Testability and Fault Tolerance for Emerging Nanoelectronic Memories

Nor Zaidi bin Haron
Testability and Fault Tolerance for Emerging Nanoelectronic Memories

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op woensdag 9 mei 2012 om 12:30 uur

door

Nor Zaidi bin HARON

Master of Science in Microelectronics
University of Newcastle upon Tyne, United Kingdom
geboren te Johor, Maleisië
This thesis has been submitted in partial fulfillment of the requirements for Degree of Doctor of Philosophy at Delft University of Technology (TU Delft), The Netherlands. The study compiled in this thesis was supported by three institutions: (1) Computer Engineering Laboratory, TU Delft, (2) Universiti Teknikal Malaysia Melaka and (3) Ministry of Higher Education of Malaysia.


Subject headings: Resistive random access memories, Memory defects, Quality, Reliability, Defect-oriented test, Memory testing, Design-for-Testability, Fault tolerance, Error correction codes, Double modular redundancy, Interleaving.

Copyright © 2012 by Nor Zaidi bin Haron
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without permission of the author.

Author email: zaidi@utem.edu.my

Typeset by the author with the L\LaTeX\ documentation system.
Printed in The Netherlands by Wöhrmann Print Service.
To my dearly loved wife and our cherished daughters.
Emerging nanoelectronic memories such as Resistive Random Access Memories (RRAMs) are possible candidates to replace the conventional memory technologies such as SRAMs, DRAMs and flash memories in future computer systems. Despite their advantages such as enormous storage capacity, low-power per unit device and reduced manufacturing difficulties, these emerging memories are expected to suffer from high manufacturing defect densities (reducing their quality) and in-field fault rates including clustered faults (reducing their reliability). These defects and faults may occur in any part of the memory system including the memory cell array, peripheral circuits and interconnects. Therefore, developing appropriate schemes to address both quality and reliability challenges is critical for the manufacturability of such devices.

This thesis discusses the quality and reliability improvement for nanoelectronic memories. In order to develop effective schemes for quality improvement, first a framework of possible defects within RRAMs has been defined. Thereafter, defect injection and circuit simulation using an electrical RRAM model have been performed. Besides conventional memory faults, simulation results also show the occurrence of unique faults. The detection of the latter faults cannot be guaranteed with conventional memory test approaches as read operations will produce random values. Therefore, Design-for-Testability (DfT) schemes have been introduced to increase the fault/defect coverage at minimum overhead. In addition, as the faults may behave differently subject to process variations, the DfT schemes are made programmable to track the changes in fault behaviors while targeting the unique faults.

On the other hand, several fault-tolerant (FT) schemes have been proposed to improve the in-field reliability of nanoelectronic memories. First, two FT schemes based on error correction codes (ECCs) have been introduced to tolerate both random and clustered faults in the memory cell array, while optimizing the area overhead and performance penalty. Second, an on-line masking scheme is combined with one of the proposed FT schemes to tolerate faults both in the decoders and the memory array; the decoding process has been modified realizing even smaller and faster decoding circuit. Third, an interleaving scheme is combined with an ECC to tolerate faults in the interconnects at minor area overhead and performance penalty.
Samenvatting (Abstract in Dutch)

Opkomende nano-elektronische geheugens, zoals Resistive Random Access Memories (RRAMs) zijn mogelijke kandidaten om conventionele geheugentechologien, zoals SRAMs, DRAMs en flash-geheugens in toekomstige computersystemen te vervangen. Ondanks hun voordelen, zoals enorme opslagcapaciteit, laag vermogen verbruik per eenheid en geringe productie problemen, zullen deze opkomende geheugens naar verwachting problemen ondervinden van hoge fabricage defectdichtheden (wat hun kwaliteit vermindert) en operationele fouten inclusief cluster fouten (wat hun betrouwbaarheid vermindert). Deze defecten en fouten kunnen in elk deel van het geheugen systeem optreden, inclusief de geheugencel matrix perifeer circuits en interconnects.

Daarom is het ontwikkelen van passende schema’s van cruciaal belang om zowel de kwaliteit en betrouwbaarheid te adresseren voor de producerbaarheid van dergelijke devices. Dit proefschrift beschrijft de verbetering van de kwaliteit en betrouwbaarheid van nano-elektronische geheugens. Om effectieve schema’s te ontwikkelen voor kwaliteitsverbetering, is ten eerste een framework gedefinieerd met mogelijke defecten binnen RRAMs. Daarna zijn defect injectie en circuit simulatie uitgevoerd gebruikmakend van het elektrische RRAM model. Naast conventionele geheugen fouten, tonen de simulatie resultaten ook het voorkomen van unieke fouten aan. De detectie van deze fouten niet kan worden gegarandeerd met conventionele geheugen tests, omdat leesbewerkingen willekeurige waarden produceren. Daarom zijn Design-for-Testability (DfT) schema’s ingevoerd om de fout/defect dekking te verhogen met minimale overhead. Bovendien, doordat fouten zich anders kunnen gedragen als gevolg van proces variaties, worden de DfT schema’s programmeerbaar gemaakt om veranderingen in fout gedrag te volgen, gericht op de detectie van unieke fouten.

Anderzijds zijn verschillende fouttolerante (FT) schema’s voorgesteld om de in operationele betrouwbaarheid van nano geheugen te verbeteren. Ten eerste zijn er twee FT schema’s op basis Error Correction Codes (ECCs) geventroduceerd om zowel willekeurige als geclusterde fouten te tolereren in de geheugen cel matrix, terwijl de oppervlakte overhead en prestatie penalty geminimaliseerd worden. Ten tweede is een on-line masking schema gecombineerd met n van de voorgestelde FT schema’s, om zowel fouten in de decoder als in de geheugenmatrix te tolereren; het decoderingsproces is aangepast en realiseert een nog kleiner en sneller decodeerschakeling. Ten derde, is een interleaving schema met een ECC schema gecombineerd om fouten te tolereren in de interconnect met minimale area overhead en performance penalty.
Acknowledgments

This thesis is a collection of not only hard work, perseverance and continuous efforts in the past four years, but also encouragement, cooperation and support from many people. I would like to take an opportunity to acknowledge these people.

First and foremost, I would like to express my utmost gratitude to my advisor Dr. ir. Said Hamdoui, who nurtures me to become an independent researcher. Said trained me to critically analyze scientific issues, educated me the art of well-organized technical writing, coached me the skill of supervising a graduate research and taught me the skill of peer reviewing. His determination and dedication in achieving perfectness earned my admiration. It has been a great pleasure doing the research under you.

