of Figure 9.3(a). This means that each process corner has an associated set of transistor parameters describing transistor behavior at that point. Besides the nominal corner, the figure shows four additional process corners:

- **snsp corner** or slow corner—This refers to the process corner where NMOS and PMOS transistors conduct the least drain current. The symbol snsp stands for slow NMOS and slow PMOS.
- **snfp corner**—This corner has slow NMOS and fast PMOS transistors. It refers to the process corner with low NMOS drain current and high PMOS drain current.
- **fnsp corner**—This corner has fast NMOS and slow PMOS transistors. It refers to the process corner with high NMOS drain current and low PMOS drain current.
- **fnfp corner** or fast corner—This refers to the process corner where NMOS and PMOS transistors conduct the most drain current.

Note that the fast and slow corners (fnfp and snsp) are the so-called 3σ-corners, since they have 3 times the standard variations of the statistical spread of the fabrication process. If the tests generated by the 3σ variations are considered too pessimistic (unrealistic), the 2σ-corners can be used to have more reliable simulations. Note that the distance between the snfp and the fnsp corners is less than the distance between the fnfp and snsp corners. This is caused by the correlation the NMOS and PMOS have with each other because they are fabricated in close proximity on the same chip [Foty97].

3D variations

Figure 9.3(b) shows how variations in the PMOS, array NMOS and peripheral NMOS may be considered independently. The x-axis represents the measured saturation current of array NMOS transistors (Id_sat NMOS array), the y-axis represents the measured saturation current of peripheral NMOS transistors (Id_sat NMOS peripheral), while the z-axis represents that of PMOS transistors (Id_sat PMOS). The figure shows a 3D ellipsoid that represents the process spread around the center. Each point in the ellipsoid represents one possible combination of Id_sat for the three sets of transistors. Transistors with Id_sat at the center of the ellipsoid (the crossing of the three principal axes) are modeled using the nominal transistor parameters. For every other point in the ellipsoid, a set of variations in the transistor parameters (Δparameter) are added to the nominal parameter set.

Note that the ellipse in Figure 9.3(a) can be retrieved from Figure 9.3(b) by intersecting the ellipsoid with the plane $x = y$ (i.e., with the condition that all NMOS transistors are totally correlated). Also note that many process corners in Figures 9.3(a) and (b) are the same: nominal corner, snsp, fnfp, snfp and fnsp. There are, however, two added corners that are specific to the 3D variations.

- **sncpfa corner**—This corner has slow peripheral NMOS, centralized or nominal PMOS, and fast array NMOS transistors.

- **fncpsa corner**—This corner has fast peripheral NMOS, centralized or nominal PMOS, and slow array NMOS transistors.

9.2 Analysis methodology

In this section, we describe the analysis methodology by discussing the analysis performed at the nominal corner of the process. Four different backgrounds (BGs) are analyzed: BG 00, BG 10, BG 11, and BG 01. We start by presenting the DRAM-specific sequences used in the analysis, followed by a detailed discussion of the results of BG 00, then the results for the other BGs are summarized.

9.2.1 Simulated sequences

In the approximate simulation approach, we need to understand the faulty behavior of the memory for any possible sequence of memory operations, but since there is an infinite number of possible sequences, it is impossible to simulate all of them. The solution is to simulate a limited number of sequences, called *basic sequences*, and then use those to *approximate* the behavior of any other sequence. Therefore, we need to find those DRAM-specific basic sequences, consisting of the commands Act, Wr, Rd, Pre and Nop, that make such an approximation possible.

Section 5.4 presents 13 DRAM-specific basic sequences using the five DRAM commands, to perform a simulation-based fault analysis of the memory. These 13 basic sequences are listed in Table 9.1. They are classified into four different groups: charge up group, discharge group, charge/discharge group (C/D group), and no charge group. The name in each group refers to the direction of expected voltage change within the memory cell during the application of the sequence.

Table 9.1. Basic sequences used to reconstruct any DRAM-specific sequence.

#	Charge up group	#	Discharge group	#	C/D group	#	No charge group
1.	`Wr1 Wr1 Wr1 ... Wr1`	5.	`Wr0 Wr0 Wr0 ... Wr0`	9.	`Act Rd Rd ... Rd`	13.	`Pre Nop Nop ... Nop`
2.	`Wr1 Rd1 Rd1 ... Rd1`	6.	`Wr0 Rd0 Rd0 ... Rd0`	10.	`Act Nop Nop ... Nop`		
3.	`Wr1 Nop Nop ... Nop`	7.	`Wr0 Nop Nop ... Nop`	11.	`Act Act Act ... Act`		
4.	`Wr1 Pre Pre ... Pre`	8.	`Wr0 Pre Pre ... Pre`	12.	`Act Pre Pre ... Pre`		

Taking the strap problem into consideration, and since the cell is only accessed between the Act and Pre commands, faults can only be sensitized when the cell is accessed using the commands performed between Act and Pre. Therefore, for the analysis of the strap, we can exclude Sequences 4, 8 and 12 in Table 9.1, since they inspect the impact of Pre commands, we can exclude Sequence 11 since it inspects the impact of Act commands, and exclude Sequence 13 since it inspects the impact of Nop commands performed after Pre. This leaves the following sequences to be analyzed:

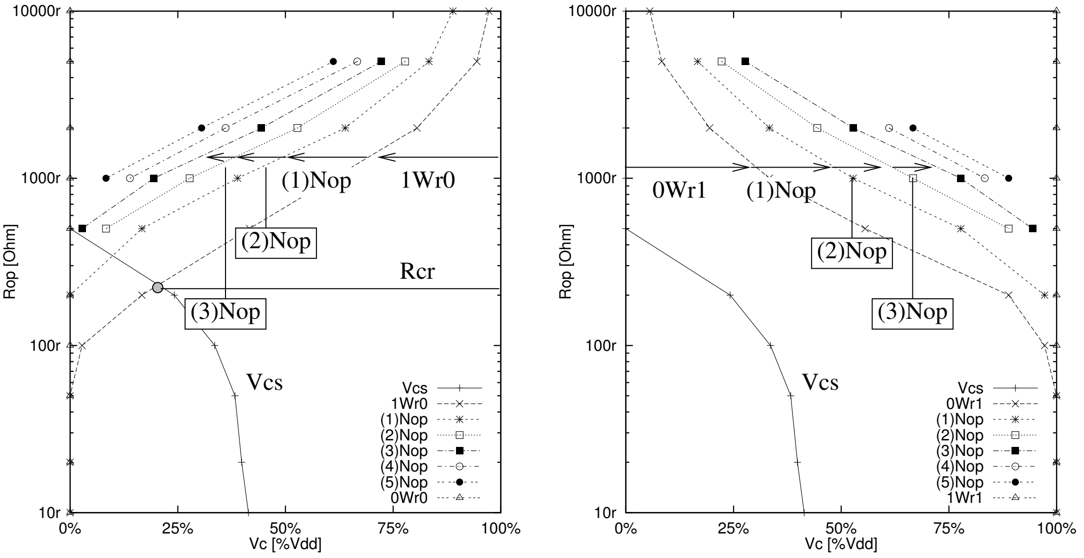
- Sequences 1, 2 and 3 of the charge up group
- Sequences 5, 6 and 7 of the discharge group, and
- Sequences 9 and 10 of the C/D group.

Fortunately, we do not need to generate all 8 result planes for these 8 sequences. Simulations show that, for the strap problem, all remaining 3 sequences in each group have similar impact on the faulty behavior, which means that the result planes they generate are similar [Al-Ars02c]. Therefore, only one sequence from the remaining 3 in each group needs to be simulated as a representative sequence. We select the sequences with the no operations as representatives, since these are the most suitable candidates for test generation. The clock cycles with no operations can be used to test multiple memory banks in parallel, thereby reducing overall test time. The following sequences are chosen to analyze the faulty behavior:

- Sequence of Wr0: `Wr0 Nop Nop ... Nop` (Sequence 7 in Table 9.1)
- Sequence of Wr1: `Wr1 Nop Nop ... Nop` (Sequence 3)
- Sequence of Act: `Act Nop Nop ... Nop` (Sequence 10)

9.2.2 Background 00

Figure 9.4 and 9.5 show the simulation results in the nominal process corner (as defined in Section 9.1.3), at nominal stresses (according to the specifications of the memory), and with BG 00 (0 is stored in cells on the adjacent BL pairs). The results are divided into three different result planes, one for each analyzed basic sequence [compare with Figure 5.4 and 5.5]. Each result plane describes the impact of performing successive commands on V_c for a given value of R_{op}, shown in Figure 9.1(c). The x-axes in the result planes represents the stored voltage within the cell (V_c), and the y-axis represents the value of the open resistance (R_{op}). The value of V_c is not given in absolute voltage levels, but as percentages of V_{dd}. In the same way, a scaled value of R_{op} is shown on the y-axis using the scale factor r.



(a) Plane of Wr0

(b) Plane of Wr1

Figure 9.4. Result planes in the nominal corner, at nominal stress and with BG 00, for the sequences (a) Wr0, and (b) Wr1.

Plane of Wr0: This result plane is shown in Figure 9.4(a). To generate this figure, the floating cell voltage V_c is initialized to the two worst case voltages, V_{dd} and GND, and then the sequence Wr0 Nop ... Nop is applied to the cell. With an initial V_c of V_{dd}, the sequence results in the gradual decrease (depending on the value of R_{op}) of V_c towards GND. With an initial V_c of GND, the value of V_c remains at GND. The voltage level after each command in the sequence is recorded on the result plane, which results in a number of curves in the plane. All curves have names, and some of them are indicated by an arrow pointing in the direction of voltage change. The 1Wr0 curve identifies the impact of Wr0 on a cell voltage initialized to V_{dd}, while the 0Wr0 curve (the last entry in the legend) identifies the impact of Wr0 on a cell voltage initialized to GND. The curves numbered as (n)Nop indicate the impact of no operations on V_c following a 1Wr0, where n is the number of Nops needed to get to the indicated curve. We stop performing the sequence when the voltage change ΔV_c as a result of the performed sequence becomes small enough. In this example, we stop the sequence when $\Delta V_c \leq 0.1$ V, a value that is arbitrarily selected at first, but can afterwards be reduced if it turns out that a longer sequence is needed to describe the faulty behavior. The figure also shows the cell sense threshold curve (V_{cs}), above which the sense amplifier senses a 1 and below which the sense amplifier senses a 0. The V_{cs} curve is copied from the plane of the Act sequence, which is explained in detail below [see "Plane of Act" below].

Plane of Wr1: This result plane is shown in Figure 9.4(b). To generate this figure, V_c is initialized to the two worst case voltages V_{dd} and GND and then the sequence Wr1 Nop ... Nop is applied to the cell. With an initial V_c of GND, the result is the gradual increase of V_c towards V_{dd}, while an initial V_{dd} remains as it is in the cell. The voltage level after each command in the sequence is recorded on the result plane, which produces a number of curves in the plane. These curves are indicated in the same way as for the curves in the plane of Wr0 above. We stop the sequence when ΔV_c becomes small enough (0.1 V in this example).

Plane of Act: This result plane is shown in Figure 9.5. To generate this figure, first the threshold within the cell that determines the sense amplifier output V_{cs} (the cell voltage above which the sense amplifier detects a 1, and below which it detects a 0) is established and indicated on the result plane. Then the sequence Act Nop ... Nop is applied twice: first for V_c that is initially marginally lower than V_{cs}, and a second time for V_c that is marginally higher than V_{cs}. The voltage level after each command is recorded on the result plane, which results in a number of curves on the plane. The +Act curve indicates the impact of performing Act with V_c marginally higher than V_{cs}, while -Act indicates the impact of performing Act with V_c marginally lower than V_{cs}. The other curves indicated the impact of the nth Nop following the initial Act.

Using the result curves in Figure 9.4, we can analyze the following aspects of the

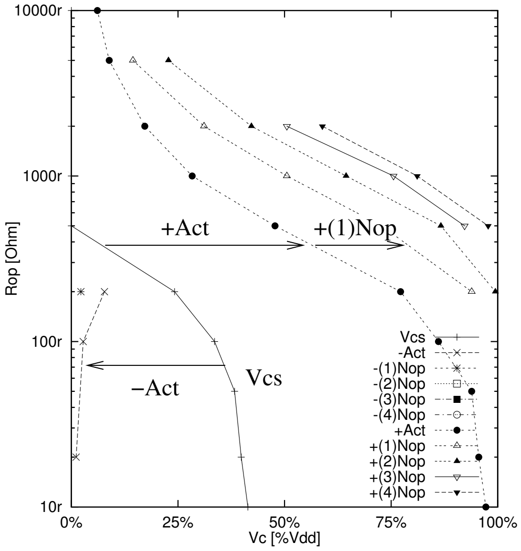


Figure 9.5. Result plane in the nominal corner, at nominal stress and with BG 00, for the Act sequence.

faulty behavior:

1. Identify the *critical resistance* (R_{cr}), which is the R_{op} value where the cell *starts* to cause faults on the output, for *any* sequence of operations.

2. Generate a test that detects the faulty behavior of the defect for any resistance value and any initial floating voltage.

(1) For the fault analysis shown in Figure 9.4, the memory behaves properly for any operation sequence as long as $R_{op} < 210r\Omega$. To understand why, note that a fault would only be detected when a Wr1 operation fails to charge V_c up above V_{cs}, or a Wr0 fails to discharge V_c to below V_{cs} (V_{cs} is indicated by a curve in Figure 9.4). In both situations, trying to read after performing the write would detect the faulty behavior. Note that for $R_{op} > 210r\Omega$, Wr0 fails to discharge V_c to the value needed by Act to sense a 0. This is indicated in Figure 9.4(a) as a dot at the intersection between the 1Wr0 curve and the V_{cs} curve. Furthermore, note that the curve 0Wr1 in Figure 9.4(b) does not intersect the V_{cs} curve, which means that Wr1 operations never fail no matter how high R_{op} becomes!

(2) Now, the result planes are used to generate a detection condition that detects the faulty behavior caused by any defect resistance for any initial floating voltage, in case a fault can be detected. Figure 9.4 shows that faults can be detected with $R_{op} \geq 210r\Omega$. Inspecting the figure shows that with $R_{op} \geq 210r\Omega$, and with any voltage V_c, the sequence Act Wr1 Nop Nop Wr0 will sensitize a fault. This, in turn, means that the faulty behavior can be represented by the FP =<Wr1 Nop Nop Wr0/1/−>. For $R_{op} = 210r\Omega$, this can be validated by noting that performing Wr1

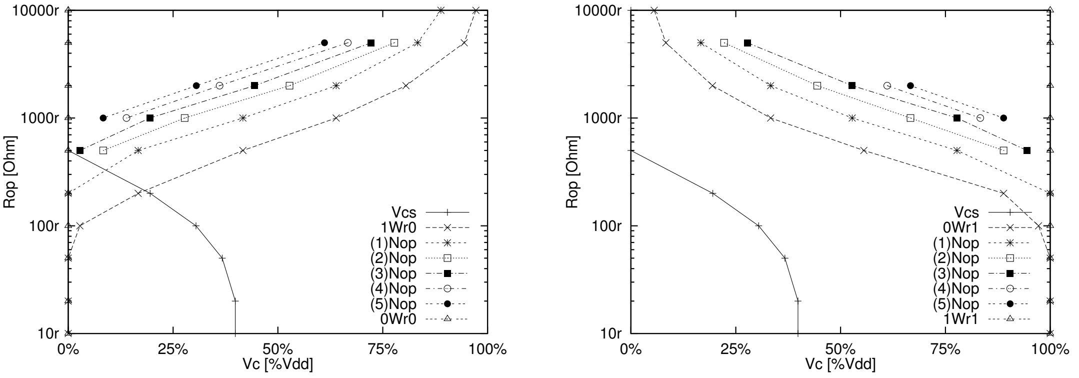
Nop Nop charges V_c up from any voltage (GND or higher) to approximately V_{dd}. With $V_c = V_{dd}$, performing Wr0 sensitizes the fault which can then be detected as discussed in point (1) above. Since the strap does not force leakage current to any specific direction, all time dependent faults are possible (hard, soft and transient). This means that the following two additional FPs are also possible: <Wr1 Nop Nop $\text{Wr0}_T/1/-$> and <$\underline{\text{Wr1}}$ $\underline{\text{Nop}}$ $\underline{\text{Nop}}$ $\underline{\text{Wr0}}/1_L/-$>. To detect the hard and transient faults, the detection condition $\Updownarrow$(..., Wr1, Nop, Nop, Wr0, Pre, Act, Rd0, ...) is sufficient. The soft fault requires the detection condition $\Updownarrow$(..., Wr1, Nop, Nop, Wr0, Pre, Del, Act, Rd0, ...), where Del stands for a delay time to test the ability of the cell to retain its data.

9.2.3 Backgrounds 10, 11 and 01

In this section, fault analysis results are given in the nominal process corner and with nominal stress conditions for BGs 10, 11 and 01. For each BG, the result planes are presented in order to discuss their faulty behavior. As the discussion in Section 9.2.2 have shown, the only important part of the result plane of Act is the V_{cs} curve, which is included and discussed as part of the result planes of Wr0 and Wr1. Therefore, only the result planes of Wr0 and Wr1 are discussed next, but not that of Act.