Next, I would like to extend my gratefulness to my promotor Prof. dr. Koen Bertels and the members of PhD examination committee: Rector Magnificus, Prof. dr. G.Q. Zhang, Prof. dr. P.J. French, Prof. dr. C.I.M. Beenakker, Dr. N.P. van der Meijs, Prof. dr. ir. A.-J. van der Veen and Dr. P. Girard. Thank you for your time to read and approve my thesis.

I also would like to thank my past and present office mates: Kazeem Gbolagade, Christos Strydis, Seyab Khan, Mottaqiullah Touil and Mafalda Cortez, who made the office alive with insightful discussion – not only scientific but non-scientific as well. My thank also go to CE friends: Marius Enachescu, George Razvan Voicu, Saleh Saifuddin, Mihai Lefter, Yao Wang, Laiq Hasan, Muhammad Faisal Nadeem, Fakhar Anjam, Imran Ashraf, Chunyang Gou, Roel Meeuws just to name few. May our friendship be blessed and will last forever. Special dedication to Mottaqiullah Touil and Roel Meeuws for translating of my abstract and propositions into Dutch.

Special thanks to Lidwina Tromp for her administrative assistance, and Erik de Vries and Eef Hartman for their technical assistance. Not to forget Dr. ir. Arjan van Genderen and Dr. ir. Georgi Gaydadjiev, whom I contacted prior to my admission to TU Delft.

I would like to acknowledge the Ministry of Higher Education of Malaysia and Universiti Teknikal Malaysia Melaka (UTeM), Malaysia for their financial support throughout my studies. My recognition also goes to the management
of UTeM especially to the Vice Chancellor Prof. Datuk Dr. Ahmad Yusoff Hassan, the Dean of Faculty of Electronics and Computer Engineering Assoc. Prof. Dr. Abdul Rani Othman and staffs of Study Leave Division.

I also would like to express my sincerest gratefulness to Malaysian colleagues and their family for friendship and enjoyable gatherings: Ahmad Riza Ghazali, Zulkarnain Abdul Rahman, Zahiraniza Mustaffa, Hazril Aswad Abdul Hamid, Imaduddin Abdul Halim, Anuar Md Ali, Mohd Shahrizal Ab Razak, Nik Shahman Nik Ahmad Ariff, Mohamad Faiz Mohd Amin, Nor Hakimin Abdullah, Mohamad Asrul Mustafar, Shahrizal Ide Moslin, Mohd. Zamri Shah Mastor, Sarah Suib, Wan Nurul Karimah Wan Ahmad, Hafiz Hassan, Mohamad Adib Sarjari, Rigel Lim Liang Jin and Hann Woei Ho. Special appreciation to Safiza Suhana Kamal Baharin and Yahaya Abd. Rahim for their hospitality when my family and I first arrived in Delft. Special dedication to Nik Shahman Nik Ahmad Ariff for designing the thesis cover. May our friendship will not be ended in Delft.

Lastly, no word can describe my indebtedness to my wife Norihan Abu Nawar, and daughters Diya Safiah and Diya Sarah for their understanding, support and patience. I am pretty sure that the experiences we gained over these years will make our love more strong. Equally to my parents Siti Asiah Mesara, Halimah Buyong and Abu Nawar Mat Arus as well as family members in Malaysia who always pray for our success. Thanks also to my late father Haron Abdullah (may his gentle soul rest in peace).

There are some I may have missed, but they are no less appreciated! Many thanks to all!

Nor Zaidi bin Haron Delft, The Netherlands, April 2012
# Table of Contents

Abstract ................................................................. i

Samenvatting (Abstract in Dutch) ................................. iii

Acknowledgments .......................................................... v

Part I Introductory ......................................................... 1

1 Introduction .......................................................... 3
  1.1 Memory technologies ............................................ 4
     1.1.1 Memory concept ........................................ 4
     1.1.2 Memory technology classification .................... 5
     1.1.3 Conventional memory technologies .................. 6
     1.1.4 Emerging memory technologies ....................... 9
  1.2 Potentials and challenges of RRAM .......................... 11
     1.2.1 Potentials .............................................. 11
     1.2.2 Challenges ............................................. 14
  1.3 Research contributions ....................................... 16
  1.4 Thesis organization .......................................... 18
  1.5 Summary ....................................................... 20

2 RRAM Architecture .................................................... 21
  2.1 Memory models ................................................ 22
  2.2 Behavioral RRAM model ...................................... 23
     2.2.1 RRAM block diagram ................................ 23
     2.2.2 RRAM timing diagram ............................... 24
  2.3 Functional RRAM model ...................................... 25
  2.4 Electrical RRAM model ....................................... 26
     2.4.1 Memory cell .......................................... 26
     2.4.2 Peripheral circuits ................................... 29
  2.5 Layout RRAM model .......................................... 32
  2.6 Summary ....................................................... 33
3 Testing and Fault Tolerance Concepts .................................. 35
  3.1 Key terminologies ................................................. 36
  3.2 Testing and fault tolerance in product manufacturing flow .... 37
  3.3 Manufacturing testing ........................................... 40
    3.3.1 Quality testing ............................................. 40
    3.3.2 Reliability testing ......................................... 41
  3.4 Design-for-Testability ........................................... 42
  3.5 Fault-tolerant schemes .......................................... 43
    3.5.1 Redundancy .................................................. 44
    3.5.2 Special design .............................................. 46
  3.6 Summary .......................................................... 47

Part II Fault Modeling and Test for RRAM ................................ 49

4 Defect-Oriented Test for RRAM ........................................ 51
  4.1 RRAM defects classification and definition .................... 52
    4.1.1 Defects in memory cell array .............................. 53
    4.1.2 Defects in CMOS to Non-CMOS Vias ...................... 54
  4.2 Simulation model ................................................. 55
    4.2.1 RRAM functional model .................................... 56
    4.2.2 RRAM cell electrical model ................................ 56
    4.2.3 Logic state definition ..................................... 58
    4.2.4 Memory operation time duration .......................... 58
  4.3 Defect injection and circuit simulation ......................... 60
    4.3.1 Simulation methodology ..................................... 60
    4.3.2 Defect-free simulation result .............................. 62
    4.3.3 Open defect simulation results ............................ 63
    4.3.4 Simulation analysis ........................................ 65
  4.4 Summary .......................................................... 66

5 Design-for-Testability Schemes for RRAM ............................ 67
  5.1 DfT concept ...................................................... 68
  5.2 Short Write Time based DfT ..................................... 70
    5.2.1 DfT circuit .................................................. 72
    5.2.2 Simulation results ......................................... 73
  5.3 Low Write Voltage based DfT .................................... 74
    5.3.1 DfT circuit .................................................. 76
    5.3.2 Simulation results ......................................... 76
  5.4 Programmable DfT concept ....................................... 78
  5.5 Programmable Short Write Time based DfT ....................... 78
  5.6 Programmable Low Write Voltage based DfT ..................... 81
  5.7 Comparison of DfT schemes ..................................... 82
  5.8 Summary .......................................................... 83
8.2.1 RRNS-based architecture 121
8.2.2 Interleaved RRNS-based architecture 122
8.3 Experimental evaluation 124
8.3.1 Experimental setup 124
8.3.2 Fault tolerance capability 124
8.3.3 Area overhead and performance penalty 126
8.4 Summary 126

9 Conclusions and Recommendations 127
9.1 Conclusion of the chapters 127
9.2 Thesis contributions 130
9.3 Recommendations for future research 133

Appendices 135
Bibliography 139
List of Publications 149
Curriculum Vitae 151
Part I: Introductory

This part presents a foundation of the study described in this thesis. Chapter 1 introduces the memory concept and discusses several memory technologies. Chapter 2 describes the architecture of the target memory technology, referred to as Resistive RAM, based on a top-down modeling approach. Chapter 3 gives a brief review of several memory test approaches and fault tolerance schemes.

Chapter 1: Introduction
Chapter 2: RRAM Architecture
Chapter 3: Testing and Fault Tolerance Concept
1.1 Memory technologies
1.2 Potentials and challenges of RRAM
1.3 Research contributions
1.4 Thesis organization
1.5 Summary

A memory chip is an essential unit in any electronic system for data storage, and the rapid development of information and electronic technology demands higher data storage capacity. For more than 50 years semiconductor memories such as SRAMs, DRAMs and Flash have successfully fulfilled the demand. This fulfillment is achieved due to the downscaling of complementary metal oxide semiconductor (CMOS) transistors, typically referred to as Moore’s Law. However, CMOS devices are approaching atomistic and quantum mechanical physics boundaries. As a result, the quality and the reliability of CMOS-based memories will be impacted. Furthermore, a hike in the manufacturing cost and reduction in fabrication accuracy accelerate the end of CMOS usage [1, 2]. In order to ensure that memory devices will still be used in the future, the electronics industry needs to explore new technologies. The near-term solution is to integrate CMOS and non-CMOS technologies such as carbon nanotubes, magnetic tunneling junctions, spintronic and memristors in a single memory chip; this memory technology is referred to as nanoelectronic memories [1, 3, 4, 5, 6, 7]. By leveraging the advantages of both CMOS and non-CMOS technologies, the nanoelectronic memories promise better performance (e.g., denser integration, lower power consumption, faster operation), reduced manufacturing difficulties (e.g., self-assembled, less lithography steps) and economic benefits (e.g., lower production costs).

Defects in memory chips made of the scaled CMOS and tiny non-CMOS technologies are inevitable. Besides permanent faults that emerge from manufacturing defects, nanoelectronic memories are expected to suffer from various intermittent and transient faults during in-field operation [1, 5, 8]. More concerning is that their defect densities and fault rates are expected to be in several
orders of magnitude higher than CMOS technology [5, 8]. These problems lead to quality and reliability issues that may overwhelm the potentials offered by the nanoelectronic memories. Therefore, new memory testing approaches and fault-tolerant schemes are needed to improve the overall quality and reliability of such memories.

This chapter serves as a global introduction of the Ph.D research presented in this thesis. Section 1.1 explains the concept of memory devices including a classification and a brief description of several memory technologies. Section 1.2 discusses the potentials and challenges offered by emerging memories. Section 1.3 presents the contributions of this Ph.D thesis. Section 1.4 gives the organization of its chapters. Finally, Section 1.5 summarizes this chapter.

1.1 Memory technologies

This section provides a brief overview of different types of memory technologies. First, the concept of memory will be presented followed by the classification of the memory technologies. Thereafter, the characteristics of the conventional and emerging memories will be discussed.

1.1.1 Memory concept

As mentioned before, a memory is used to store data for retrieval. There are two operations that can be performed to a memory: (i) a write operation and (ii) a read operation. The write operation is performed to store data into the memory, whereas the read operation is performed to retrieve data from the memory. Figure 1.1(a) depicts a generic block diagram of a memory chip with the essential input/output signal lines, comprising of three input and one output signals. The input signals consist of the Address signal that identifies the selected memory cells for operations, the Write/read signal that controls the write and read operations, and the Data-in signal that carries a written data to be stored. The output signal consists of the Data-out signal that carries a retrieved data.

![Figure 1.1: Schematic of (a) memory device (b) memory cell](image-url)
1.1. Memory technologies

from the memory. Note that the *Address*, *Data-in* and *Data-out* signals are usually composed of several lines, which are referred to as *bus*.

Figure 1.1(b) shows the schematic of a single memory cell inside a memory chip. The memory cell is connected to a word line *WL* and a bit line *BL*; these signal lines provide the connection to the inputs and output signals of the memory. Essentially, *WL* carries the address signals for selecting the memory cell, while *BL* carries the written data into or retrieved data from the selected memory cell.

1.1.2 Memory technology classification

Many types of memories are available for use in electronic systems. The memories can be classified based on their characteristics for desired applications, for example, high performance, bulk data capacity, non-volatile storage, etc. In this thesis, however, the classification is done according to the maturity level of such memories in product development and commercialization.

Figure 1.2 classifies numerous memory technologies into two groups: (i) *conventional* and (ii) *emerging*. The name conventional implies that the memories are mature in development and have been in sold as products for quite some time. On the other hand, the name emerging reflects that the memories are in the early stage of development such that they either entered the market in

![Figure 1.2: Classification of memory technology](image-url)
recent years or are in the prototyping phase. It is worth noting that resistive random access memory (RRAM) is the main focus of this thesis.

1.1.3 Conventional memory technologies

The conventional memory technologies can be further classified into two groups: (i) random access memories (RAMs) and (ii) read-only memories (ROMs). The primary difference between these conventional memory technologies is the capability to retain data in the absence of power supply, which is referred to as volatility. In this context, RAMs are volatile and ROMs are non-volatile. Going further, RAMs can be classified into: (i) static RAMs (SRAMs) and (ii) dynamic RAMs (DRAMs). The main difference between these RAM technologies is the requirement of a periodic refreshment. On the other hand, non-volatile ROMs can be divided into: (i) masked ROMs, (ii) one time programmable ROMs (OTP ROMs), (iii) erasable programmable ROMs (EPROMs), (iv) electrically EPROMs (EEPROMs), and (iv) flash memories. The differences among these ROMs are the erasure mechanism and the number of cycles they can be rewritten (i.e., the endurance capability).