Background 10

Figure 9.6 shows the result planes for BG 10. Figures 9.6(a) and (b) give the results for Wr0 and Wr1, respectively. The curves in the figures show the same tendencies in the behavior as those in Figure 9.4.



(a) Wr0

(b) Wr1

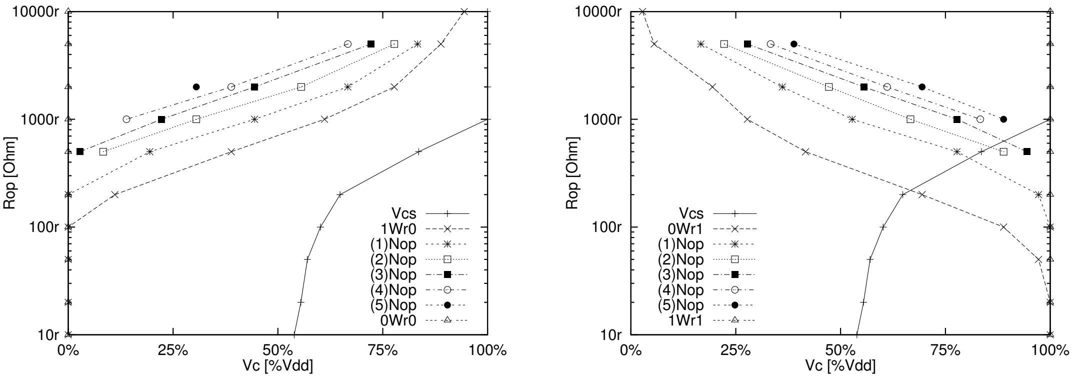
Figure 9.6. Result planes with BG 10 for (a) Wr0 and (b) Wr1.

Each figure has a number of curves. The curve V_{cs} represents the sense amplifier threshold voltage, below which a 0 is detected on the output and above which a 1 is detected. The other curves represent the impact of each corresponding command on the cell voltage V_c. For example, points of 1Wr0 indicate the voltage the cell contains after Wr0 to a cell initialized to V_{dd}, for a given R_{op} value. In the same way, points of (1)Nop indicate the voltage the cell contains after one Nop command is performed, and so on.

Although the results in this figure are not exactly the same as those shown in Figure 9.4, the important aspects of the faulty behavior have not changed. The cell starts to fail when $R_{op} \geq 205r\Omega$ when the curve V_{cs} intersects 1Wr0 in Figure 9.6(a). This means that modifying BG from 00 to 10 does not have a big impact on the faulty behavior of the memory. Yet, the small increase in the range of failing R_{op} values indicates that BG 10 is slightly more effective in stressing the test than BG 00.

Background 11

Figure 9.7 shows the analysis results for BG 11. Figures 9.7(a) and (b) give the results for Wr0 and Wr1, respectively. The figures show that the V_{cs} curve changed significantly with BG 11, compared to BGs 10 and 00. The sense amplifier is now biased towards detecting a stored 0 instead of detecting a stored 1. A detailed discussion of the impact of BGs on the way the sense amplifier senses stored cell voltages can found in Chapter 6.



(a) Wr0

(b) Wr1

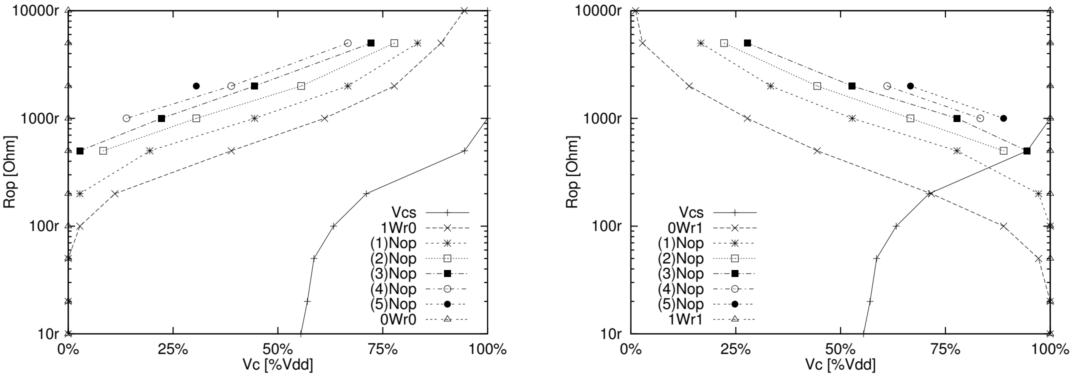
Figure 9.7. Analysis results with BG 11 for (a) Wr0 and (b) Wr1.

Inspecting Figure 9.7(a) reveals that the V_{cs} curve does not intersect the 1Wr0 or any of the (n)Nop curves, which means that the Wr0 sequence never fails. However, the V_{cs} curve does intersect the 0Wr1 curve in Figure 9.7(b) at about $R_{op} = 205r\Omega$. This indicates that the Wr1 sequence starts to fail with $R_{op} \geq 205r\Omega$.

A detection condition to detect the hard and transient types of this fault is $\Updownarrow$(..., Wr0, Nop, Nop, Wr1, Pre, Act, Rd1, ...). The soft fault requires the detection condition $\Updownarrow$(..., Wr0, Nop, Nop, Wr1, Pre, Del, Act, Rd1, ...), where Del stands for a delay time to test the ability of the cell to retain its data. It is interesting to note that this detection condition has a similar sequence of commands as the detection condition derived for BG 00, with the exception that the data used in this detection condition is complementary to that used in the condition for BG 00 (i.e., 1s are replaced with 0s, and vice versa).

Background 01

Figure 9.8 shows the analysis results for BG 01. The results in this figure are similar to those shown in Figure 9.7. The cell starts to fail when $R_{op} \geq 200r\Omega$ when the curve V_{cs} intersects the 0Wr1 curve in Figure 9.8(b). This means that modifying BG from 11 to 01 does not have a big impact on the faulty behavior of the memory.



(a) Wr0

(b) Wr1

Figure 9.8. Analysis results with BG 01 for (a) Wr0 and (b) Wr1.

Still, close inspection of the results associated with BG 11 and 01 reveals that using BG 01 slightly increases the range of failing R_{op} as compared to BG 11. This means that it is actually easier to detect a fault with BG 01. A similar remark has been made for BGs 00 and 10.

9.3 Results and tests

In this section, a summary of the results is given for the fault analysis performed in the 7 different process corners, as discussed in Section 9.1.3. In addition, tests are given that satisfy the detection conditions required to detect the faulty behavior of the strap.

9.3.1 Summary of results

All 7 process corners (nominal, fnfp, snsp, snfp, fnsp, sncpfa and fncpsa) have been simulated and analyzed [see Section 9.1.3] at nominal stresses (nominal voltage, temperature and timing). For each process corner and each BG, the three result plains (Wr0, Wr1 and Act) have been generated [Al-Ars02d]. To simplify the discussion here, only the two most important outcomes of the analysis are given:

(1) the critical resistance (R_{cr}) at which the memory starts to fail, and (2) the detection condition needed to detect the faulty behavior. Table 9.2 lists these two outcomes for each process corner. The column “Corner” lists the process corner at which the analysis is performed, the column “BG” indicates the background data, the column R_{cr} states the critical resistance, while the column “Det. condition” lists the needed detection condition.

Table 9.2. Summary of results in the 7 different process corners [Al-Ars02d].

Corner	BG	R_{cr}	Det. condition	Corner	BG	R_{cr}	Det. condition
Nominal	00	$210r\Omega$ ■	CondT	fnfp	00	$220r\Omega$ ■	CondT
	10	$205r\Omega$ ■	CondT		10	$205r\Omega$ ■	CondT
	11	$205r\Omega$ ■	CondC		11	$220r\Omega$ ■	CondC
	01	$200r\Omega$ ■	CondC		01	$205r\Omega$ ■	CondC
snsp	00	$200r\Omega$ ■	CondT	snfp	00	$210r\Omega$ ■	CondT$^+$
	10	$185r\Omega$ ■	CondT		10	$205r\Omega$ ■	CondT$^+$
	11	$105r\Omega$ ▪	CondC$^-$		11	$205r\Omega$ ■	CondC
	01	$100r\Omega$ ▪	CondC$^-$		01	$200r\Omega$ ■	CondC$^-$
fnsp	00	$205r\Omega$ ■	CondT	sncpfa	00	$205r\Omega$ ■	CondT
	10	$200r\Omega$ ■	CondT		10	$200r\Omega$ ■	CondT
	11	$200r\Omega$ ■	CondC$^-$		11	$160r\Omega$ ▪	CondC
	01	$160r\Omega$ ▪	CondC$^-$		01	$130r\Omega$ ▪	CondC
fncpsa	00	$210r\Omega$ ■	CondT				
	10	$205r\Omega$ ■	CondT				
	11	$210r\Omega$ ■	CondC$^-$				
	01	$205r\Omega$ ■	CondC$^-$				

CondT: $\Updownarrow$(..., Wr1, Nop, Nop, Wr0, Pre, Act, Rd0, ...)
CondC: $\Updownarrow$(..., Wr0, Nop, Nop, Wr1, Pre, Act, Rd1, ...)
CondT$^-$: $\Updownarrow$(..., Wr1, Nop, Wr0, Pre, Act, Rd0, ...)
CondC$^-$: $\Updownarrow$(..., Wr0, Nop, Wr1, Pre, Act, Rd1, ...)
CondT$^+$: $\Updownarrow$(..., Wr1, Nop, Nop, Nop, Wr0, Pre, Act, Rd0, ...)

The detection conditions listed in the table detect the hard as well as the transient faults cased by the strap, but not the soft faults. In order to detect the soft faults, a delay of (Del) should be added to each detection condition, after the Pre and before the Act commands. Section 9.4 below discusses how to apply test stresses in order to eliminate soft faults without the need to include this time delay into the detection conditions.

The table identifies the following characteristics for the faulty behavior, resulting from an elevated strap resistance with nominal stresses:

1. The value of R_{cr} varies between a minimum of $100r\Omega$ for the snsp corner with BG 01, and a maximum of $220r\Omega$ for the fnfp corner with BG 00 and BG 11.

2. For each process corner, BG 01 always results in the lowest R_{cr}, while BG 00 results in the highest. This means that BG 01 is the most stressful BG, while BG 00 is the most relaxed one.
3. Depending on the process corner, the difference in R_{cr} between different BGs could be as large as 100 $r\Omega$ for snsp, or as small as 5 $r\Omega$ for fncpsa.
4. BG 00 results in the most stable R_{cr} value across different process corners. With BG 00, R_{cr} changes by only 20 $r\Omega$ from a high of 220 $r\Omega$ for fnfp and the a low of 200 $r\Omega$ for snsp. This is a change of about 9% from the high R_{cr} value.
5. The process corners sncpfa and fncpsa, specifically considered for fault analysis, did not result in significantly low nor significantly high R_{cr} values. This could probably be attributed to a compensating effect between array NMOS and periphery NMOS transistor (i.e., an increase in the speed of array devices compensates a decrease in the speed of periphery devices, and vice versa).
6. The detection conditions needed to detect the faulty behavior caused by the defect have the same general structure for all process corners. The only difference is in the number of initializing Wr and Nop operations needed to precharge V_c to a strong enough voltage.
7. In general, slow array NMOS transistors require more initializing Nops (longer detection condition), while fast array CMOS transistors require less initializing Nops (shorter detection condition). However, PMOS transistors also have an influence on the length of the detection conditions.

9.3.2 Test generation

In this section, we use the detection conditions listed in Table 9.2 to generate a march test to detect the fault, taking into account the needed BG pattern. Note that these tests are only suitable to detect the hard and transient faults caused by the strap, but not to detect soft faults, as they require incorporating a delay time into the test. Section 9.4 below shows how to use test stresses rather than delay time to eliminate soft faults.

Simulation-based tests

According to Table 9.2, BGs 00 and 10 share the same worst-case detection condition, which is $\Updownarrow$(..., Wr1, Nop, Nop, Nop, Wr0, Pre, Act, Rd0, ...) corresponding to the corner snfp. The fact that changing the value of Cellt between 0 and 1 does not change the detection condition means that Cellt is insignificant for the faulty behavior. This indicates that in order for this detection condition to work, only Cellb [see Figure 9.2] should contain a 0. This is done using a memory initialization step to write a 0 in the whole memory, as follows $\Updownarrow$(Act, Wr0, Pre).

For BGs 11 and 01, the worst-case detection condition is $\Updownarrow$(..., Wr0, Nop, Nop, Nop, Wr1, Pre, Act, Rd1, ...). In this case, the BG indicates that only a value of 1 contained in Cellb is significant for the detection condition. This is done using an initialization step to write a 1 in the whole memory, as follows $\Updownarrow$(Act, Wr1, Pre).

Therefore, there are two possible tests, any of which detects the faulty behavior. The subscripts 00 and 11 stand for the BG used in the test.

1. T_{00} = {$\Updownarrow$(Act, Wr0, Pre);
   $\Updownarrow$(Act, Wr1, Nop, Nop, Nop, Wr0, Pre,
   Act, Rd0, Pre)}
2. T_{11} = {$\Updownarrow$(Act, Wr1, Pre);
   $\Updownarrow$(Act, Wr0, Nop, Nop, Nop, Wr1, Pre,
   Act, Rd1, Pre)}

Since each of the two tests detects the faulty behavior, we only need to perform one of them to properly test for the strap problem. Referring to Table 9.2, it is clear that T_{11} has a lower R_{cr} than T_{00} in all analyzed process corners. Therefore, we choose to use T_{11}.

Practical aspects

If we assume that the memory functions exactly as the electrical model does on a simulator, the generated tests above would function exactly as expected. Practically, however, real silicon may deviate from the simulated behavior. As a result, the exact number of operations needed to charge the memory to the desired voltage may differ. It is possible to increase the test time in an attempt to ensure coverage by increasing the number of Nop operations in the test. This gives the following list of tests:

- Original test: T_{00} = {$\Updownarrow$(Act, Wr0, Pre);
  $\Updownarrow$(Act, Wr1, Nop, Nop, Nop, Wr0, Pre,
  Act, Rd0, Pre)}
- Add 1 Nop: T_{00}^{+} = {$\Updownarrow$(Act, Wr0, Pre);
  $\Updownarrow$(Act, Wr1, Nop, Nop, Nop, **Nop,** Wr0, Pre,
  Act, Rd0, Pre)}
- Add 2 Nops: T_{00}^{++} = {$\Updownarrow$(Act, Wr0, Pre);
  $\Updownarrow$(Act, Wr1, Nop, Nop, Nop, **Nop, Nop,** Wr0, Pre,
  Act, Rd0, Pre)}
- etc.

Keep in mind that the more Nops are added, the less the impact each additional Nop will have, because the closer the cell voltage gets to V_{dd} the more difficult it becomes to charge the cell higher.

Common industrial tests

The following test is commonly used for detecting an increased strap resistance:

$T_{\text{ind}} = \{\Updownarrow$(Act, Wr0, Pre); M0
$\Updownarrow$(Act, Wr1, Pre,
Act, Rd1, Nop, Nop, Nop, Wr0, Pre,
Act, Rd0, Pre) M1
$\Updownarrow$(Act, Wr1, Pre); M2
$\Updownarrow$(Act, Wr0, Pre,
Act, Rd0, Nop, Nop, Nop, Wr1, Pre,
Act, Rd1, Pre)$\}$ M3

The march element M0 takes care of initializing the whole memory to an initial 0 state; M0 is identical to the first march element of T_{00}. M1 starts with a Wr1 operation to charge the cell up, which is done in the same way by T_{00}. The test then checks the stored value in the cell using a Rd1 to ensure that V_{dd} has been reached, followed by three Nops to ensure full restoration of the stored 1, and then a Wr0 is performed to sensitize the fault. Finally, a read operation detects whether a fault is sensitized. March elements M2 and M3 have the same sequence as M0 and M1, but apply the complementary data to the memory; this part of the test corresponds to T_{11}.

This test applies a modified version of both T_{00} and T_{11}. The modification involves an added sequence of Pre Act Rd operations in M1 and M3. According to the fault analysis performed above, there is no clear benefit for this added Rd in detecting the strap problem. In addition, since T_{11} always has a higher coverage than T_{00}, it is sufficient to perform T_{11} instead of both.

In conclusion, the analysis performed in this chapter recommends reducing the industrially used test with 32 operations to the much reduced test T_{11} with 13 operations, which reduces the test time by 59%.

Industrial evaluation

An experimental version of the strap test has been included into the test flow of a commodity DRAM product in order to evaluate the effectiveness of the tests proposed by our analysis as opposed to the commonly used industrial test for the strap problem (T_{ind}). The experimental version of the strap test employs the following sequence:

$T_{\text{exp}} = \{\Updownarrow$(Act, Wr0, Pre); M0
$\Updownarrow$(Act, Wr1, Nop, Nop, Nop, Wr0, Pre,
Act, Rd0, Pre) M1
$\Updownarrow$(Act, Wr1, Pre); M2
$\Updownarrow$(Act, Wr0, Pre Nop, Nop, Nop, Wr1, Pre,
Act, Rd1, Pre)$\}$ M3

T_{exp} is similar to T_{ind}, but with the sequence "Pre, Act, Rd1" removed from M1 and M3. T_{exp} is made up of the concatenation of the two tests T_{00} and T_{11}. As mentioned earlier in this section, it is sufficient in principle to use T_{11} to detect all cells with a strap problem, as long as the physical high voltage V_{dd} is used to represent the background BG 11. Although it is possible to write physical data for most of the cells in the cell array, it is not possible to do this for parts of the cell array where failing BL pairs are repaired with redundant elements. Therefore, it is important to test for both BG 00 and BG 11, to insure that the most stressful background is used.

T_{ind} and T_{exp} have been included into the test flow for months, and the detection results of both tests have been compared. The comparison shows that the coverage of both tests is identical for those failing cells diagnosed subsequently by failure analysis to suffer exclusively from an increased strap resistance. The comparison also indicates that T_{ind} has a slightly increased fault coverage for cells that suffer from other forms of failure mechanisms (such as sense amplifier mismatch). This means that T_{exp} not only detects problematic cells with an increased strap resistance, it is also more exclusive in detecting less defective cells suffering from other failure mechanisms.

9.4 Optimizing test stresses

This section discusses the problem of test stress (ST) optimization for the strap problem. ST optimization for a test means properly adjusting the STs (timing, temperature and voltage) such that a higher coverage of a given test can be achieved. As discussed in Section 4.4, the application of STs is necessary to eliminate soft faults, without the need to include costly delay time into the test. STs able to eliminate soft faults have the property of being decisive STs, which means that they are able to force a directly detectable fault in *any* memory (defective as well as functional). STs not having this property are referred to as indecisive.

Simulation-based optimization of STs is made possible using the concept of the *critical resistance* (R_{cr}) of a defect [Al-Ars03b, Al-Ars03c]. Table 9.2 lists examples of R_{cr} values generated for the strap resistance problem, for example. R_{cr} is the resistive value of a defect at which the memory starts to show faulty behavior. As discussed in Section 5.4.2, this important piece of information can be used for optimizing any test ST, as follows:

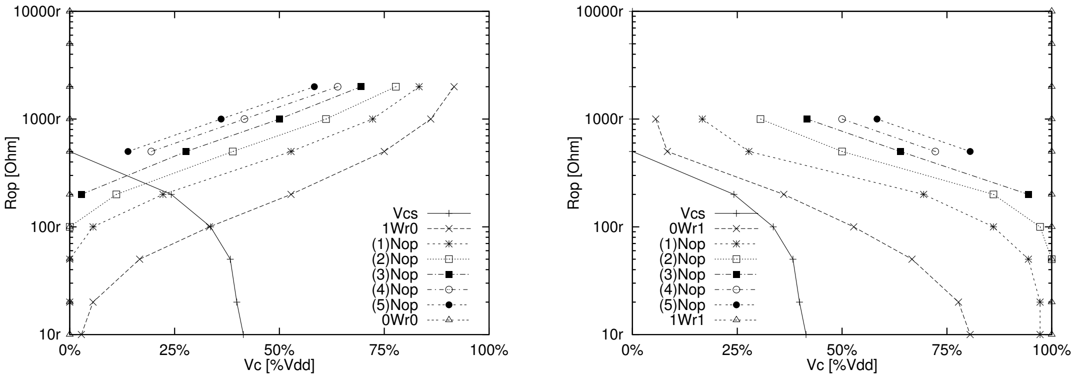
> A change in a given ST should modify the value of the critical resistance in that direction which maximizes the resistance range that results in a detectable functional fault.

In the following, two different STs are analyzed: timing represented by the clock cycle time (t_{cyc}), and temperature. The optimization of voltages is not discussed here, because of the confidential nature of this information [Al-Ars02d].

9.4.1 Optimizing t_{cyc}

In this section, the clock cycle time is optimized by inspecting the impact of a number of t_{cyc} values on the resulting R_{cr}. The fault analysis results of Section 9.2 indicate that R_{cr} is specified by the intersection point of the first Wr command curve and the V_{cs} curve [see Figure 9.4(a), for example]. As a result and in order to identify R_{cr} for every t_{cyc}, it is sufficient to trace the value of R_{cr} by generating the Wr command curve and the V_{cs} curve, and subsequently tracing their point of intersection. In order for this approach to work, however, R_{cr} must be defined by exactly the same two curves across the whole range of relevant t_{cyc} values.

Figure 9.9 shows the result planes with BG 00 in the nominal corner, using $t_{cyc} = 0.5t_{nom}$, where t_{nom} is the nominal clock cycle time, according to the specifications of the memory [Infineon04]. The figure shows that reducing t_{cyc} limits the ability of the memory to charge up or discharge cell voltages. This can be seen in the degradation of the stored cell voltage after performing a write command. On the other hand, the V_{sc} curve remains unchanged, which means that modifying t_{cyc} has no impact on the activation command.



(a) Wr0 (b) Wr1

Figure 9.9. Result planes with BG 00 and $t_{cyc} = 0.5t_{nom}$ for (a) Wr0, and (b) Wr1.

The figure shows that reducing t_{cyc} does not influence the way the critical resistance is defined as the intersection point of the V_{cs} curve and the $1Wr0$ curve. However, as a result of the degradation in the ability of the write command, the value of R_{cr} decreases accordingly. Figure 9.9(a) shows that the critical resistance decreases from $210r\Omega$ to about $100r\Omega$, as a result of the reduction in t_{cyc}.

Now that we have a general picture of the faulty behavior of the memory with nominal t_{cyc} and with a very stressful t_{cyc}, this section traces the value of the critical resistance as a function of t_{cyc}. This shows in detail the dependence of the critical resistance on t_{cyc}, and enables the comparison of the impact of t_{cyc} in different process corners and with different BGs.

Figure 9.10 shows the critical resistance as a function of t_{cyc} in the nominal corner. The x-axis in the figure lists t_{cyc}, while the y-axis in the figure lists the value of R_{cr}. There are four curves in the figure, one for each BG. The first important conclusion one can derive from the figure is that t_{cyc} is a *decisive* ST for the strap problem. This is indicated by the linear decline of R_{cr} toward 0 $r\Omega$ with decreasing t_{cyc}, which means that it is possible to use timing to induce a detectable fault in any cell (defective as well as functional). As a result, t_{cyc} can be used to effectively eliminate the soft fault problem caused by the strap in the memory, without the need to include costly delay time into the test. Analysis of the impact of STs shows that t_{cyc} is a very effective in general to eliminate soft faults [Al-Ars03c].

Furthermore, the following conclusions can be derived from the figure:

- In general, the critical resistances with BG 00 and BG 10 have values that are close to each other. The same is true for the critical resistance values with BG 11 and BG 01.
- As t_{cyc} decreases below its nominal value, the critical resistances with BG 00 and 10 start to deviate in a clear way from those with BG 11 and 01.
- With the exception of very low values of t_{cyc}, the critical resistance with BG 10 and 01 have either a lower or exactly the same value as that with BG 00 and 11, respectively.
- For a given t_{cyc} that is below t_{nom}, BG 11 and BG 01 are more stressful than BG 00 and BG 10.

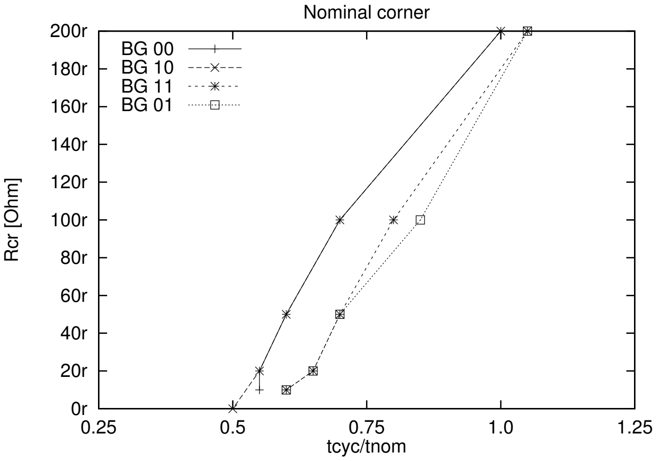
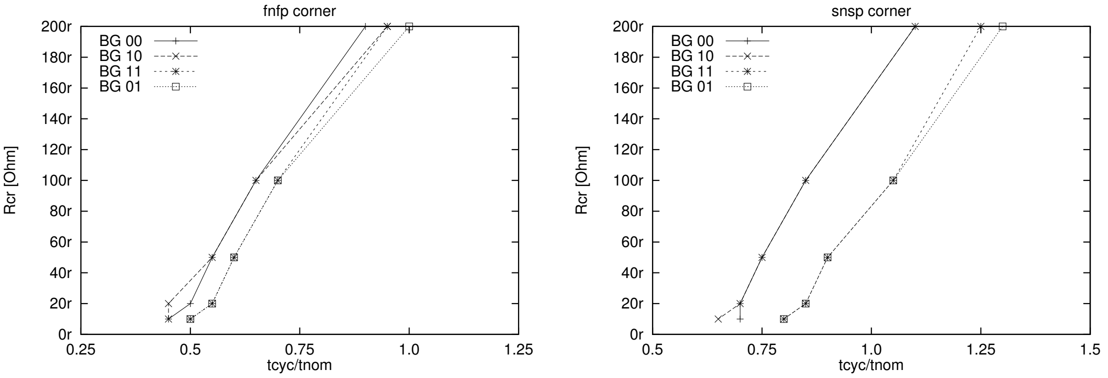


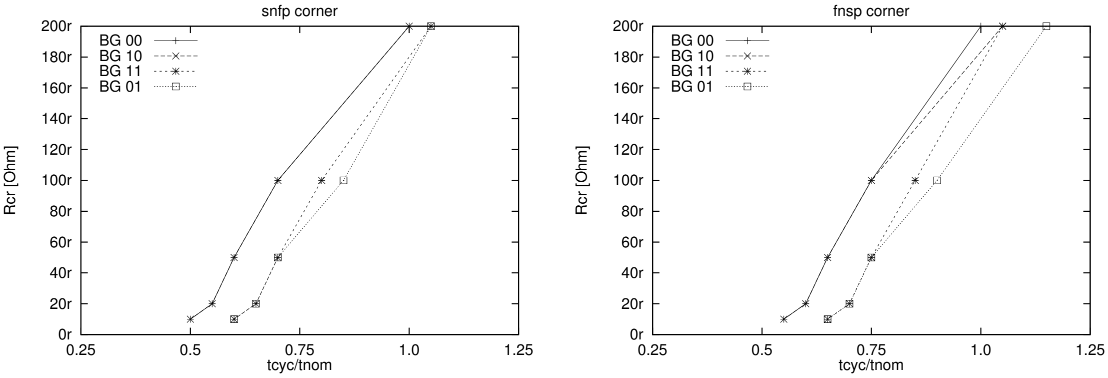
Figure 9.10. Critical resistance as a function of cycle time in the nominal corner.

Figure 9.11 shows the critical resistance as a function of the cycle time in the rest of the process corners, other than the nominal one. The general behavior of the critical resistance in these corners indicates the same tendencies discussed for the nominal corner. The figure shows that, in the fnfp corner, the impact of different BGs on the critical resistances is less than that in the nominal corner. The



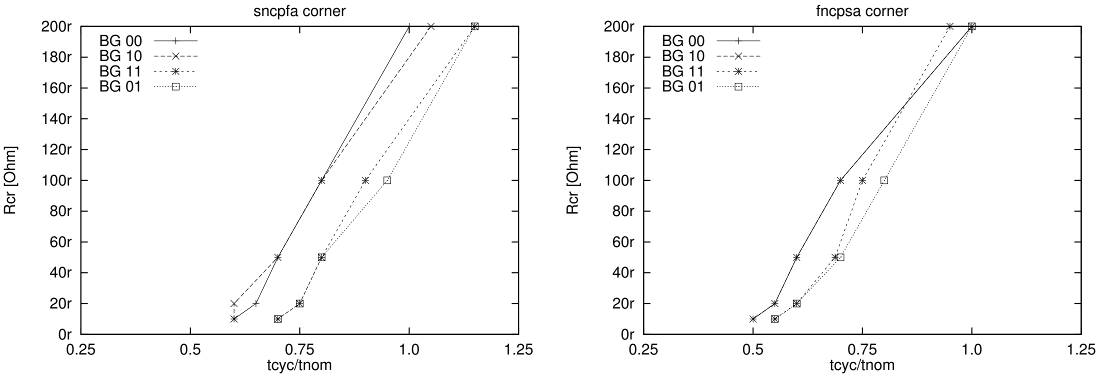
(a) fnfp corner

(b) snsp corner



(c) snfp corner

(d) fnsp corner



(e) sncpfa corner

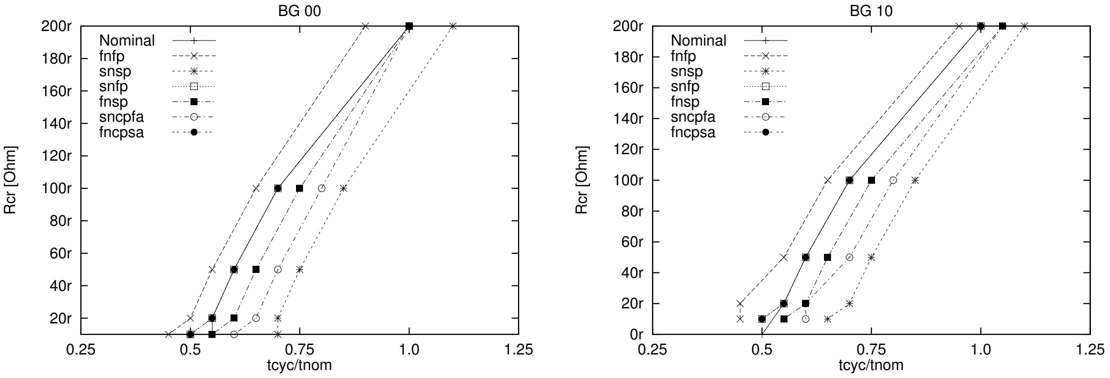
(f) fncpsa corner

Figure 9.11. Critical resistance as a function of t_{cyc} for (a) the fnfp corner, (b) the snsp corner, (c) the snfp corner, (d) the fnsp corner, (e) the sncpfa corner and (f) the fncpsa corner.

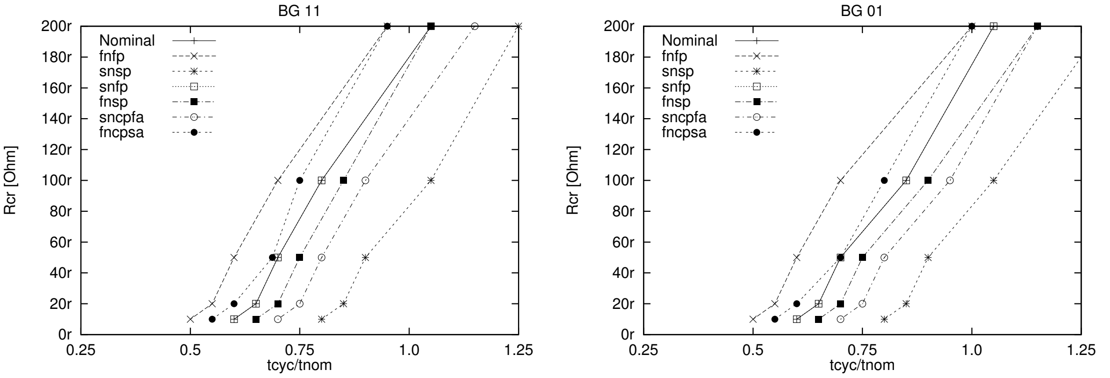
opposite effect takes place in the snsp corner, where the impact of different BGs on the critical resistances increases. In the other corners, the effect of different BGs is similar to their effect in the nominal corner.

Figure 9.12(a) shows, for BG 00, the impact of different process corners on the critical resistance value as a function of t_{cyc}. The figure shows clearly that, for a given t_{cyc}, the fnfp corner results in highest R_{cr} (hence, lowest test coverage), while snsp corner results in lowest R_{cr} (hence, highest coverage). Moreover, the critical resistances in the nominal corner, the snfp corner and the fncpsa corner, not only fall between those of the fnfp and snsp corners, but also have almost the same value for most of the t_{cyc} range.

At the same time, Figure 9.12 shows the critical resistance as a function of cycle time with BGs 10, 11 and 01 for all process corners. The behavior shown in this figure is almost identical to that with BG 00. For a given value of t_{cyc}, the fnfp corner results in the highest value of the critical resistance, while the snsp corner results in lowest value. The critical resistance in all other corners falls between those of the fnfp and snsp corners.



(a) BG 00 (b) BG 10



(c) BG 11 (d) BG 01

Figure 9.12. Critical resistance as a function of t_{cyc} in all process corners with (a) BG 00, (b) BG 10, (c) BG 11 and (d) BG 01.