RAMs

This memory type is able to hold the stored data only when it is connected to a power supply. Figure 1.3(a) and (b) depict the electrical structure of an SRAM and DRAM cell, respectively. A typical SRAM cell is formed by four transistors locked together and two access transistors. Conversely, a DRAM cell is constructed from a capacitor along with an access transistor.

**SRAMs.** SRAM typically uses four transistors interlocked together to store data. The four transistors comprise two p-type metal oxide semiconductor

![Figure 1.3: Electrical structure of (a) SRAM and (b) DRAM cells](image_url)
1.1. Memory technologies

(PMOS) transistors and two n-type MOS (NMOS) transistors, creating a latch; see Figure 1.3(a). The latch, which holds two stable states (corresponding to logic 1 and 0), is accessed through the two access transistors made of NMOS. Each access transistor is connected to a true bit line $BL$ and a complement bit line $\overline{BL}$. As long as the SRAM cell is connected to a power supply, the logic data is retained; this property is referred to as static. However, the cell loses its data whenever the power supplied is removed. Compared to DRAMs, SRAMs operate faster but requires bigger area. Therefore, SRAMs are usually used for high-performance and small-density applications such as cache in microprocessors.

DRAMs. DRAMs use a capacitor to store data where the capacitor is charged by connecting it to the power supply and is discharged by connecting it to the ground. This charging and discharging are controlled by the access transistor. The charged capacitor corresponds to logic 1, while the discharged capacitor corresponds to logic 0. Because the charges inside the capacitor are gradually leaked away, DRAMs need to be refreshed periodically; this property is referred to as dynamic. Compared to SRAMs, DRAMs require a smaller area but operate slower. Therefore, DRAMs are usually used for high-density applications such as main memory in computer systems.

ROMs

This memory type is able to hold the stored data permanently even when the power supply is removed. Figure 1.4(a) illustrates the electrical structure of an elementary cell of masked ROMs, whereas Figure 1.4(b) shows the other ROM variants (i.e., OTP ROMs, EPROMs, EEPROMs and flash). Each elementary masked ROM cell consists of a single transistor and a fuse. In contrast, each elementary cell of other ROM variants consists of a single transistor and a floating gate. These non-volatile memories evolve from the hardwired to erasable-and-programmable. Note that the word “programmable” refers to the ability of a memory to perform write operation repeatedly.

![Figure 1.4: Electrical cell structure of (a) masked ROMs (b) other ROM variants](image-url)
**Masked ROMs.** Masked ROMs are a hardwired technology that can be programmed only one time during manufacturing. Thereafter, the stored data cannot be changed. The masked ROMs cell is connected to the ground through a fuse, which can be disconnected by blowing it after the fabrication. If the fuse is blown, then the connection to the ground is missing where this state represents logic 1; otherwise, 0 is represented. Because blowing the fuse is an irreversible operation, reprogramming is impossible. The main advantage of masked ROMs is low production cost when produced in a large quantity.

**OTP ROMs.** OTP ROMs can be programmed only one time by end users using a special equipment commonly referred to as a device programmer. Once OTP ROMs have been programmed, the stored data can never be changed. As a result, OTP ROMs lack reusability limiting its applications.

**EPROMs.** EPROMs can be programmed repeatedly using an electrical source. Prior to programming, the memory must be erased by exposing it to a strong source of ultraviolet light. This type of ROMs is simply recognized by a window in the top of the package, which allows the light to reach the silicon. Although EPROMs can be reprogrammed, such memory is susceptible to light that may lead to an unintentional erase. Moreover, the quartz window package incurs a high production cost.

**EEPROMs.** EEPROMs are introduced as the improvement to EPROMs. In contrast to EPROMs that requires ultraviolet light, EEPROMs can be electrically erased prior to reprogramming. Other advantages of EEPROMs are reprogrammability while in a system, cheap packaging and insensitivity to light. Yet, the main disadvantage is the high production cost due to a complex cell circuit. Therefore, EEPROMs are suitable for, e.g., electronic systems that require non-volatility and in-system reprogrammability.

**Flash.** Flash memories are a variant of EEPROMs that provides the best trade-off between cost and reprogrammability. The “flash” name implies that the erasure process is faster as compared to EEPROMs. Flash memories erase and program data in blocks, whereas EEPROMs in bytes. Furthermore, flash memories are able to store more than one bit per cell, e.g., four states per cell that correspond to two bits of information per cell. Because of these advantages, this memory is becoming the most popular and widely used ROMs in computer and other electronic systems.
1.1.4 Emerging memory technologies

Emerging memories shown in Figure 1.2 consist of numerous types that combine both characteristics of RAMs and ROMs \[1, 6, 7\]. This means that such emerging memories are accessed just like RAMs, yet the stored data is retained even when the power supply is turned off just like ROMs. The main difference with conventional memories is that the storage elements of these emerging memories are fabricated using nanodevices, and neither CMOS transistors nor capacitor. For example, Ferroelectric RAMs (FRAMs) use ferroelectric devices, Magnetoresistive RAMs (MRAMs) and Spin-transfer torque RAMs (STTRAMs) use magnetic devices, Phase-change RAMs (PCRAMs) use phase-change devices, Resistive RAMs (RRAMs) use resistive devices and Organic RAMs (ORAMs) use organic molecules.

**FRAMs.** FRAMs use the electric field and ferroelectric effects to store data \[6, 7\]. Figure 1.5(a) shows the electrical structure of an FRAM cell. Ferroelectric devices (e.g., lead zirconate titanate (PZT)) that form the cell can be configured into two reversible spontaneous polarizations: (i) a parallel polarization and (ii) an antiparallel polarization. The parallel polarization of ions in the material, when in the same direction with the applied electric field, corresponds to logic 1. In contrast, the antiparallel ions polarization that is in the different direction with the applied electric field corresponds to logic 0. The FRAM cell structure resembles a DRAM cell with the exception of plateline \(PL\), which carries variable voltages to realize the polarization switching of ferroelectric capacitor.