9.4.2 Optimizing temperature

In this section, an analysis is performed of the temperature effect on the faulty behavior. The objective is to find the temperature that is most stressful for our test. In the nominal corner, discussed in Section 9.3, simulations have been performed using the room temperature of 27°. In this section, the critical resistance (R_{cr}) is evaluated at −50° C, −10° C, 27° C, 87°, 110° C and 150° C. We begin with a discussion of the impact of temperature on the strap resistance then analysis results are presented.

Temperature dependence of strap resistance

In order to optimize temperature with respect to the strap resistance, it is important to include the impact of temperature into the model of the strap. To achieve this goal, measurement results are needed to identify the behavior of the strap at different temperatures.

As shown in Figure 9.1, the strap resistance is modeled as a resistive open between the pass transistor and trench capacitor. Since we perform simulations for different strap resistances, it is important to measure the temperature dependence of the strap using straps that have different resistance values at room temperature.

Such a measurement has been performed on a wafer that has straps with a fairly distributed resistance values ranging from $40r\Omega$ to $400r\Omega$ [Al-Ars02e]. The measured strap resistances are located on 8 dies distributed across the wafer. The measurements have been performed at four different temperatures (−10° C, 27° C, 87° C and 110° C).

The measurements indicate that the value of the strap resistance increases as temperature decreases, which is the expected semiconductor behavior. The measurements also indicate that starting with different strap resistance values at 27° C, the temperature related change in the resistance follows a similar trend. A commonly used model to approximate the temperature dependence of semiconductor resistivity is described by the following relation [Weste94]:

$$R(T) = R(T_0)\left[1 + a \cdot (T - T_0) + b \cdot (T - T_0)^2\right] \tag{9.1}$$

where $R(T_0)$ is the resistance at a given temperature T_0, while a and b are the statistical curve fitting constants. Curve fitting for a and b with respect to the measured resistance data, and using a least square approximation, results in a good curve fit. The model is subsequently programmed into Spice to simulate the temperature dependence of the strap resistance.

Critical resistance trace

In this section, we present the simulation results of the fault analysis performed by modifying the temperature between −50° C and 150° C. The analysis has been performed for all 4 BGs and in all 7 process corners. The analysis is performed with nominal voltages and using $t_{cyc} = 0.75t_{nom}$.

Figure 9.13 shows the R_{cr} value as a function of temperature with all BGs (in the nominal process corner). According to the figure, the value of R_{cr} increases with increasing temperature. This is expected since the value of the defect resistance (R_{op}) decreases with increasing temperature as described in Section 9.4.2 above. This means that a higher temperature reduces the impact of the defect on the faulty behavior, thereby increasing R_{cr}.

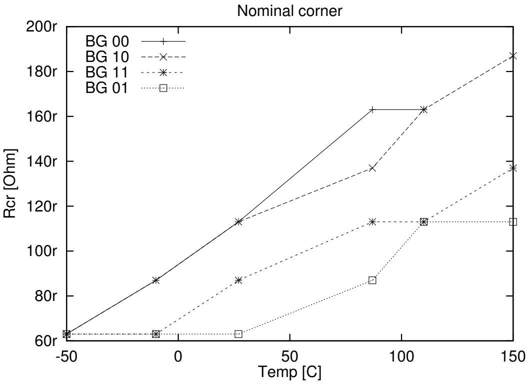
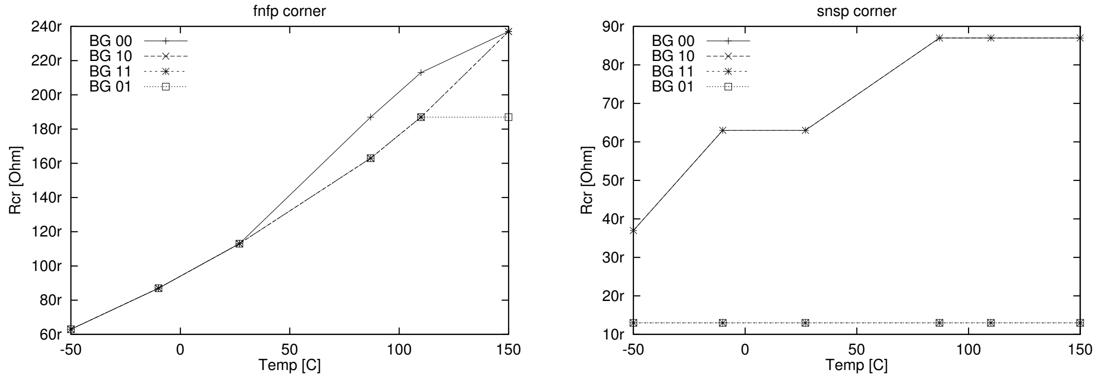


Figure 9.13. Critical resistance as a function of temperature in the nominal corner.

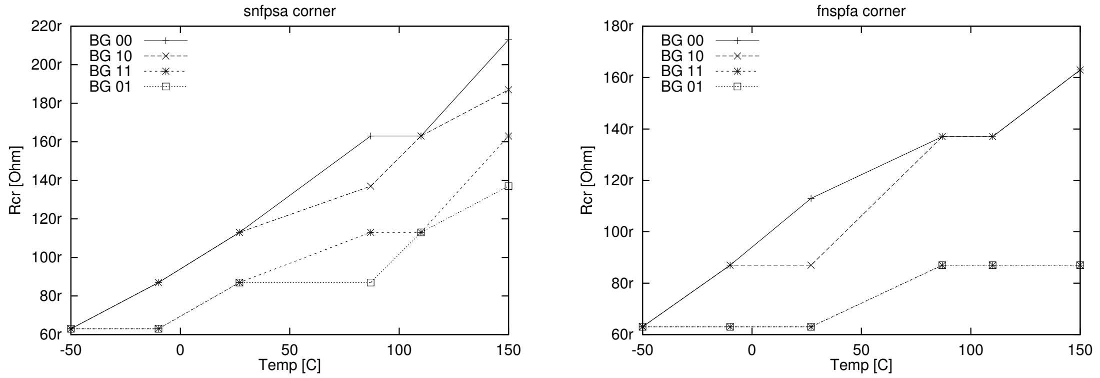
The first important conclusion one can derive from the figure is that temperature is an *indecisive* ST, not suitable in eliminating soft faults from the faulty behavior of the strap. This is indicated by the very slow decline in R_{cr} with decreasing temperature, which reaches a minimum of about 60 $r\Omega$ within the analyzed temperature range. An even lower temperature may result in a lower R_{cr}, but such extreme temperatures are both expensive and unreasonable, falling well beyond the bounds of industrial standards.

The figure shows the same trends observed with the optimization analysis of the cycle time. Two groups of R_{cr} traces can be identified, BG 00 and 10 and BG 11 and 01, where the traces in each group change in the same way and remain close together. In addition, the BGs 10 and 01 are still more effective than their counterparts BG 00 and 11, respectively. The most stressful BG is 01 along the whole analyzed temperature range.

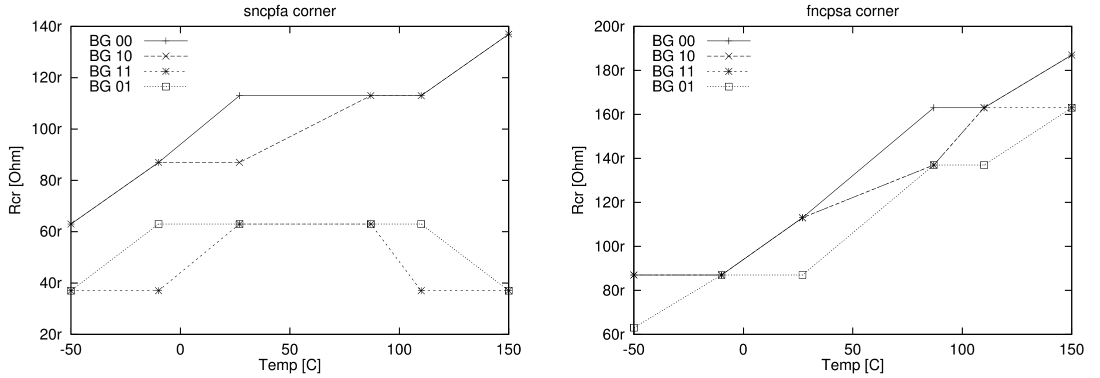
Figure 9.14 shows the critical resistance as a function of temperature in each process corner, with all BGs. The same trends observed in the nominal corner are also present in the other process corners: R_{cr} increases with temperature, R_{cr} for BG 00 and 10 are close together, the same is true for BG 11 and 01, and BG 01 is the most stressful BG in all process corners. The figure also indicates that the difference between R_{cr} with different BGs is smallest in the fnfp corner and largest in the snsp corner. The sncpfa corner shows a particularly interesting behavior with BG 11 and BG 01, where R_{cr} increases between $-50°$ C and $50°$



(a) fnfp corner (b) snsp corner



(c) snfp corner (d) fnsp corner

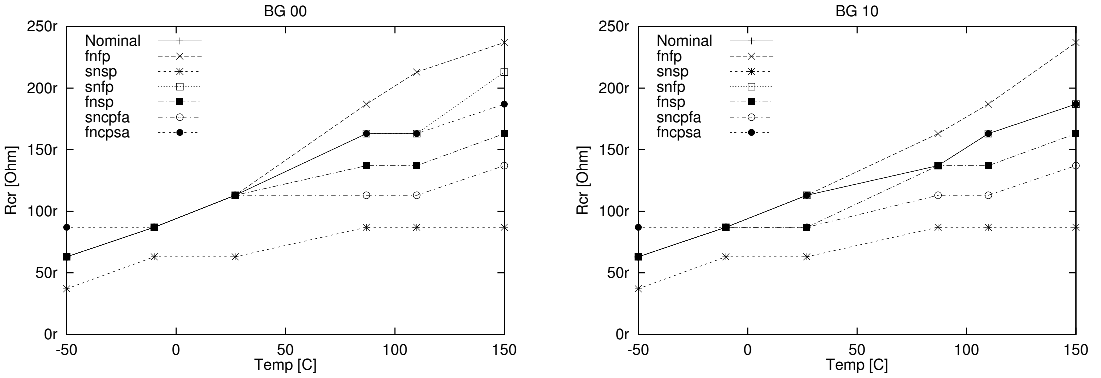


(e) sncpfa corner (f) fncpsa corner

Figure 9.14. Critical resistance as a function of temperature in (a) the fnfp corner, (b) the snsp corner, (c) the snfp corner, (d) the fnsp corner, (e) the sncpfa corner and (f) the fncpsa corner.

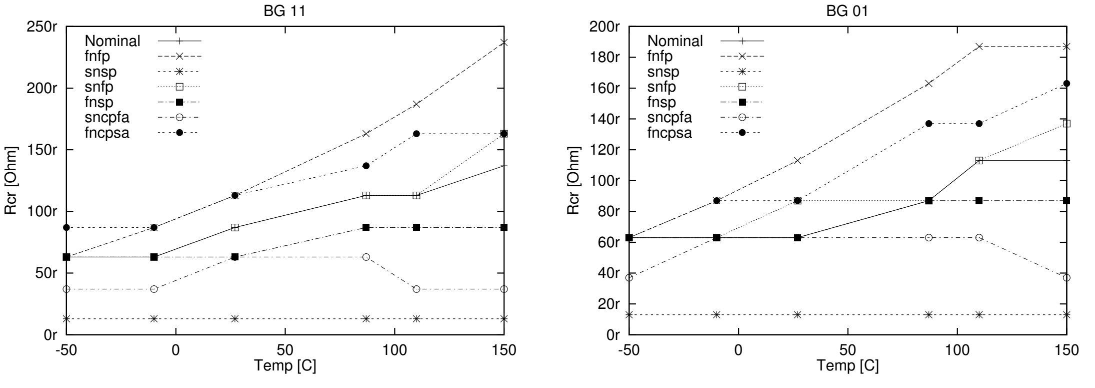
C, and then decreases back between 50° C and 150° C. This means that for this corner and with BG 11 and 01, both high and low temperatures are more stressful than room temperature. This can be attributed to the presence of two independent mechanisms that limit the flow of current into the cell as temperature changes. One of them is the increased strap resistance as temperature decreases, while the other is the gradual reduction of the drain current flowing into the cell as temperature increases.

Figure 9.15(a) shows the critical resistance trace as a function of temperature in all process corners with BG 00. The figure indicates that snsp results in the lowest R_{cr} values while fnfp results in the highest R_{cr} values. The nominal corner results in R_{cr} values that are between the snsp and fnfp corners. The difference between R_{cr} in the snsp and fnfp corners decreases gradually as temperature decreases. Between -10° C and 27° C, R_{cr} has the same value for all process corners with the exception of the snsp corner.



(a) BG 00

(a) BG 10



(b) BG 11

(c) BG 01

Figure 9.15. Critical resistance as a function of temperature in all process corners with (a) BG 00, (b) BG 10, (c) BG 11 and (d) BG 01.

At the same time, Figure 9.15 shows the critical resistance as a function of temperature with BGs 10, 11 and 10 for all process corners. The behavior shown in this figure has the same trends observed with BG 00. The fnfp corner results in the highest value of the critical resistance, while the snsp corner results in lowest value. The critical resistance in all other corners falls between those of fnfp and snsp. An interesting behavior takes place at $-50°$ C and with BG 10 and 11, where R_{cr} has the highest value in the fncpsa corner overtaking the usually highest fnfp corner. The difference is rather small, however.

Summary

This chapter presented a case study to apply the simulation-based fault analysis method in analyzing the faulty behavior of the elevated strap problem. The analysis results make it possible to both generate test patterns to detect the faulty behavior, and to optimize those tests with respect to various memory stresses. The chapter discusses the application of the analysis in detail, starting from the stage of defining an electrical model used in the simulation, through test derivation, till the stage of test application. The main issues introduced in this chapter are the following.

- Evaluation of the elevated strap problem, and derivation of a corresponding electrical Spice model to be injected into the memory model and used in the simulation.
- Selection of an electrical Spice model for the memory that is suitable for the needed type of simulation-based fault analysis.
- Introduction of the concept of fabrication process variations, and identifying a method to model them on the electrical level, and to include them into the fault analysis process.
- Application of the simulation-based fault analysis of the strap problem, which results in the identification of the faulty behavior of the memory in terms of fault primitives (FPs), and results subsequently in deriving suitable memory tests to detect the faulty behavior of the strap.
- Optimization of the tests generated for the strap problem, with respect to two stresses (timing and temperature), in order to increase the fault coverage of the tests and eliminate the problem of soft faults.
- Identification of timing as a *decisive* stress for the strap problem, suitable to eliminate soft faults. Temperature, on the other hand, is identified as an *indecisive* stress, not suitable to eliminate soft faults.

Contents of this chapter

10.1 Conclusions
10.2 Contributions
10.3 Recommendations

10

Conclusions and recommendations

Memory devices generally exhibit a rather complex form of faulty behavior, that is both difficult to analyze and difficult to test for. This thesis discussed the problem of fault analysis and testing of memory devices, and proposed systematic methods to tackle this problem. This chapter summarizes the fault analysis and test results presented in the thesis, lists the contributions of the thesis to the field of memory testing, and presents recommendations to further continue this work in the future.

Section 10.1 starts with a discussion of the major conclusions of the thesis, organized per chapter, and attempts to put the findings in a more general industrial context. Section 10.2 lists the contributions of this thesis, organized in terms of scientific as well as industrial contributions. Section 10.3 presents a vision to implement the scientific theory presented in this thesis in a framework of automated design tools, used to help test engineers in the field to automatically use simulation in the analysis of the faulty behavior of a given defective memory.

10.1 Conclusions

This thesis is mainly concerned with the problem of analyzing the faulty behavior of DRAMs, and in facilitating the implementation of electrical Spice simulation to solve this problem. Chapter by chapter, the thesis presented this problem in detail, and outlined the proposed solutions for it. The main conclusions associated with each chapter of this thesis are listed below.

Chapter 1—Introduction

This chapter presents the fundamental concepts of semiconductor memories in general and memory testing in particular. It discusses the definition of a memory, its interface, and its physical form. A number of different semiconductor memory devices are discussed along with the advantages and disadvantages of each. The chapter identifies the three basic types of memory testing, testing by the manufacturer, testing by the system integrator, and testing by the end user. Among these three parties, the task of applying tests with high fault coverage rests squarely on the shoulders of the memory chip manufacturer. Details are given of the different phases of the industrial test flow, starting from the identification of new tests for a new memory technology, to the final adaptation of the used tests to detect the subtle memory-specific fails. The chapter also describes the major differences between the DRAM and SRAM testing process. The main conclusions of this chapter are:

- DRAMs are the most widely used memories in the market today with a considerable market share that currently stands at about 70%.
- DRAM tests are particularly difficult to construct, since they have to fulfill special requirements with respect to their buildup and abilities.