**MRAMs.** MRAMs use the magneto-resistive effect of magnetic devices to store data \[6, 7\]. Figure 1.5(b) shows the electrical structure of an MRAM cell. Magnetic devices (e.g., magnetic tunneling junctions (MTJ)) that form the cell can be configured into two distinct configurations: (i) a parallel configuration and (ii) an antiparallel configuration. The configuration of one of the layers can be changed by applying a current induced magnetic field. When both layers are in a parallel configuration (the same direction), the MRAM cell has low resistivity corresponding to logic 1. Conversely, when both layers are in antiparallel configuration (different directions), the MRAM cell has high resistivity corresponding to logic 0.

**STTRAMs.** STTRAMs use the electron spin of spintronics devices to store data \[6, 7\]. This memory technology is a variant of MRAMs introduced to improve the performance. The same layer configurations as in MRAMs determine the logic hold by STTRAMs. However, instead of using current induced magnetic field as in MRAMs, STTRAMs use the spin-polarized current to perform write and read operations. The spin electrons injected from one layer develop a torque to move toward another layer. This mechanism lowers the amount of current needed to write and read the STTRAM cell.
**PCRAMs.** PCRAMs use the reversible phase change properties of chalcogenide devices to store data. Figure 1.5(c) shows the electrical structure of a PCRAM cell [6, 7]. Chalcogenide devices (e.g., germanium-antimony-tellurium (GAT)) that form the cell can be configured into two distinct properties: (i) a crystalline phase and (ii) an amorphous phase. The crystalline phase that corresponds to low resistivity (logic 1) is configured by applying a low-power pulse. On the other hand, the amorphous phase that corresponds to high resistivity (logic 0) is configured by applying a high-power electric pulse with a shorter duration than that of the crystalline phase.

Figure 1.5: Electrical structure of (a) FRAM and (b) MRAM and STTRAM (c) PCRAM (d) RRAM (e) ORAM cell [1]
1.2 Potentials and challenges of RRAM

RRAMs. RRAMs use the resistive effect of transition-metal oxide films to store data [6, 7, 9]. Figure 1.5(d) shows the electrical structure of an RRAM cell. Transition-metal oxide films (e.g., titanium-oxide (TiO$_2$) and nickel-oxide (NiO)) that form the cell can be configured into two distinct configurations: (i) a low-resistive configuration and (ii) a high-resistive configuration. An application of a positive voltage will set the RRAM cell to the low-resistive configuration (logic 1); while a negative voltage will set it to the high-resistive configuration (logic 0). Compared to the other emerging memory technologies, RRAM cells can be fabricated without access transistors. This advantage enables RRAM to be integrated in crossbar arrays and stacked in multiple layers forming 3D memories.

ORAMs. ORAMs use the electron reduction-oxidation (redox) process of organic molecules to store data [6]. Figure 1.5(e) shows the electrical structure of an ORAM cell. Molecules (e.g., catenane and rotaxanes) can be configured into two distinct configurations: (i) an oxidized configuration and (ii) a reduction configuration. The oxidized configuration corresponds logic 1, while the reduction configuration corresponds to logic 0. The main advantage of ORAM is that the cell can be downscaled to a single molecule. However, these organic molecules cannot survive the heat and chemical substances during fabrication process introducing defective ORAM cells.

1.2 Potentials and challenges of RRAM

This section discusses the potentials and challenges of RRAMs, which are the main focus in this thesis. The potentials imply the reason why RRAMs are chosen, while the challenges suggest the research questions addressed in this thesis; see Figure 1.9.

1.2.1 Potentials

The potentials offered by RRAMs are as follows.

1. Enormous data storage capacity

Table 1.1 and Table 1.2 summarize some of the features of several memory technologies, as mentioned in the International Technology Roadmap for Semiconductor version 2011 [1, 9]. The tables show that RRAM potentially to have $37\times$ higher density than DRAM (for comparison to the most dense conventional memory) and $168\times$ higher density than PCRAM (for comparison among emerging memories). This is mainly contributed by the simple RRAM cell element structure, which does not require an access transistor. Furthermore, the ability to store multilevel data persistently (non-volatile) enables RRAM to have even
bigger density. Note that for cell elements feature: $T$ denotes transistor, $C$ denotes capacitor and $R$ denotes resistor.

Table 1.1: Conventional memory technologies

<table>
<thead>
<tr>
<th>Features</th>
<th>SRAM</th>
<th>DRAM</th>
<th>NAND Flash</th>
<th>NOR Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density (Gbit/cm$^2$)</td>
<td>0.17</td>
<td>6.67</td>
<td>2.47</td>
<td>1.23</td>
</tr>
<tr>
<td>Cell element</td>
<td>6T</td>
<td>1T1C</td>
<td>1T</td>
<td>1T</td>
</tr>
<tr>
<td>Multilevel capability</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-volatility</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1.2: Emerging memory technologies

<table>
<thead>
<tr>
<th>Features</th>
<th>FRAM</th>
<th>MRAM</th>
<th>STTRAM</th>
<th>PCRAM</th>
<th>RRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density (Gbit/cm$^2$)</td>
<td>0.14</td>
<td>0.13</td>
<td>0.13</td>
<td>1.48</td>
<td>250</td>
</tr>
<tr>
<td>Cell element</td>
<td>1T1C</td>
<td>1T1R</td>
<td>1T1R</td>
<td>1T1R</td>
<td>1R</td>
</tr>
<tr>
<td>Multilevel capability</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Non-volatility</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

$T$=transistor, $C$=capacitor, $R$=resistor

Figure 1.6: 3D crossbar concept for RRAM [10]
1.2. Potentials and challenges of RRAM

2. Three-dimensional and small form-factor chips

RRAM cells are based on two-terminal nanodevices, which are tinier than CMOS transistors or capacitors in existing semiconductor memories. Moreover, the cells do not require an access transistor as is the case with conventional memories and other emerging memories (except ORAM). Therefore, these simple cells can be fabricated in between two nanowires realizing a crossbar memory cell array. This crossbar structure enables the RRAM cells to be fabricated very close to each other. Additionally, the crossbar-based RRAM memory cell array can be stacked in multiple layers creating a three-dimensional chip, as shown in Figure 1.6 [10]. As the additional circuits (layers) are stacked in the vertical direction, the chip requires a small form-factor (horizontal and flat area).

3. Less complex fabrication of the crossbar memory cell array

RRAM memory cell array is fabricated based on a regular structure of nanowire crossbar, as shown in Figure 1.7(a). This regular structure alleviates the utilization of arbitrary features and the high number of masks as in existing technology. Moreover, the two-terminal devices (e.g., memristor) embedded at each crossbar junction pose only one critical dimension, that is, the vertical direction between the nanowires as illustrated in Figure 1.7(b). This simple feature can be fabricated using small numbers of mask [11, 12, 13, 14]; thus, eliminating the use of complex lithography processes.