Chapter 2—DRAM behavior and architecture

This chapter discusses the most important aspects of DRAM behavior and architecture. The discussion starts with a definition of a top-down modeling approach typically used in the design and analysis of complex electronic systems. The chapter then describes the external behavior of the DRAM and the way it is defined using timing diagrams in memory datasheets. For a memory, the external behavior is characterized by the different types of possible memory operations. The chapter continues to describe the functional model of the DRAM, where the internal blocks of the memory are treated. At this level, the internal behavior of the memory is presented and related to the external behavior. As a result, all externally possible memory operations can be mapped to five internal DRAM commands that are more general and more flexible in describing the behavior of the memory. These five different DRAM commands are: activate (Act), write (Wr), read (Rd), precharge (Pre) and no operation (Nop). These DRAM commands can be combined together (under certain conditions) to construct all possible memory operations. The main conclusions of this chapter are:

- DRAMs use specialized internal DRAM-specific commands, that are more general than the simple external write and read operations commonly used to describe DRAM operation.
- A DRAM-specific analysis of the faulty behavior of DRAMs should take into consideration these DRAM commands, rather than the simpler external memory operations.

Chapter 3—DRAM design and implementation

This chapter gives a detailed analysis of the internal structure and implementation of DRAM chips at both the electrical and the layout levels. The electrical description focuses on the data path circuits since they are the most important parts, used to evaluate the faulty behavior of the memory. In total, five circuits are analyzed: the memory cell, the sense amplifier, the precharge circuits, the access devices, and the data buffer. The layout description, on the other hand, focuses on the implementation of the cell and the memory cell array, since it occupies the largest amount of memory surface area. The main conclusions of the chapter are:

- The core functionality of DRAMs is inherently analog in nature, despite the fact that they exhibit a digital external behavior.
- The behavior of a DRAM is heavily dependent on time as a result of the leaky nature of the DRAM storage cell.

Chapter 4—Modeling memory faulty behavior

This chapter presents the fundamentals of modeling faulty behavior of memory devices, both in general and specifically for DRAM devices. The DRAM-specific fault modeling approach presented here reduces the complexity of the general analysis of memory faults and restricts the fault space to those faults relevant to the memory behavior. The chapter starts with a formal definition of the concepts of fault primitives (FPs) and functional fault models (FFMs), the cornerstones of current-day memory fault analysis. This is followed by a discussion of the exponential complexity needed to analyze the full space of dynamic memory faults, as each increment in the number of operations to be analyzed results in a large relative increase in the number of faults a memory should be inspected for. Then, the space of DRAM-specific faults is identified, using five individual fault attributes, which result in 12 different attribute combinations. These 12 attribute combinations modify the behavior of a given generic FP in such a way that describes all DRAM-specific faults. In addition, the FP notation is generalized to take DRAM-specific commands into consideration, along with the stress combinations necessary for DRAM testing. The main conclusions of this chapter are:

- The space of DRAM faults is infinitely large, where a linear increase in the number of analyzed operations requires an exponential increase in the size of the analysis space.
- There are two different types of DRAM-specific faults: 1. those related to improperly set voltages (due to the analog nature of DRAM operation), and 2. those dependent on time (due to leakage current).

Chapter 5—Fault analysis approximation methods

This chapter introduces the simulation-based algorithms used to analyze the faulty behavior of the memory within a practical amount of time. These algorithms solve the two main problems in applying Spice simulation to analyze the faulty behavior of memories: 1. the excessive amount of simulation time needed, and 2. the difficulty to interpret simulation results in terms of faults and tests. This chapter establishes the exponential time complexity of using Spice simulation to precisely analyze the faulty behavior of memories. Then, the 1D approximate simulation method is introduced, which approximates the faulty behavior of the memory, thereby limiting the needed simulation time. This method is subsequently generalized to the 2D method, used to approximate the faulty behavior of more complex defects, such as bit line opens causing a floating cell voltage and a floating bit line voltage. This chapter also extends the approximate simulation method to account for DRAM-specific commands. This is done by identifying a minimum set of DRAM-specific command sequences, that could be used to approximate any possible sequence of commands. The main conclusions of this chapter are:

- The approximate fault analysis simulation method reduces the complexity of the analysis time from exponential to linear, with respect to the number of performed memory operations.
- The approximate simulation method is not only able to generate efficient test patterns (sequences of memory operations), but is also able to optimize those tests with respect to different memory stresses.

Chapter 6—Effects of bit line coupling

This chapter discusses the impact of BL coupling on the faulty behavior of DRAM devices. For some defects, BL coupling has a significant influence on the faulty behavior, that should be taken into consideration when testing the memory for possible faults. The analysis identifies two main causes of BL coupling: 1. pre-sense coupling effects, and 2. post-sense coupling effects. Both of these causes are theoretically discussed and then validated using a simulation-based fault analysis approach of a defective memory, where the simulated faulty behavior is shown to support the theoretical analysis. The chapter also evaluates the effectiveness of two industrial BL twisting techniques (single twist and triple twist) in eliminating the effects of BL coupling, and identifies the way they modify the faulty behavior of the memory. These principles are used in the rest of the thesis to take BL coupling into consideration when realistic tests are derived for the memory. The main conclusions of this chapter are:

- When a memory defect reduces the voltage margins the memory needs to operate within, BL coupling would have a big impact on the faulty behavior of the memory, and should therefore be taken into consideration when designing memory tests.

- When advanced BL twisting techniques are used, analytical evaluation of second-order BL coupling effects on the behavior becomes overly complex, and electrical simulation can provide considerable insight into the faulty behavior of a defective memory.

Chapter 7—Application of the approximation method

This chapter presents the application results of the simulation-based fault analysis algorithms, meant to evaluate the faulty behavior of a defective memory. The results indicate the effectiveness of the proposed fault analysis algorithms, and their ability to describe any DRAM-specific faulty behavior, within a reasonable amount of simulation time. The chapter introduces Spice model reduction techniques, used to reduce the size of a memory simulation model, while keeping the high accuracy of the simulation. These techniques are particularly suited for memory devices, as a result of a number of memory-specific characteristics, related to their design, structure and operation. The chapter also discusses a method to classify the different defects that may take place in the memory into opens, shorts and bridges. The main conclusions of this chapter are:

- The approximate simulation method is effective in the derivation of test patterns to detect the targeted faulty behavior for all the analyzed defects.
- All DRAM-specific faults described in this thesis (Chapter 4) do take place in practice, with most of them commonly observed for many types of defects.

Chapter 8—Space of DRAM tests

This chapter derives the DRAM-specific tests needed to test for all the DRAM-specific fault models introduced in this thesis. Using these tests should give memory test engineers insights into the way fault models should be used to generate memory tests practically and efficiently. The chapter gives a detailed discussion of the space of DRAM-specific faults, using five individual fault attributes, which result in 12 different attribute combinations. These 12 attribute combinations modify the behavior of a given generic FP in such a way that describes all DRAM-specific faults. The full space of faults is then reduced to a set of realistic memory faults that have been observed by simulation in the faulty behavior of a DRAM. Subsequently, six general DRAM tests are proposed to detect single-cell and two-cell faults. The main conclusions of the chapter are:

- Generally derived DRAM tests are efficient in detecting some, but not all, DRAM-specific faults. For soft fault, it is important to use high operational stress in memory tests to reduce the needed test time. Simulations can be effectively used to derive the needed stresses for each test.
- The efficiency of DRAM-specific tests can be increased by customizing them to suit specific memory organizations and designs. Four different customized

tests have been proposed that significantly reduce the complexity of the general tests.

Chapter 9—Case study: the strap problem

This chapter presents a case study to apply the simulation-based fault analysis method in analyzing the faulty behavior of the elevated strap problem. The analysis results make it possible to both generate test patterns to detect the faulty behavior, and to optimize those tests with respect to various memory stresses. The chapter discusses the application of the analysis in detail, starting from the stage of defining an electrical model used in the simulation, through test derivation, till the stage of test application. The chapter also introduces the concept of fabrication process variations, and identifies a method to model them at the electrical level, and to include them into the fault analysis process. The main conclusions of the chapter are:

- The approximate simulation method is effective and practically efficient in both the derivation of suitable test patterns, and the optimization of the test with respect to the various stresses.

- The analysis identifies timing as a *decisive* stress for the strap problem, suitable to eliminate soft faults. Temperature, on the other hand, is identified as an *indecisive* stress, not suitable to eliminate soft faults.

10.2 Contributions

The main contribution of this thesis lies in the introduction of a new industrial test development approach, referred to as *the simulation-based test approach*, which strikes a tradeoff between the two currently established test development alternatives: the specifications and the manufacturing-based test approaches [see Section 1.3]. Figure 10.1 shows a model of the industrial test flow, where the block representing the contribution of the thesis is grayed out (the “Simulation-based” block). The figure indicates that simulation-based test generation uses information from the design stage of the design flow, where an electrical Spice model of the memory is generated to evaluate the expected memory behavior. The Spice model of the memory represents its internal design and behavior, in addition to an electrical description of the fabrication process to be used to manufacture the memory. This provides a fairly accurate representation of the specific behavior of the memory under analysis.

In order to materialize the vision of simulation-based test development for DRAMs, a number of theoretical (scientific) as well as practical (industrial) hurdles have been tackled and solved in this thesis. It is worth noting here that this vision has been *exclusively and completely* carried out within the framework of this

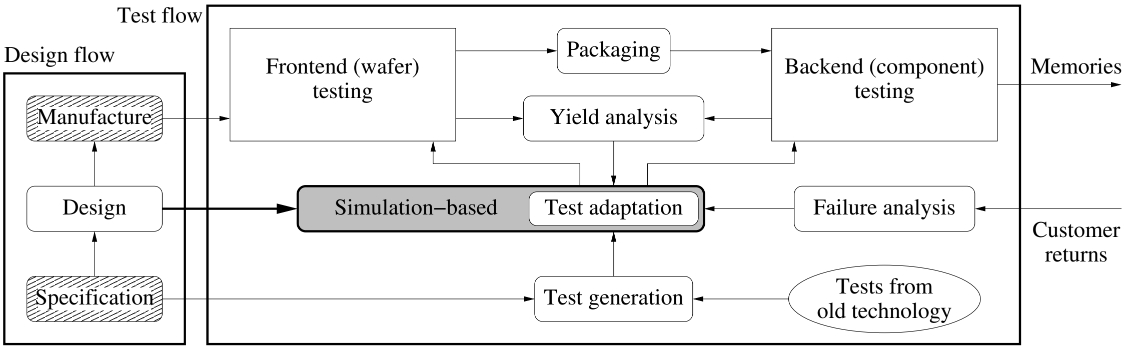


Figure 10.1. Manufacturing test flow where the contribution of this thesis is grayed out.

project, since there is no previously published work on employing electrical simulation for DRAM test development in the literature. First, the scientific contributions are listed, followed by the industrial contributions.

Scientific contributions

- The definition of a general space of memory faults in combination with a taxonomy that describes any possible faulty behavior exhibited by the memory [Al-Ars99, Al-Ars00, Al-Ars01a, Al-Ars03a, vdGoor00]. This general space is treated in Chapter 4.
- The identification of the specific fault classes needed to describe the faulty behavior of DRAMs, such that the rather complex general space of memory faults is reduced to a smaller, manageable size [Al-Ars01b, Al-Ars02a, Al-Ars04a]. These DRAM-specific fault classes are treated in Chapter 4.
- The inclusion of stresses (voltage, timing and temperature) as a theoretically fundamental part of memory testing, and devising a language to model it [Al-Ars01c, Al-Ars01d].
- The analysis of interactions between different memory faults in a way that may limit the ability of memory tests to detect them [Al-Ars04c, Hamdioui03b, Hamdioui04b].
- The theoretical derivation of a number of memory tests to effectively test special types of faulty behavior [Hamdioui02, Hamdioui03a, Hamdioui04c, vdGoor04a, vdGoor04b].

Industrial contributions

- Solving the problem of the long simulation time needed to simulate the faulty behavior of the memory by introducing the concept of a *reduced memory model* [Al-Ars01e]. This issue is treated in Chapter 7.

- Introducing simulation-based fault analysis methods to properly interpret the faulty behavior in the simulation results and to correctly map them into memory faults [Al-Ars02b, Al-Ars02c, Al-Ars03d, Al-Ars05]. This issue is treated in Chapter 5.
- Proposing a simulation-based stress optimization method to use circuit simulation for the identification of the most optimal stresses for a given defect [Al-Ars03c, Al-Ars03b].
- Dealing with simulation model inaccuracy and the issue of variations in the manufacturing process to ensure the high quality fault analysis approach [Al-Ars02d, Al-Ars02e, Al-Ars03e].
- Evaluating the influence of parasitic capacitances in the simulation model, and analyzing the effect of bit line coupling on the simulated faulty behavior for a given defect [Al-Ars04b].

10.3 Recommendations

As discussed in the previous section, the simulation-based fault analysis approach of memory devices presented in this thesis provides a new industrial alternative for fault analysis that strikes a tradeoff between specifications and manufacturing-based test generation. The simulation-based approach provides an alternative that is both moderately cheap and device-specific, since it gives a fairly accurate representation of the specific behavior of the memory under analysis. These advantages make the proposed approach a suitable candidate to be included into the industrial flow of a modern memory manufacturing test environment.

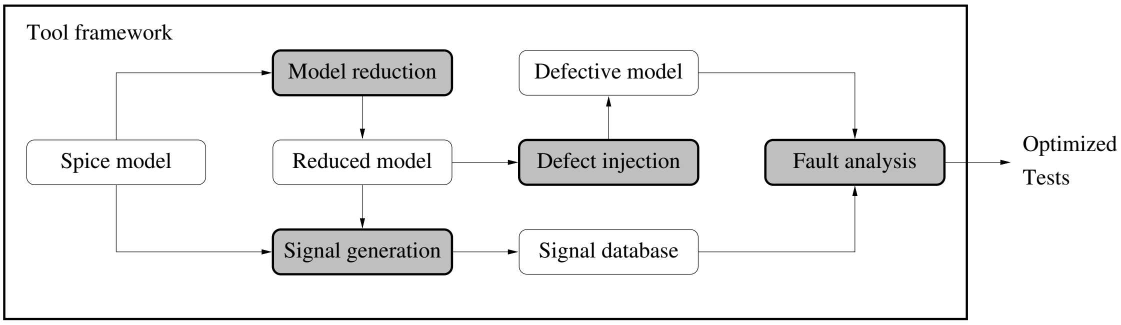


Figure 10.2. Framework of tools to implement the simulation-based test generation approach.

In this section, we suggest a framework of *electronic design automation* (*EDA*) tools to implement the simulation-based fault analysis and test generation approach into an industrial test flow. The framework is shown in Figure 10.2, where a number of steps and tools are identified that would generate the required tests based on an initial Spice simulation model provided by the memory designers. The blocks that represent tools are shaded in Figure 10.2, and are discussed in detail below. In order to bring this framework of tools into the market, a startup company has been established, CatRam Solutions, to supervise the development and implementation activities associated with the framework [CatRam].

Model reduction

The framework starts with an electrical Spice model that describes the various circuits and subcircuits of the memory, and makes it possible to simulate their behavior using an electrical simulator. This model is fed into a “Model reduction” tool that generates a reduced simulation model, which helps eliminate parts of the simulation model that are not needed during the fault analysis process. Model reduction is important to keep the simulation time of the analysis manageable [Naik93]. Model reduction is not necessary if only a short fault analysis is needed.

Signal generation

Since a reduced simulation model is used in the fault analysis, a new set of input signals need to be generated for driving the reduced model [Galarraga98]. These signals are generated from the original spice model using the “Signal generation” tool of Figure 10.2. The generated signals are collected in a database that internally represent the different external operations performed on the memory, in combination with a number of different operational parameters (voltages, temperatures, etc.), as required by the analysis. Again, if no reduction in the simulation model is needed, then there is no need for this step either.

Defect injection

In order to perform the fault analysis on the reduced model, the failure mechanism to be analyzed needs to be modeled as well and injected into the simulation model. The failure mechanism is modeled using some kind of a Spice defect (resistive, capacitive, change in device parameters, etc.) and injected using the “Defect injection” tool in Figure 10.2. This results in a defective simulation model ready for use to apply fault analysis activities on the memory [Müller98].

Fault analysis

The fault analysis itself is performed in the “Fault analysis” tool of Figure 10.2 which generates optimized tests using the reduced, defective simulation model, and using signals from the internal signal database. The algorithms used within this tool depend on the memory to be analyzed and on the type of the required result. A number of algorithms have been suggested in the literature to tackle a number of different problems, such as simulation-based test pattern generation in DRAMs [Al-Ars02b, Al-Ars03d] and in SRAMs [Hamdioui00].

Summary

This chapter presents the conclusions of the thesis and summarizes the most important aspects of this Ph.D. work. This is followed by a discussion of the contributions of this thesis to the field of memory fault analysis and test generation. The chapter then ends with recommendations to further continue with the work outlined here. The main issues discussed in this chapter are the following.