4. A leading candidate for universal memory

The ability to store data persistently even in the absence of a power supply, together with enormous data storage capacity allow RRAM to be one of the leading candidates for use as universal memory [15, 16, 17]. Universal memory is the one that possesses the combined attributes of non-volatility of flash, density of DRAM and performance of SRAM. This eliminates the use of mem-

![Figure 1.7](image_url)

Figure 1.7: (a) Image of 17 memristors embedded between crossbar array built at Hewlett-Packard Labs [12] (b) a single memristor within a crossbar junction
Figure 1.8: (a) Existing memory hierarchy (b) universal memory

Figure 1.8: (a) Existing memory hierarchy (b) universal memory

Figure 1.8: (a) Existing memory hierarchy (b) universal memory

Figure 1.8: (a) Existing memory hierarchy (b) universal memory

1. Massive defect density and fault rate.

Because the miniscule nanodevices composing RRAMs are very recent in development and because they lack a reliable fabrication process, the memories are expected to suffer from massive defect density. As forecasted by [5, 18], defect density of such unreliable devices is likely to reach several orders of magnitude higher than CMOS devices. In [19], the researchers claim that a high fault rate up to 15% is expected for wires and interconnects of nanoscale circuit. Besides, these unreliable devices will also be susceptible to faults originated from noise and environmental disturbances during operation [20].
2. **Electrical RRAM models for defect/fault analysis.**

It has been proven in [21, 22] that the electrical models provide an accurate and fast analysis of defects for quality improvement for SRAMs and DRAMs. However, electrical models for RRAM are still unavailable. Therefore, developing an RRAM electrical model is necessary in order to understand the faulty behavior of the RRAM devices in the presence of defects. Such an analysis will enable the development of appropriate fault models and efficient quality test schemes. In addition to the memory test development, various activities can be carried out using the electrical memory model; it encompasses support for test adaptation after yield analysis of manufacturing tests and failure analysis for customer returns [22].

3. **Design-for-Testability schemes for RRAMs.**

Design-for-Testability (DfT) schemes have been implemented for conventional memories in order to reduce test time and improve the defect/fault coverage [23]. This widely used scheme can be further explored for RRAMs to facilitate quality testing. Therefore, suitable DfT schemes could be introduced not only to reduce test time, but also to assist in analyzing the faulty behavior of RRAM.

4. **Fault-tolerant schemes for clustered errors in nanoelectronic memories.**

Due to its dense structure, a nanoscale circuit including RRAMs will be affected by faults that induce clustered errors [24, 25]. When this happens, a group of adjacent cells in the memory array may flip causing clustered errors. Published work on fault-tolerant scheme deals mainly with random errors, which can be ineffective for clustered errors and high fault rate problems. Therefore, fault-tolerant schemes to tolerate clustered errors need to be investigated. Design optimization should also be taken into consideration to minimize the overhead that would arise from the introduction of the fault-tolerant schemes.

5. **Fault-tolerant schemes for memory peripheral circuits and interconnects.**

Most of the published work so far assumes that peripheral circuits and interconnects are reliable. This assumption is no longer valid because even in 130nm CMOS technology, logic circuits have shown their susceptibility to transient faults at a level similar to unprotected memories [26]. In addition, aging-induced faults due to failure mechanisms (such as Negative Bias Temperature Instability (NBTI)) exaggerate this problem especially when operating at the critical operating temperature and frequency. Interconnect scaling and multilevel wire stack introduces crosstalk and latency, affecting the circuit reliability as well [27, 28]. Therefore, designing reliable emerging memories require not only protecting the memory cell array, but also the peripheral circuits and interconnects.
1.3 Research contributions

The research carried out in the course of this Ph.D project is motivated by the RRAM potentials and challenges mentioned in the previous section; see also Figure 1.9. The objective is to improve the quality and reliability of emerging memories such as RRAM. The outcomes of this research are compiled in several scientific publications; they are summarized as follows.

1. Fault models for open defects in RRAMs.

The fault modeling is based on a test framework consisting of a taxonomy of defects, RRAM electrical model and defect simulation. The simulation of open defects that impact a single RRAM cell has resulted in the development of new fault models. This fault modeling is given in Chapter 4 of this thesis and presented at Asian Test Symposium 2011 [29].

2. Two DfT schemes for open defects in RRAMs.

The two DfT schemes are introduced to detect the new fault models and improve the detection of open defects in RRAMs. Each DfT scheme is developed by exploiting the access time duration and the voltage applied to the RRAM cell. The proposed schemes are given in Chapter 5 of this thesis and presented at Design, Automation, and Test in Europe Conference 2012 [30].

3. Two programmable DfT schemes for open defects in RRAMs.

The two programmable DfT schemes ensure the detection of open defects with different values in RRAMs. The basic DfT schemes are extended with multiple access durations and multiple voltage levels, which can be digitally programmed. The proposed schemes are given in Chapter 5 of this thesis and submitted for IEEE Transactions on Computers publication [31].

4. Fault-tolerant schemes for clustered faults/errors in nanoelectronic memory cell array.

The two fault-tolerant schemes are modified from an existing symbol-based error correction code (ECC). In addition to providing a competitive fault tolerance capability as compared with existing ECCs, the first error correction scheme provides smaller memory area overhead, while the second error correction scheme offers higher performance decoding operation. The first scheme is given in Chapter 6 of this thesis and presented at IEEE/ACM International Symposium on Nanoscale Architectures 2009 [32], IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems 2009 [33] and HiPEAC Workshop on Design for Reliability 2011 [34], as well as published in ACM Journal of Emerging
1.3. Research contributions

Emerging memories

- Enormous data storage
- 3D chip
- Relaxed fabrication complexity
- Universal memory candidate

Potentials

- Massive defect densities/fault rates
- Electrical simulation models
- Suitable testing schemes
- Clustered faults/errors tolerance techniques
- Fault-tolerant technique for peripheral circuits
- Fault-tolerant technique for vias

Challenges

- Memory testing and fault tolerance study
- RRAM electrical model and two new fault models
- Design-for-Testability schemes
- Two modified symbol-based ECCs
- A technique combining an ECC and Muller C
- A technique combining an ECC and interleaving

Contributions

Figure 1.9: Mapping of the motivations and contributions of the thesis
Chapter 1. Introduction

Technologies in Computing Systems 2011 [35]. The second scheme is also given in Chapter 6 and presented at IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems 2010 [36] and part of the manuscript submitted for IEEE Transactions on VLSI Systems publication [37].