- Summary of the major aspects of each chapter of the thesis. The summary concentrates on specific contributions of this thesis work, that are otherwise not found in the literature.
- Listing of a number of new and significant conclusions in the field of memory fault analysis and test generation, reached within this thesis.
- Enumeration of the major contributions of this thesis to the filed of memory testing. The contributions are classified into two different classes: scientific contributions and industrial contributions.
- Identification of the scientific publications generated in the course of this Ph.D. work. A total number of 26 papers have been published, 3 of which appeared in IEEE transactions journals.
- Proposal of a framework of electronic design automation tools that incorporates the fault analysis methodologies presented in the thesis. The tool framework has four automated steps: model reduction, signal generation, defect injection and fault analysis.
- Suggestion of the approach to incorporate this tool framework into the industrial test flow of memory manufacturers to support and accelerate the fault analysis and test generation activities.

A

List of symbols

Due to the large number of symbols used in this thesis to describe a multitude of new mathematical concepts, we provide the following list of symbols to help the reader identify the meaning of a given symbol in a fast and easy way. The list does not include some symbols that are defined and used specifically in only one section.

β	Transconductance parameter [A/V^2], the change in the drain current of a MOS transistor for a change of one (volt)^2 in the gate voltage.
BW	Bandwidth [Byte/s], the maximum number of bytes a memory can transfer across its data bus per second.
$\text{Cost}_{\text{stage}}$	The cost [$] of testing for a given stage of system integration.
$\#C$	Number of different cells accessed in a memory sensitizing operation sequence.
C_c	Cell capacitance [F], the amount of capacitance a memory cell has.
C_b	Bit line capacitance [F], the total amount of capacitance exhibited by a single bit line in a memory.

GND	Ground voltage [V], the low voltage level of the power supply (usually 0 V).
h	Hammer [operations], the number of times an operation must be performed to sensitize a partial fault.
I_D	Drain current [A], the amount of current that flows through a MOS transistor.
L	Lifetime [s], the time it takes the memory to correct a transient memory fault.
n	The total number of cells a memory has.
$\#O$	Total number of memory operations performed in a sensitizing operation sequence.
R_{br}	Bridge resistance [Ω], the ohmic resistance of an electrical bridge defect.
R_{cr}	Critical resistance [Ω], the ohmic defect resistance value when the circuit starts to fail.
R_{def}	Defect resistance [Ω], the ohmic resistance of any electrical resistive defect.
R_{op}	Open resistance [Ω], the ohmic resistance of an electrical open defect.
R_{sh}	Short resistance [Ω], the ohmic resistance of an electrical short defect.
T	Delay time [s], the amount of time needed after a sensitizing operation sequence for a soft fault to be sensitized. Temperature [^{o}], the operational temperature of a circuit.
t_{cyc} or t_{CK}	Clock cycle time [s], the time a clock signal takes to complete one cycle.
t_{DH}	Data hold time [s], the amount of time input data should remain on the data bus after a write operation starts.

t_{DS}	Data setup time [s], the amount of time input data should be present on the data bus before a write operation starts.
t_{IH}	Input hold time [s], the time needed for control signals to remain on the command bus after a command starts.
t_{IS}	Input setup time [s], the time needed for control signals to be present on the command bus before a command starts.
t_{OX}	Oxide thickness [m], the thickness of the isolation layer between the gate and the body of a MOS transistor.
t_{RAS}	Row address strobe time [s], the time allowed for a row of memory cells to stay connected to the data path in a memory.
t_{RC}	Row cycle time [s], the total amount of time a memory operation may last.
t_{RCD}	Row-column delay time [s], the amount of time needed to separate accessing a full row of memory cells, and addressing a given cell in that accessed row.
t_{RP}	Row precharge time [s], the amount of time needed for the precharge memory command.
t_{WR}	Write recovery time [s], the time needed for a write operation to function properly.
V_{BL}	Bit line voltage [V], the voltage waveform measured on a bit line in the memory.
V_{boost}	Boost voltage [V], the amplified word line voltage used to ensure a good connection between the cell and the data path.
V_c	Cell voltage [V], the voltage stored within a memory cell.
V_{cs}	Cell sense threshold voltage [V], the cell voltage level that distinguishes a stored 0 from a stored 1.

V_{dd}	Supply voltage [V], the high voltage level of the power supply.
V_{DS}	Drain-source voltage [V], the voltage difference in a MOS transistor between the drain and the source terminals.
V_{dsat}	Drain saturation voltage [V], the voltage that separates the linear region from the saturation region of a MOS transistor.
V_{GS}	Gate-source voltage [V], the voltage difference in a MOS transistor between the gate and the source terminals.
V_T	Threshold voltage [V] of a transistor.
V_{WL}	Word line voltage [V], the voltage waveform measured on a word line in the memory.
Y	Yield [%], the fraction of functional IC chips relative to the total number of chips produced by a manufacturing process.

B

List of abbreviations

This is a list of acronyms and abbreviations used in the theses, along with their definitions for easy reference to the reader.

1D	one dimensional
1T	one transistor
2D	two dimensional
A	activation part
a	aggressor
AC	alternating current
Act	activate
AD	addressing direction
B	byte, *also* bulk
b	bit
BC	complement bit line
BG	data background pattern
BL	bit line
BT	true bit line
BW	bandwidth
c	cell
CAS	column address strobe
CD	critical dimension
CF	coupling fault
CFdr	deceptive read destructive coupling fault

CFds	disturb coupling fault
CFir	incorrect read coupling fault
CFrd	read destructive coupling fault
CFst	state coupling fault
CFtr	transition coupling fault
CFwd	write destructive coupling fault
CL	CAS latency
CM	counting method
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
CS	column select
D	drain
DC	complement data line, *also* direct current
DDR	double data rate
Del	delay time
DG	diagonal
DIMM	dual in-line memory module
DLL	delay-locked loop
DRAM	dynamic random access memory
DRDF	deceptive read destructive fault
DT	true data line
EDA	electronic design automation
*e*DRAM	embedded dynamic random access memory
EEPROM	electrically erasable programmable read-only memory
EPROM	erasable programmable read-only memory
F	false
FFM	functional fault model
FP	fault primitive
G	gate
GND	ground voltage
I	initialization part
IC	integrated circuit
I/O	input/output
IP	intellectual property
IRF	incorrect read fault
ITRS	International Technology Roadmap for Semiconductors
L	life time
MOS	metal oxide semiconductor
NMOS	negative-channel metal oxide semiconductor

Nop	no operation
PC	personal computer
PCB	printed circuit board
PMOS	positive-channel metal oxide semiconductor
ppm	parts per million
Pre	precharge
r	read
PROM	programmable read-only memory
RAM	random access memory
RAS	row address strobe
Rd	read
RDC	complement read data line
RDF	read destructive fault
RDT	true read data line
ROM	read-only memory
R/W	read/write
S	source, *also* sensitizing operation sequence
SA	sense amplifier
SC	stress combination
SDRAM	synchronous dynamic random access memory
SEM	scanning electron microscope
SF	state fault
SIMM	single in-line memory module
ST	stress
T	true, *also* temperature, *also* delay time
TBGA	tape ball grid array
TF	transition fault
TSMC	Taiwan Semiconductor Manufacturing Company
TSOP	thin small outline package
UV	ultraviolet
v	victim
w	write
WDC	complement write data line
WDF	write destructive fault
WDT	true write data line
WL	word line
Wr	write
Y	yield

Bibliography

[Abramovici90] A. Abramovici, M. Breuer and A. Friedman, *Digital System Testing and Testable Design*, IEEE press, New York, 1990.

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

[Adler95] E. Adler *et al.*, "The Evolution of IBM CMOS DRAM Technology," *in IBM J. of Research and Development*, vol. 39, no. 1–2, 1995, pp. 167–188.

[Al-Ars99] Z. Al-Ars, *Analysis of the Space of Functional Fault Models and Its Application to Embedded DRAMs*, Masters Thesis no. 1-68340-28(1999)-07, CARDIT, Delft Univ. of Technology, Delft, The Netherlands, 1999.

[Al-Ars00] Z. Al-Ars and A.J. van de Goor, "Impact of Memory Cell Array Bridges on the Faulty Behavior in Embedded DRAMs," *in Proc. Asian Test Symp.*, 2000, pp. 282–289.

[Al-Ars01a] Z. Al-Ars and A.J. van de Goor, "Static and Dynamic Behavior of Memory Cell Array Opens and Shorts in Embedded DRAMs," *in Proc. Design, Automation and Test in Europe*, 2001, pp. 496–503.

[Al-Ars01b] Z. Al-Ars and A.J. van de Goor, "Transient Faults in DRAMs: Concept, Analysis and Impact on Tests," *in Proc. IEEE Int'l Workshop on Memory Technology, Design and Testing*, 2001, pp. 59–64.

[Al-Ars01c] Z. Al-Ars *et al.*, "Simulation based Analysis of Temperature Effect on the Faulty Behavior of Embedded DRAMs," *in Proc. IEEE Int'l Test Conference*, 2001, pp. 783–792.

[Al-Ars01d] Z. Al-Ars, A.J. van de Goor, J. Braun and D. Richter, "A Memory Specific Notation for Fault Modeling," *in Proc. Asian Test Symp.*, 2001, pp. 43–48.

[Al-Ars01e] Z. Al-Ars, A.J. van de Goor, J. Braun, B. Gauch, D. Richter and W. Spirkl, "Development of a DRAM Simulation Model for Fault Analysis Purposes," *in Proc. Workshop on Testmethods and Reliability of Circuits and Systems*, 2001.

[Al-Ars02a] Z. Al-Ars and A.J. van de Goor, "Modeling Techniques and Testing for Partial Faults in Memory Devices," *in Proc. Design, Automation and Test in Europe*, 2002, pp. 89–93.

[Al-Ars02b] Z. Al-Ars and A.J. van de Goor, "Approximating Infinite Dynamic Behavior for DRAM Cell Defects," *in Proc. IEEE VLSI Test Symp.*, 2002, pp. 401–406.

[Al-Ars02c] Z. Al-Ars and A.J. van de Goor, "DRAM Specific Approximation of the Faulty Behavior of Cell Defects," *in Proc. Asian Test Symp.*, 2002, pp. 98–103.

[Al-Ars02d] Z. Al-Ars, *Analysis of the Elevated Strap Resistance Problem*, Internal Technical Report, Infineon Confidential, Infineon Technologies, Munich, Germany, 2002.

[Al-Ars02e] Z. Al-Ars and U. Weber, *Measurement of Temperature Impact on Strap Resistance*, Internal Technical Report, Infineon Confidential, Infineon Technologies, Munich, Germany, 2002.

[Al-Ars03a] Z. Al-Ars and A.J. van de Goor, "Static and Dynamic Behavior of Memory Cell Array Spot Defects in Embedded DRAMs," *in IEEE Trans. on Comp.*, vol. 52, no. 3, 2003, pp. 293-309.

[Al-Ars03b] Z. Al-Ars and A.J. van de Goor, "Test Generation and Optimization of DRAM Cell Defects Using Electrical Simulation," *in IEEE Trans. on CAD*, vol. 22, no. 10, 2003, pp. 1371–1384.

[Al-Ars03c] Z. Al-Ars, A.J. van de Goor, J. Braun and D. Richter, "Optimizing Stresses for Testing DRAM Cell Defects Using Electrical Simulation," *in Proc. Design, Automation and Test in Europe*, 2003, pp. 484–489.

[Al-Ars03d] Z. Al-Ars and A.J. van de Goor, "Systematic Memory Test Generation for DRAM Defects Causing Two Floating Nodes," *in Proc. IEEE Int'l Workshop on Memory Technology, Design and Testing*, 2003, pp. 27–32.

[Al-Ars03e] Z. Al-Ars and A.J. van de Goor, "Analyzing the Impact of Process Variations on DRAM Testing Using Border Resistance Traces," *in Proc. Asian Test Symp.*, 2003, pp. 24–27.

[Al-Ars04a] Z. Al-Ars and A.J. van de Goor, "Soft Faults and the Importance of Stresses in Memory Testing," *in Proc. Design, Automation and Test in Europe*, 2004, pp. 1084–1089.

[Al-Ars04b] Z. Al-Ars, S. Hamdioui and A.J. van de Goor, "Effects of Bit Line Coupling on the Faulty Behavior of DRAMs," *in Proc. IEEE VLSI Test Symp.*, 2004, pp. 117–122.

[Al-Ars04c] Z. Al-Ars, Martin Herzog, Ivo Schanstra and A.J. van de Goor, "Influence of Bit Line Twisting on the Faulty Behavior of DRAMs," *in IEEE Int'l Workshop on Memory Technology, Design and Testing*, 2004, pp. 32–37.

[Al-Ars05] Z. Al-Ars, S. Hamdioui, G. Mueller and A.J. van de Goor, "Framework for Fault Analysis and Test Generation in DRAMs," *in Proc. Design, Automation and Test in Europe*, 2005, pp. 1020–1021.

[Antonin91] G. Antonin, H.-D. Oberle and J. Kolzer, "Electrical Characterization of Megabit DRAMs, 1. External Testing," *in IEEE Design & Test of Computers*, vol. 8 , no. 3, 1991, pp. 36–43.

[Aoki88] M. Aoki *et al.*, "A 60-ns 16-Mbit CMOS DRAM with a Transposed Data-Line Structure," *in IEEE J. Solid-State Circuits*, vol. 23, no. 5, 1988, pp. 1113–1119.

[Baker97] K. Baker and J. van Beers, "Shmoo Plotting: The Black Art of IC Testing," *in IEEE Design and Test of Computers*, vol. 14, no. 3, 1997, pp. 90–97.

[Cataldo98] A. Cataldo, "Players Shift Seats in Quest for 1-Gbit DRAM," *in EE Times*, December 07, 1998, http://www.eetimes.com.

[CatRam] CatRam Solutions, The Computer-Aided Testing Company, Delft, The Netherlands, http://www.catram.com.

[Cheng96] Y. Cheng, M. Chan, K. Hui, M.-C. Jeng, Z. Liu, J. Huang, K. Chen, J. Chen, R. Tu, P.K. Ko and C. Hu, *BSIM3v3 Manual*, Department of EECS, Univ. of California, Berkeley, CA, 1996.

[Dekker90] R. Dekker *et al.*, "A Realistic Fault Model and Test Algorithms for Static Random Access Memories," *in IEEE Trans. on CAD*, vol. C-9, no. 6, 1990, pp. 567–572.

[Falter00] T. Falter and D. Richter, "Overview of Status and Challenges of System Testing on Chip with Embedded DRAMs," *in Solid-State Electronics*, no. 44, 2000, pp. 761–766.

[Foty97] D. Foty, *MOSFET Modeling with Spice, Principles and Practice*, Prentice Hall Inc., New Jersey, 1997.

[Geib92] H. Geib, W. Weber, E. Wohlrab and L. Risch, "Experimental Investigation of the Minimum Signal for Reliable Operation of DRAM Sense Amplifiers," *in IEEE J. of Solid-State Circuits*, vol. 27, no. 7, 1992, pp. 1028–1035.

[Goto97] H. Goto, S. Nakamura and K. Iwasaki, "Experimental Fault Analysis of 1Mb SRAM Chips," *in Proc. IEEE VLSI Test Symp.*, 1997, pp. 31–36.

[Hamada93] M. Hamada, M. Kumanoya, M. Ishii, T. Kawagoe and M. Niiro, "A High-Speed Boundary Search SHMOO PLOT for ULSI Memories," *in Rec. IEEE Workshop on Memory Testing*, 1993, pp. 4–9.

[Hamdioui00] S. Hamdioui and A.J. van de Goor, "Experimental Analysis of Spot Defects in SRAMs: Realistic Fault Models and Tests," *in Proc. Asian Test Symp.*, 2000, pp. 131–138.

[Hamdioui02] S. Hamdioui, Z. Al-Ars and A.J. van de Goor, "Testing Static and Dynamic Faults in Random Access Memories," *in Proc. IEEE VLSI Test Symp.*, 2002, pp. 395–400.

[Hamdioui03a] S. Hamdioui, Z. Al-Ars, A.J. van de Goor, and M. Rodgers, "Dynamic Faults in Random Access Memories: Concept, Fault Models and Tests," *in J. Elecronic Testing: Thoery and Applications*, vol. 19, no. 2, 2003, pp. 195–205.

[Hamdioui03b] S. Hamdioui, Z. Al-Ars, A.J. van de Goor and M. Rodgers, "March SL: A Test for All Static Linked Memory Faults," *in Proc. Asian Test Symp.*, 2003, pp. 372–377.

[Hamdioui04a] S. Hamdioui, *Testing Static Random Access Memories: Defects, Fault Models and Test Patterns*, Kluwer Academic Publishers, Boston, MA, 2004.

[Hamdioui04b] S. Hamdioui, Z. Al-Ars, A.J. van de Goor and M. Rodgers, "Linked Faults in Random Access Memories: Concept, Fault Models, Test Algorithms and Industrial Results," *in IEEE trans. on Computer-Aided Design*, vol. 23, no. 5, 2004, pp. 737–757.

[Hamdioui04c] S. Hamdioui, J.D. Reyes, and Z. Al-Ars, "Evaluation of Intra-Word Faults in Word-Oriented RAMs," *in Proc. Asian Test Symp.*, 2004, pp. 283–288.