5. A fault-tolerant scheme for nanoelectronic memory cell array and peripheral circuits.

The fault-tolerant scheme is developed by combining the proposed high-performance fault-tolerant scheme and a masking scheme. In addition to tolerating faults, the error correction circuits are optimized by reversing the decoding sequence; this realizing a smaller area overhead and higher performance of the circuits. This proposed scheme is given in Chapter 7 of this thesis and presented at Design, Automation, and Test in Europe Conference 2011 [38]. This proposed scheme is also part of the manuscript submitted for IEEE Transactions on VLSI Systems publication [37].

6. A fault-tolerant scheme for nanoelectronic memory cell array and vias.

The fault-tolerant scheme is developed by combining a symbol-based ECC and an interleaving scheme. This evaluation is given in Chapter 8 of this thesis and presented at IEEE International Conference on Nanotechnology 2010 [39].

1.4 Thesis organization

This thesis is organized in three parts: introductory, quality and reliability. The introductory and reliability parts consist of three chapters each, whereas the quality part consists of two chapters.

Part I: Introductory

This part serves as the background for understanding the research described in this thesis. The information provided in this part, which consists of Chapter 1, 2 and 3, is the foundation of the topics that will be presented in next two parts. With the exception to those who are equipped with prior knowledge of topics presented, the information presented in this part is important to understand the whole thesis.

Chapter 1 briefly introduces different semiconductor memories and classify into two groups: conventional memories and emerging memories. First, a description of their structures and data storage capability is given. Thereafter, the chapter discusses a number of potentials and challenges of RRAM, which is the main focus of the study in this thesis.
Chapter 2 discusses RRAM architecture in top-down modeling approach. The modeling approach consists of five levels where each level concerns relevant information of the memory. First, the chapter explains the highest level, referred to as behavioral model, which treats the memory as a single black box. Then, it describes the internal blocks that compose the behavioral model. Next, it delineates electrical memory model that illustrates the electrical components and their connections. Finally, the chapter provides the layout model of RRAM. One level, i.e., logical model is not given in detail as it is not common for memory systems.

Chapter 3 presents a short review of several memory testing approaches and fault tolerance schemes. Some of them are adopted in the study, thereby this chapter builds the required basic knowledge for understanding the contribution of this thesis. First, it defines several key terminologies associated with testing and fault tolerance. Then, it discusses the importance of testing and fault tolerance during product manufacture in order to achieve the specified quality and reliability levels. Thereafter, the chapter classifies and briefly describes memory testing, and several Design-for-Testability (DfT) schemes. Finally, it identifies and briefly defines several fault-tolerant schemes.

**Part II: Fault Modeling and Design-for-Testability for RRAM**

This part covers the modeling of faults in RRAM and the memory test based on DfT schemes. These two topics aim to improve the quality of RRAM.

Chapter 4 presents a framework of defect-oriented test and fault modeling for RRAM based on electrical simulation. First, it discusses the classification and definition of possible defects that will occur in RRAM. Then, it introduces the SPICE RRAM model used for defect injection and simulation. Thereafter, it provides the case study of open defects impacting a single RRAM cell including the defect-free and open defect simulation, as well as their fault analysis and modeling.

Chapter 5 introduces DfT schemes to facilitate manufacturing tests in detecting resistive open defects in RRAM. First, it describes the concept of DfT schemes that are based on the access time and supply voltage of the RRAM cell. Then, it provides the methodology used in designing two DfT schemes followed by the circuits. Next, the experimental evaluation of the proposed schemes is discussed. Thereafter, the extended version of the DfT schemes are described including their concept, design methodology and circuits.

**Part III: Fault-tolerant Architecture for Emerging Memories**

This part discusses schemes studied to develop a fault-tolerant architecture; this architecture aims to improve the reliability of emerging memories.

Chapter 6 presents the fault-tolerant schemes used to mitigate faults in the memory cell array. An error correction code (ECC) is adopted, modified and
combined with other fault-tolerant schemes in order to improve the area overhead and performance, without compromising the fault tolerance capability. First, it reviews the circuit of the ECC in an RRAM. Thereafter, it introduces two proposed error correction schemes modified from an existing Redundant Residue Number System (RRNS) code. Finally, a comprehensive simulation results and analysis of the proposed error correction scheme is given; they are compared to the conventional ECCs such as RRNS and Reed-Solomon codes.

Chapter 7 presents the fault-tolerant scheme used to mitigate faults in the memory cell array and decoders. The decoders operate a modified error correction scheme that will be presented in Chapter 6. First, it reviews the architecture of an emerging memory equipped with the basic decoder. Thereafter, it introduces the proposed fault-tolerant decoder, along with a modified decoding procedure for a cost-effective implementation. Finally, it gives the simulation results of the proposed fault-tolerant scheme and the hardware implementation.

Chapter 8 presents the fault-tolerant scheme used to mitigate faults in the memory cell array and vias. First, it explains the CMOS to Non-CMOS Vias concepts in a three-dimensional architecture utilized by RRAM and their defect types. Then, it presents the existing architecture followed by the proposed fault-tolerant scheme. Thereafter, the chapter gives the simulation results and the hardware implementation of both the existing and the proposed fault-tolerant schemes.

This thesis ends with Chapter 9 summarizing the chapters and contributions, followed by a number of recommendations for future research. Appendices provide examples of the encoding and decoding calculation for the considered error correction code and their modified versions.

1.5 Summary

This chapter provided a basic introduction associated with the research presented in this thesis. The main topics covered are as follows.

- Identification of several memory technologies classified – based on the maturity level – into two main groups: conventional and emerging. A brief description of each group is also given in terms of their structure, operations, advantages and disadvantages.
- Clarification of the potentials and challenges of Resistive Random Access Memories.
- Summary of the new scientific contributions of the Ph.D project.
- The organization and a brief description of the content of the chapters of this thesis.
Several nanoarchitectures such as NanoFabrics [41], NanoPLA [42, 43], NASIC [44], nano-BOX [45], CMOL [46, 47], FPNI [48] and CMOS/nano [49, 50] have been proposed as a future architecture for electronic circuits. These nanoarchitectures integrate CMOS and non-CMOS devices. Among these nanoarchitectures, CMOL offers the most promising potentials such as enormous density and heterogeneous applications [46, 47, 51]; CMOL is a hybrid CMOS/nanowire/MOLEcular nanoarchitecture introduced by Likharev and Strukov from Stony Brooke University, USA. The nanoarchitecture density is even becoming bigger by stacking the nanowire array on the top of each other realizing a three-dimensional (3D) nanoarchitecture [52, 53]. Three applications have been proposed using CMOL architecture: field programmable logic arrays, memories and neuromorphic processors. Of these three applications, memories are the most straightforward application of CMOL architecture where memristor-based Resistive Random Access Memory (RRAM) is one of the proposed memory technologies.