[Henderson91] C.L. Henderson, J.M. Soden and C.F. Hawkins, "The Behavior and Testing Implications of CMOS IC Logic Gate Open Circuits," *in Proc. IEEE Int'l Test Conf.*, 1991, pp. 302–310.

[Hidaka89] H. Hidaka *et al.*, "Twisted Bit-Line Architectures for Multi-Megabit DRAMs," *in IEEE J. Solid-State Circuits*, vol. 24, no. 1, 1989, pp. 21–27.

[Galarraga98] D. Galarraga, *Development of a Simulation Model for the Analysis of the Temperature Related Behavior of Embedded DRAMs*, Masters Thesis, University of Central England, Birmingham, UK, 1998.

[IBM02] IBM, *IBM043616CXLBC, 16Mb Double Data Rate SRAM*, Product Datasheet, IBM Microelectronics Division, Hopewell Junction, NY, 2002, http://www.ibm.com.

[Infineon04] Infineon Technologies, *HYB25D512[40/16/80]0B[E/F/C/T], 512Mb Double Data Rate SDRAM*, Product Data Sheet, ver. 1.2, Infineon Technologies, Munich, Germany, 2004, http://www.infineon.com.

[Itoh01] K. Itoh, *VLSI Memory Chip Design*, Springer-Verlag, Berlin, Germany, 2001.

[Iyer99] S.S. Iyer and H.L. Kalter, "Embedded DRAM Technology: Opportunities and Challenges," *in IEEE Spectrum*, vol. 36, no. 4, April 1999, pp. 56–64.

[Jha03] N.K. Jha and S. Gupta, *Testing of Digital Systems*, Cambridge Univ. Press, Cambridge, United Kingdom, 2003.

[Kang96] S.-M. Kang and Y. Leblebici,*CMOS Digital Integrated Circuits: Analysis and Design*, McGraw-Hill, NY, 1996.

[Keshavarzi97] A. Keshavarzi, K. Roy and C.F. Hawkins, "Intrinsic Leakage in Low Power Deep Submicron CMOS ICs," *in Proc. IEEE Int'l Test Conf.*, 1997, pp. 146–155.

[Konishi89] Y. Konishi *et al.*, "Analysis of Coupling Noise between Adjacent Bit Lines in Megabit DRAMs," *in IEEE J. Solid-State Circuits*, vol. 24, no. 1, 1989, pp. 35–42.

[Majhi05] A.K. Majhi *et al.*, "Memory Testing Under Different Stress Conditions: An Industrial Evaluation," *in Proc. Design, Automation and Test in Europe*, 2005, pp. 438–443.

[McConnell98] R. McConnell, U. Moller and D. Richter, "How we test Siemens Embedded DRAM Cores," *in Proc. IEEE Int'l Test Conf.*, 1998, pp. 1120–1125.

[MOSIS] The MOSIS Service, An Integrated Circuit Fabrication Service, Marina del Rey, CA, http://www.mosis.com.

[Muhmenthaler91] H.-D. Oberle and P. Muhmenthaler, "Test Pattern Development and Evaluation for DRAMs with Fault Simulator RAMSIM," *in Proc. IEEE Int'l Test Conf.*, 1991, pp. 548–555.

[Müller98] B. Müller, B. Straube and W. Vermeiren, *Analog Fault Simulator aFSIM-Titan*, User's Manual, Fraunhofer-Institut für Integrierte Schaltungen, Dresden, Germany, 1998.

[Nagi96] N. Nagi and J. Abraham, "Hierarchical Fault Modeling for Linear Analog Circuits," *in Analog Integrated Circuits and Signal Processing*, vol. 10, no. 1–2, June–July 1996, pp. 89–99.

[Naik93] S. Naik, F. Agricola and W. Maly, "Failure Analysis of High Density CMOS SRAMs," *in IEEE Design and Test of Computers*, vol. 10, no. 2, 1993, pp. 13–23.

[Nakamae03] K. Nakamae, H. Ikeda and H. Fujioka, "Evaluation of Final Test Process in 64-Mbit DRAM Manufacturing System Through Simulation Analysis," *in Advanced Semiconductor Manufacturing Conf. and Workshop*, 2003, pp 202–207.

[Niggemeyer99] D. Niggemeyer and M. Rüffer, "Parametric Built-in Self-Test of VLSI Systems," *in Proc. Design, Automation and Test in Europe*, 1999, pp. 376–380.

[Offerman97] A. Offerman and A.J. van de Goor, "An Open Notation for Memory Tests," *in Proc. IEEE Int'l Workshop on Memory Technology, Design and Testing*, 1997, pp. 71–78.

[Prince91] B. Prince, *Semiconductor Memories, A Handbook of Design Manufacturing and application*, 2nd ed., John Wiley & Sons, West Sussex, UK, 1991.

[Prince99] B. Prince, *High Performance Memories, new architecture DRAMs and SRAMs*, Revised ed., John Wiley & Sons, West Sussex, UK, 1999.

[Redeker02] M. Redeker, B.F. Cockburn and D.G. Elliott, "An Investigation into Crosstalk Noise in DRAM Structures," *in Proc. IEEE Int'l Workshop Memory Technology, Design and Testing*, 2002, pp. 123–129.

[Rudolph04] U. Rudolph, E. Weikmann, A. Kinne, A. Henke, P. VanHolt, S. Wege, A. Khan, S. Pamarthy, F. Schaftlein and T. Lill, " Extending the Capabilities of DRAM High Aspect Ratio Trench Etching," *in Proc. IEEE Advanced Semiconductor Manufacturing*, 2004, pp. 89–92.

[Sarpeshkar91] R. Sarpeshkar, J.L. Wyatt, N.C. Lu and P.D. Gerber, "Mismatch Sensitivity of a Simultaneously Latched CMOS Sense Amplifier," *in IEEE J. of Solid-State Circuits*, vol. 26, no. 10, 1991, pp. 1413–1422.

[Schanstra99] I. Schanstra and A.J. van de Goor, "Industrial Evaluation of Stress Combinations for March Tests applied to SRAMs," *in Proc. IEEE Int'l Test Conf.*, 1999, pp. 983–992.

[Schanstra03] I. Schanstra and A.J. van de Goor, "Consequences of RAM Bitline Twisting for Test Coverage," *in Proc. Design, Automation and Test in Europe*, 2003, pp. 1176–1177.

[Simonse98] J.E. Simonse, *Circuit Structures, Design Requirements and Fault Simulation of CMOS SRAMs*, Masters Thesis no. 1-68340-44(1998)-09, CARDIT, Delft Univ. of Technology, Delft, The Netherlands, 1998.

[vdGoor98] A.J. van de Goor, *Testing Semiconductor Memories, Theory and Practice*, ComTex Publishing, Gouda, The Netherlands, 1998, http://ce.et.tudelft.nl/~vdgoor.

[vdGoor99] A.J. van de Goor and J. de Neef, "Industrial Evaluation of DRAM Tests," *in Proc. Design, Automation and Test in Europe*, 1999, pp. 623–630.

[vdGoor00] A.J. van de Goor and Z. Al-Ars, "Functional Memory Faults: A Formal Notation and a Taxonomy," *in Proc. IEEE VLSI Test Symp.*, 2000, pp. 281–289.

[vdGoor04a] A.J. van de Goor, S. Hamdioui and Z. Al-Ars, "Tests for Address Decoder Delay Faults in RAMs due to Inter-Gate Opens," *in Proc. IEEE European Test Symp.*, 2004, pp. 146–151.

[vdGoor04b] A.J. van de Goor, S. Hamdioui and Z. Al-Ars, "The effectiveness of scan test and its new variants," *in Proc. IEEE Int'l Workshop on Memory Technology, Design and Testing*, 2004, pp. 26–31.

[Vollrath97] J.E. Vollrath, "Cell Signal Measurement for High-Density DRAMs," *in Proc. IEEE Int'l Test Conf.*, 1997, pp. 209–215.

[Vollrath00] J. Vollrath, "Tutorial: Synchronous Dynamic Memory Test Construction, A Field Approach," *in Proc. IEEE Int'l Workshop Memory Technology, Design and Testing*, 2000, pp. 59–64.

[Vollrath02] J. Vollrath, "Signal Margin Analysis for DRAM Sense Amplifiers," *in Proc. IEEE Int'l Workshop Electronic Design, Test and Applications*, 2002, pp. 123–127.

[Waser03] R. Waser, *Nanoelectronics and Information Technology*, Wiley-VCH, Berlin, Germany, 2003.

[Weste94] N.H.E. Weste and K. Eshraghian *Principles of CMOS VLSI Design, A Systems Perspective*, Addison-Wesley Publishing, 2003.

Index

A

AC 136
access devices 36, 58
Act 39
activate command 39
activation partial fault 86, 162
AD 91
address buffer 36
address bus 28
    see I/O bus
address bus multiplexing 29
    see bus multiplexing
address decoder 35
address order 77
address path 50, 134
    see signal path
address scrambling 64
    see scrambling
addressing direction 91
aggressor (a) 70
    see victim
aggressor complementary fault 140
aging process 13
algorithmic stress 91, 93
alternating current 136
analysis space 99
approximate simulation 101
array column 37
    see bit line pair
array folding 36
array row 37
    see word line
aspect ratio 61

B

B transistor terminal 44
back-to-back mode 86, 172
backend testing 10
    see memory testing
background pattern 134, 146
balance transistor 57
bandwidth 29, 34
basic sequence 199
basic sequences 112
BC 5, 38
behavioral model 26, 27
    see modeling level
Berkeley BSIM3v3 model 45
BG 134, 146
bistable element 5, 55, 59
bit 28
bit line 5, 38
bit line folding 38
bit line pair 38
BL 5, 38
BL coupling 117
BL pair 38
    see cell array column
body (of transistor) 44
body-effect coefficient 45
boost voltage 51
bridge defect 138, 143
BSIM3v3 model 45
BT 5, 38
bulk (of transistor) 44
buried strap 61
burn-in testing 13
    see memory testing
bus multiplexing
    address bus 29
    data bus 28
    time 29
BW 29

byte 28

C

cache memory 6
capacitive coupling 117
CAS 29
case study 193
CatRam 229
CD 62
cell address scrambling 64
    see scrambling
cell array 35
cell array column 37
    see bit line pair
cell array row 37
    see word line
cell charge up 51
cell discharge 53
cell neighborhood 63
cell sense threshold voltage 84
central processing unit 6
CFdr 76
CFds 75
CFir 76
CFrd 76
CFst 74
CFtr 75
CFwd 76
channel (of transistor) 44
channel length (L) 44
channel length modulation 48
channel width (W) 44
charge up behavior 51
clock cycle time 30
clock period 30
CM 91
CMOS 49
CMOS sense amplifier 54
column 37
    see bit line pair
column address 29
    see row address
column address strobe 29
column decoder 58
column gating 58
column select line 36, 58
command 39
command bus 29
    see I/O bus
compatibility 162
complement bit line 5, 38
complement read data line 50
complement write data line 50
complementary fault 139
completing operation 81
component 2
component level testing 10
    see memory testing
conclusions 221
conduction region 46
continuous stress 90
control logic 35
control path 50, 134
    see signal path
counting method 91
coupling 117
coupling FP 70
CPU 6
critical dimension 62
critical resistance (R_{cr}) 104, 111, 202
cross-coupled SA 54
cross-shaped DRAM region 60
crosstalk noise 117
CS 58
customer returns 9, 17
cutoff region 46

D

D transistor terminal 44
data background 13
data buffer 36, 59
data bus 28
    see I/O bus
data bus multiplexing 28
    see bus multiplexing
data line 58
data path 50, 134
    see signal path

data pattern 13
data rate 29
data retention fault 85
data retention test 13
data scrambling 39
    see scrambling
data transfer rate 28
data word 28
datasheet 28
DC 136
DDR DRAM 3
deceptive read destructive CF 76
deceptive read destructive fault 74
decisive stress 90, 210
deep trench capacitor 61
defect 138
delay-locked loop 32
descrambling 64
design-for-testability 85, 180
detection condition 78, 106, 206
    see march test
device 2
DFT 85, 180
DIMM 3
direct current 136
dirty fault 81, 162
    see DRAM-specific fault
discharge behavior 53
disturb coupling fault 75
disturb test 13
DL 58
DLL 32
drain (of transistor) 44
drain current (I_D) 44
drain saturation voltage 46
DRAM 2
    DDR 3
    synchronous 28
DRAM cell 50
DRAM command 39, 92, 199
    activate 39, 200
    no operation 39
    precharge 39
    read 39
    write 39, 200
DRAM module 3
DRAM operation
    fast page mode 41
    read 30
    read modify write 41
    refresh 33, 41
    write 31
DRAM-specific fault 78, 161
    dirty fault 81
    hard fault 83
    partial fault 80
    soft fault 85
    time dependent fault 82
    transient fault 85
    voltage dependent fault 79
DRAM-specific test 166
DRDF 74
dynamic fault 15
dynamic FP 70
dynamic RAM 3
    see static RAM

E

EDA 228
*e*DRAM 62
EEPROM 7
effects of coupling 120
electrical model 27, 49
    see modeling level
electrical schematics 49
electrically erasable PROM 7
    see programmable ROM
electron surface mobility 47
electronic design automation 228
electronic switch 43, 48, 51
embedded DRAM 62
end user 8
environmental stress 91, 94
EPROM 6
equalization transistor 57
erasable PROM 6
    see programmable ROM

F

fab 12
failure analysis 11
fast page mode 33, 41
fault 68
fault activation part 79
fault compatibility 162
fault complement 139
fault detection 14
fault diagnosis 14
fault life time (L) 86
fault localization 14
fault modeling 67
fault primitive 69
fault taxonomy 161
faulty behavior 68
feature size 133
Fermi potential 45
FFM 67, 69
first level memory 6
first silicon 11
flash memory 7
floatingnode 99
floor plan 60
fncpsa corner 198
fnfp corner 197
fnsp corner 197
folded bit line 38
folding 36, 38
form factor 37
FP 69
FP hierarchy 70
frontend testing 10
    see memory testing
functional fault model 67, 69
functional model 26, 34
    see modeling level

G

G transistor terminal 44
gate (of transistor) 44
gate oxide breakdown 51
gate oxide permittivity 47
general fault 88, 163
generic fault 69, 161
geometrical model 27, 60
    see modeling level
guard band 13

H

hammer 80
hammer test 13
hard fault 83, 162
    see DRAM-specific fault
hard fault region 84
hole surface mobility 47

I

I/O bus
    address 28
    command 29
    data 28
IBM 7, 61
IC 2
ideal sense amplifier 54
incorrect read coupling fault 76
incorrect read fault 73
indecisive stress 210
Infineon 1, 2, 33, 60, 61, 133
initialization part 79
    see sensitization part
initialization partial fault 80, 86, 162
input/output (I/O) 2
integrated circuit 2
intellectual property 16
interchanged fault 140
IP 16
IRF 73

L

latch 59
layout model 27, 60
    see modeling level
leakage current 5, 15
level of abstraction 26
Level 1 Spice model 45
Level 8 Spice model 45
life time of fault (L) 86

life-support system 11
linear region 47
logical cell address 64
logical data 39
logical model 27
    see modeling level

M

macro 138
March 1CH 170
March 1CH$_{swp}$ 184
March 1CS 180
March 1CS$_{swp}$ 187
March 1CT 173
March 1CT$_{swp}$ 185
March 2CH 170
March 2CS 180
March 2CT 175
March 2CTr 189
march element 77, 209
march test 77, 207
masked ROM 6
MATS+ 77
memory bank 60
memory cell array 35
memory chip manufacturer 8
memory page 33
memory testing 8
    backend 10
    burn-in 13
    component level 10
    frontend 10
    package 13
    postfuse 12
    prefuse 12
    SRAM 14
    wafer level 10
metal oxide semiconductor 43
microprocessor 6
mid-point voltage 55
mission-critical application 11
mixed-level modeling 26
modeling level
    behavioral 26, 27
    electrical 27, 49
    functional 26
    geometrical 27, 60
    layout 27, 60
    logical 27
    structural 27
MOS transistor 43
MOSIS 45
multiplexing 28

N

N-channel MOS transistor 44
neighborhood 63
nested march element 173, 175
NMOS transistor 4, 44
no operation command 39
non-volatile memory 6
Nop 39
number of cells ($\#C$) 69
number of operations ($\#O$) 69

O

one dimensional analysis 101, 122
one transistor cell 51
open bit line 38
open defect 138, 141
operation region (of transistor) 45, 46
operation sequence 13, 68
oxide thickness (t_{OX}) 44

P

P-channel MOS transistor 44
package testing 13
    see memory testing
page 33
part 2
partial fault 80, 162
    see DRAM-specific fault
parts per million 9
pass transistor 48, 51
PC 2
PCB 3
personal computer 2
physical cell address 64

physical data 39
pinch-off point 47
plane of r 103
plane of $w0$ 102
plane of $w1$ 103
PMOS transistor 4, 44
post-sense coupling 119
postfuse testing 12
    see memory testing
ppm 9
Pre 39
pre-sense coupling 119
precharge circuits 57
precharge command 39
precharging and isolation 136
precise simulation 97
prefuse testing 12
    see memory testing
printed circuit board 3
probe 12
process corner 197
process spread 197
process variations 51, 56, 196
programmable ROM 6
    electrically erasable 7
    erasable 6
PROM 6

R

RAM 3
RAS 29
Rd 39
RDF 73
read command 39
read destructive coupling fault 76
read destructive fault 73
read modify write 41
read operation 30
    see DRAM operation
read/write gating 58
realistic fault space 165
recommendations 228
reduced memory model 20, 227
reduced simulation model 135
reflected WL organization
    see physical cell address
refresh counter 36
refresh operation 33, 41
    see leakage current
region
    conduction 46
    cutoff 46
    linear 47
    saturation 47
result plane 102
result space 108
ROM 3
    masked 6
    programmable 6
row 37
    see word line
row address 29
    see column address
row address strobe 29
rule of tens 9

S

S transistor terminal 44
SA 54
saturation region 47
SC 89
scan test 12
scanning electron microscope 63, 194
schematics 49
scrambling
    address 64
    data 39
SDRAM 28
SEM 63, 194
sense amplifier 36, 54
sensitization part 79
    see initialization part
sensitization partial fault 80, 86, 162
sensitizing operation sequence 68
sequence of operations 68
SF 72
Shmoo plot 114
Shockley equations 45

short defect 138, 142
Siemens 56
signal development time 55
signal margin 55
signal path 50, 134
    address path 50
    control path 50
    data path 50
SIMM 3
simulation macro 138
simulation model reduction 134
single-cell FP 70
single-cell static FP 71
single-sided complementary fault 140
sncpfa corner 198
snfp corner 197
snsp corner 197
soft fault 85, 162
    see DRAM-specific fault
soft fault region 85
sorting 32
source (of transistor) 44
speed sorting 32
Spice model
    Level 1 45
    Level 8 45
SRAM 4
SRAM testing 14
    see memory testing
ST 89
stacked capacitor 62
state coupling fault 74
state fault 72
static fault 14
static FP 70
static RAM 4
    see dynamic RAM
strap connection 61, 194
stress 13, 89
    algorithmic 93
    decisive 90
    environmental 94
    indecisive 90
    temperature 92, 215
    timing 92, 211
    voltage 92
stress combination 89
stress optimization 113, 210
structural model 27
    see modeling level
substrate 44
surface mobility 47
switch 43, 48, 51
synchronous DRAM 28
    see DRAM
system integrator 8

T

tape out 19
taxonomy 161
taxonomy of FPs 70
TBGA 3
temperature stress 92, 215
test adaptation loop 19
test components 13
test cost 85
test development
    manufacturing-based 18
    simulation-based 19
    specifications-based 18
test flow 10
test insertion 12
test requirements 14
test space 161
test stress 13, 180
    see stress
testing 8
    see memory testing
TF 73
three dimensional variations 198
three-coupling FP 70
threshold voltage 44, 45
time dependent fault 82, 162
    see DRAM-specific fault
time multiplexing 29
    see bus multiplexing
timing diagram 28, 30

timing generator 35
timing parameter 32
timing stress 92, 211
transconductance parameter 47
transient fault 85, 162
    see DRAM-specific fault
transient fault region 86
transistor 43
transistor operation region 46
transition coupling fault 75
transition fault 73
transmission gate 49
trench capacitor 61, 194
tri-state 137
true bit line 5, 38
true read data line 50
true write data line 50
TSMC 45
TSOP 2
TU Delft 1
two dimensional analysis 107, 153
two dimensional variations 196
two-cell FP 70
two-cell static FP 74
two-coupling FP 70
two-operation dynamic FP 70

U

ultraviolet (UV) light 6

V

victim (v) 70
    see aggressor
victim complementary fault 140
volatile memory 6
voltage dependent fault 79, 161
    see DRAM-specific fault
voltage stress 92

W

wafer 11
wafer level testing 10
    see memory testing
wafer-level organization 63
WDF 73
weak voltage 85
WL 4, 36, 38
word 28
word line 4, 36, 38
Wr 39
write back window 31, 53
write command 39
write destructive coupling fault 76
write destructive fault 73
write operation 31
    see DRAM operation

Y

yield 11

List of publications

Journal papers

1. Z. Al-Ars and A.J. van de Goor, “Static and Dynamic Behavior of Memory Cell Array Spot Defects in Embedded DRAMs,” *in IEEE Trans. on Computers (TC’03)*, vol. 52, no. 3, March 2003, pp. 293–309.
2. S. Hamdioui, Z. Al-Ars and A.J. van de Goor, “Dynamic Faults in Random-Access-Memories: Concept, Fault Models and Tests,” *in Journal of Electronic Testing: Theory and Applications (JETTA’03)*, April 2003, pp. 195–205.
3. Z. Al-Ars and A.J. van de Goor, “Test Generation and Optimization for DRAM Cell Defects Using Electrical Simulation,” *in IEEE Trans. on Computer-Aided Design (TCAD’03)*, vol. 22, no. 10, October 2003, pp. 1371–1384.
4. S. Hamdioui, Z. Al-Ars, A.J. van de Goor and M. Rodgers, “Linked Faults in Random-Access-Memories: Concept, Fault Models, Test Algorithms and Industrial Results,” *in IEEE Trans. on Computer-Aided Design (TCAD’04)*, vol. 23, no. 5, 2004, pp. 737–757.

Conference papers

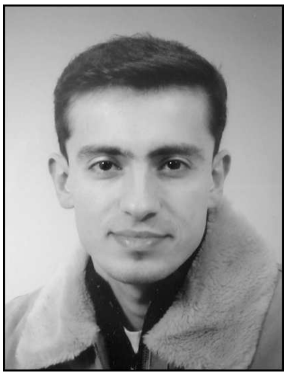
1. A.J. van de Goor and Z. Al-Ars, “Functional Memory Faults: A Formal Notation and a Taxonomy,” *in Proc. IEEE VLSI Test Symp. (18th IEEE VTS’00)*, Montreal, Canada, April 30–May 4, 2000, pp. 281–289.
2. Z. Al-Ars and A.J. van de Goor, “Impact of Memory Cell Array Bridges on the Faulty Behavior in Embedded DRAMs,” *in Proc. Asian Test Symp. (9th ATS’00)*, Taipei, Taiwan, December 4–6, 2000, pp. 282–289.
3. Z. Al-Ars and A.J. van de Goor, “Static and Dynamic Behavior of Memory Cell Array Opens and Shorts in Embedded DRAMs,” *in Proc. Design, Automation and Test in Europe (4th DATE’01)*, Munich, Germany, March 13–16, 2001, pp. 496–503.
4. Z. Al-Ars and A.J. van de Goor, “Transient Faults in DRAMs: Concept, Analysis and Impact on Tests,” *in Proc. IEEE International Workshop on Memory Technology, Design and Testing (9th IEEE MTDT’01)*, San Jose, California, August 6–7, 2001, pp. 59–64.

5. Z. Al-Ars, A.J. van de Goor, J. Braun and D. Richter, “Simulation based Analysis of Temperature Effect on the Faulty Behavior of Embedded DRAMs,” *in Proc. IEEE International Test Conference (32nd IEEE ITC’01)*, Baltimore, Maryland, October 28–November 2, 2001, pp. 783–792.
6. Z. Al-Ars, A.J. van de Goor, J. Braun and D. Richter, “A Memory Specific Notation for Fault Modeling,” *in Proc. Asian Test Symp. (10th ATS’01)*, Kyoto, Japan, November 19–21, 2001, pp. 43–48.
7. Z. Al-Ars and A.J. van de Goor, “Modeling Techniques and Testing for Partial Faults in Memory Devices,” *in Proc. Design, Automation and Test in Europe (5th DATE’02)*, Paris, France, March 4–8, 2002, pp. 89–93.
8. Z. Al-Ars and A.J. van de Goor, “Approximating Infinite Dynamic Behavior for DRAM Cell Defects,” *in Proc. IEEE VLSI Test Symp. (20th IEEE VTS’02)*, Monterey, California, April 28–May 2, 2002, pp. 401–406.
9. S. Hamdioui, Z. Al-Ars and A.J. van de Goor, “Testing Static and Dynamic Faults in Random Access Memories,” *in Proc. IEEE VLSI Test Symp. (20th IEEE VTS’02)*, Monterey, California, April 28–May 2, 2002, pp. 395–400.
10. Z. Al-Ars and A.J. van de Goor, “DRAM Specific Approximation of the Faulty Behavior of Cell Defects,” *in Proc. Asian Test Symp. (11th ATS’02)*, Guam, USA, November 18–20, 2002, pp. 98–103.
11. Z. Al-Ars, A.J. van de Goor, J. Braun and D. Richter, “Optimizing Stresses for Testing DRAM Cell Defects Using Electrical Simulation,” *in Proc. Design, Automation and Test in Europe (6th DATE’03)*, Munich, Germany, March 3–7, 2003, pp. 484–489.
12. Z. Al-Ars, S. Hamdioui and A.J. van de Goor, “A Fault Primitive Based Analysis of Linked Faults,” *in Proc. IEEE International Workshop on Memory Technology, Design and Testing (11th IEEE MTDT’03)*, San Jose, California, July 28–29, 2003, pp. 33–39.
13. Z. Al-Ars, A.J. van de Goor, “Systematic Memory Test Generation for DRAM Defects Causing Two Floating Nodes,” *in Proc. IEEE International Workshop on Memory Technology, Design and Testing (11th IEEE MTDT’03)*, San Jose, California, July 28–29, 2003, pp. 27–32.
14. Z. Al-Ars and A.J. van de Goor, “Analyzing the Impact of Process Variations on DRAM Testing Using Border Resistance Traces,” *in Proc. Asian Test Symp. (12th ATS’03)*, Xi’an, China, November 17–19, 2003, pp. 24–27.
15. S. Hamdioui, Z. Al-Ars, A.J. van de Goor and M. Rodgers, “March SL: A Test for All Static Linked Memory Faults,” *in Proc. Asian Test Symp. (12th ATS’03)*, Xi’an, China, November 17–19, 2003, pp. 372–377.
16. Z. Al-Ars and A.J. van de Goor, “Soft Faults and the Importance of Stresses in Memory Testing,” *in Proc. Design, Automation and Test in Europe (7th DATE’04)*, Paris, France, February 16–20, 2004, pp. 1084–1089.

17. Z. Al-Ars, S. Hamdioui and A.J. van de Goor, "Effects of Bit Line Coupling on the Faulty Behavior of DRAMs," *in Proc. IEEE VLSI Test Symp. (22nd IEEE VTS'04)*, Napa, California, April 25–29, 2004, pp. 117–122.
18. A.J. van de Goor, S. Hamdioui and Z. Al-Ars, "Tests for Address Decoder Delay Faults in RAMs due to Inter-Gate Opens," *to appear in Proc. European Test Symp. (9th IEEE ETS'04)*, Corsica, France, May 23–26, 2004.
19. Z. Al-Ars, Martin Herzog, Ivo Schanstra and A.J. van de Goor, "Influence of Bit Line Twisting on the Faulty Behavior of DRAMs," *in Proc. IEEE International Workshop on Memory Technology, Design and Testing (11th IEEE MTDT'04)*, San Jose, California, August 9–10, 2004, pp. 32–37.
20. A.J. van de Goor, S. Hamdioui and Z. Al-Ars, "The Effectiveness of Scan Test and Its New Variants," *in Proc. IEEE International Workshop on Memory Technology, Design and Testing (11th IEEE MTDT '04)*, San Jose, California, August 9–10, 2004, pp. 26–31.
21. S. Hamdioui, J. D. Reyes, and Z. Al-ars, "Evaluation of Intra-Word Faults in Word-Oriented RAMs," *in Proc. Asian Test Symp. (13th ATS'04)*, Kenting, Taiwan, November 15–17, 2004, pp. 283–288.
22. Z. Al-Ars, S. Hamdioui, G. Mueller and A.J. van de Goor, "Framework for Fault Analysis and Test Generation in DRAMs," *in Proc. Design, Automation and Test in Europe (8th DATE'05)*, Munich, Germany, March 7–11, 2005, pp. 1020-1021.
23. S. Hamdioui, R. Wadsworth, A.J. van de Goor and Z. Al-Ars, "Impact of Stresses on the Fault Coverage of Memory Tests," *to appear in Proc. IEEE International Workshop on Memory Technology, Design and Testing (13th IEEE MTDT'05)*, Taipei, Taiwan.

DRAM Fault Analysis and Test Generation

Curriculum vitae of the author



Zaid Al-Ars was born on October 28, 1974 in Baghdad, Iraq. In 1993, he received his high school diploma from the Kuwait Private School in Farwaniya, Kuwait, where he ranked 13th on the country-wide list of graduates scoring an average grade percentage of 98.3%. Upon completing his high school studies, he enrolled in the BSEE program at the Technical University of Budapest, Budapest, Hungary, where he received the Student Certificate of Merit twice and was granted a study scholarship for his academic achievements. In 1995, he joined the MSEE program at the Delft University of Technology, Delft, The Netherlands, where he majored in computer engineering and did his masters project in the field of memory testing in association with Siemens Semiconductors, Munich, Germany. He received the MSEE degree with honors (cum laude) in 2000, and finished his graduation project with a full mark of 10. In the same year, he received a grant from Infineon Technologies, Munich, Germany, to start his doctoral studies in electrical engineering at the Laboratory of Computer Engineering in Delft. His Ph.D. work was mainly carried out at Infineon, and has been concerned with the systematic fault analysis, and test generation and optimization for commodity as well as embedded DRAM products.

In 2005, he received the VENI grant from the Netherlands Organization for Scientific Research (NWO) to continue his research in memory testing. The grant is given to the candidates who achieve the highest scores in a country-wide competition among eligible scientists in *all* fields of scientific research.

Mr. Al-Ars published numerous papers in the field of electrical defect simulation, fault modeling and test generation in memory devices, and is part of the review committees of a number of IEEE journals and conferences. He is a member of the IEEE.

DRAM Fault Analysis and Test Generation

Statements to accompany the thesis

1. As memory gradually dominates electronic circuits, and as the costs of memory testing increase, memory testing is set to become the most valuable activity in the production process of these circuits.
2. Simulation of memory faulty behavior is essential to effectively construct memory tests. Test designers need to see the memory fail, just like circuit designers need to see it function.
3. The value of a scientific theory can be measured by the number of phenomena it explains. This Ph.D. work not only explains years of observation, but it sometimes proves it wrong!
4. Acceptance without proof is the fundamental characteristic of religion. Rejection without proof is the fundamental characteristic of science.
5. People are as happy as they make up their mind to be.
6. The best things in life are free.
7. Good enough is never good enough, but often good enough is the best you can do.
8. Success is a journey not a destination.
9. There are no bad teams, only bad leaders. There are no bad companies, only bad managers. There are no bad countries, only bad governments.
10. Without justice, the dream of peace will always remain just that: a dream.
11. You cannot achieve the impossible without attempting the absurd.
12. Never underestimate the power of a committed individual to change the world. Indeed, this is the only thing that ever has.
13. Statements on this list might sound contradictory. No wonder. They describe us, and the wonderful world we live in.

Fout Analyseren en Test Genereren in DRAMs

Stellingen behorende bij het proefschrift (in Dutch)

1. Gegeven dat elektronische geheugens geleidelijk elektronische circuits overheersen, en gezien de continue stijging van de kosten van geheugen testen, wordt geheugen testen de waardevolste activiteit van het productieproces ven deze circuits.

2. De simulatie van het gedrag van defecte geheugens is essentieel om efficiënt testen te construeren. Testontwerpers moeten het geheugen zien falen, net zoals circuitontwerpers het moeten zien functioneren.

3. De waarde van een wetenschappelijke theorie kan worden gemeten door het aantal fenomenen die het verklaart. Dit proefschrift verklaart niet alleen jaren van observatie, maar soms heeft het de observatie verkeerd bewezen!

4. Geloven, zonder bewijs, is het fundamentele kenmerk van een godsdienst. De verwerping, zonder bewijs, is het fundamentele kenmerk van de wetenschap.

5. Mensen zijn zo gelukkig als zij beslissen te zijn.

6. De beste dingen in het leven zijn gratis.

7. Goed genoeg is nooit goed genoeg, maar vaak zal het toch genoeg moeten zijn.

8. Succes is de weg zelf en niet de bestemming.

9. Er zijn geen slechte teams, maar wel slechte leiders. Er zijn geen slechte bedrijven, maar wel slechte managers. Er zijn geen slechte landen, maar wel slechte overheden.

10. Zonder rechtvaardigheid, zal de droom van vrede altijd zo blijven: een droom.

11. U kunt het onmogelijke alleen bereiken door het absurde te proberen.

12. Onderschat nooit het vermogen van een toegewijd individu om de wereld te veranderen. Dit is namelijk het enige dat ooit heeft.

13. De stellingen in deze lijst kunnen tegenstrijdig klinken. Geen wonder. Zij beschrijven ons, en de prachtige wereld waarin wij wonen.