In this chapter, a top-down hierarchical modeling that describes a 3D memristor-based RRAM is presented. A top-down hierarchical modeling is an approach commonly utilized to evaluate complex electronic systems in a simple manner and short time duration. Section 2.1 gives an overview of a top-down hierarchical modeling approach divided into five levels. Section 2.2 presents the highest level model referred to as behavioral memory model, which treats the memory as a single black box. Section 2.3 describes the functional memory model, which provides the internal blocks that form the behavioral model. Section 2.4 explains the electrical memory model, which illustrates the electrical
components and their connections. Section 2.5 discusses the lowest level model referred to as layout memory model, which exhibits the physical composition of the memories. Finally, Section 2.6 summarizes this chapter.

2.1 Memory models

As mentioned before, a 3D RRAM consists of different device technologies fabricated at different layers. Such a complex circuit is difficult to evaluate without translating it into a model. Memory modeling eases the description and evaluation of memory systems using simulation tools. This approach is carried out by dividing the models into several abstraction levels as shown in Figure 2.1. Depending on the area of interest, researchers can focus only on one level while concealing the irrelevant information. The behavioral model is the highest abstraction level and the layout model is the lowest abstraction level. As the abstraction levels shift from the highest toward the lowest, the model provides lesser information about the memory functionality and more about the memory physical structure. It is possible to have a model that consists of different abstraction levels. A brief explanation of each modeling level is explained as follows.

**Behavioral model.** This model provides only the information concerning the way the memory behaves. At this level, the model illustrates a memory as a *black box* with external input and output signals. There is no single information about the internal structure of the memory. A timing diagram is typically used to describe the write and read operations of the memory.

![Figure 2.1: Abstraction levels of memory modeling](image)
2.2 Behavioral RRAM model

Functional model. This model provides the information of the internal organization of the memory. At this level, numerous subsystems referred to as functional blocks interact with each other to accomplish a specific function.

Logical model. This model provides the information of the logic gates that form the functional blocks of the memory. At this level, simple boolean expressions are used to describe the logic gates.

Electrical model. This model provides the information of the electrical components that build up the memory. At this level, the functional blocks and logic gates of the memory are described using electrical components such as resistors, transistor and capacitors.

Layout model. This model provides the information of the actual physical design of the memory. At this level, all relevant information such as the geometrical configurations are taken into consideration.

2.2 Behavioral RRAM model

This section explains in detail the behavioral model of RRAM in terms of its block diagram and timing diagram.

2.2.1 RRAM block diagram

Behavioral model is the highest abstraction level that illustrates a system as a black box with external input and output signals. The behavioral model for RRAM consists of a memory block that receives four input signal and produces one output signal, as shown in Figure 2.2. The input signals comprise Clock, A-bit Address, B-bit Data-in and C-bit Control bus where $A$, $B$ and $C$ are integers; the output signal consists of $B$-bit Data-out bus. Note that bus refers to several lines that carry the same type of signal.

![Figure 2.2: Behavioral model of RRAM](image-url)
Chapter 2. RRAM Architecture

The Clock signal synchronizes all functional blocks inside the RRAM. The Address bus carries the address of the selected memory cell to be accessed. The Data-in bus carries a group of data to be written into the memory. The Control bus carries the Write/read signal that determines the desired memory operation (whether read or write operation) and the Chipselect signal that selects the desired memory part to operate. The Data-out bus carries a group of data retrieved from memory cells to be sent out from the memory. Typically, the Data-in and Data-out are combined to form a bidirectional data lines.

2.2.2 RRAM timing diagram

The interaction of signals during a memory operation is usually described using a timing diagram. A timing diagram is a waveform representation of the input and output signals of a memory in time domain.

A typical timing diagram for the write operation of a memory is illustrated in Figure 2.3(a). The write operation begins by setting up the selected memory cell address before Clock signal goes high and the Chipselect signal goes low. At the same time, the Write/read signal is set to high to acknowledge the memory that write operation is being performed. One clock later during Chipselect signal goes low, the data that exists on the input data lines will be written into the selected cell. Note that the shaded part in the signals are don’t care logic values, which can be the desired values or else.

A typical timing diagram of the read operation for a memory is illustrated in Figure 2.3(b). After the address activation, the Write/read signal is set to low to acknowledge the memory that read operation is being performed. One clock later during Chipselect signal goes low, the data that exists on the output data line is sent out of the memory.

![Timing diagram for (a) write operation and (b) read operation](image)

Figure 2.3: Timing diagram for (a) write operation and (b) read operation
2.3 Functional RRAM model

This section describes the functional model of 3D RRAM where its non-CMOS memory cell array is structured on the top of CMOS peripheral circuits, while connecting the two layers using vertical vias; see Figure 2.4. A brief explanation of the functional blocks inside RRAM is discussed as follows.

**Memory cell array.** This functional block is the heart of every memory chip in which data is stored. The memory cell array of RRAM is organized as an array of nanowire word line NWL rows and nanowire bit line NBL columns. At each NWL×NBL crosspoint, a two-terminal non-CMOS device (such as memristor, single-electron junction and organic molecules) is embedded representing a single bit; thus, the memory has a total of \( n = NWL \times NBL \) bits [46, 47].

**CMOS to Non-CMOS Vias.** This functional block provides the interface to transfer data between the memory cell array and the peripheral circuits. These vertical CMOS to Non-CMOS Vias (CNVs) are analogous to the Through Silicon Vias (TSVs) implemented in existing three-dimensional

![Figure 2.4: Functional model of RRAM](image-url)
Peripheral circuits. This functional block allows data transfer between
the memory cell array and external parts, for instance, the microprocessor. The
peripheral circuits of RRAM consist of six main sub-blocks including row and
column address decoders, control circuits, write/read circuits, sense amplifiers,
row and column multiplexers, and input and output data buffers [55]. Both
row and column decoders operate together to access the selected memory cell in
the cell array. The write/read circuits supply the appropriate voltages for the
write (depending on the input data value in data buffer) and read operations.
The sense amplifiers sense read current, convert it into a voltage and amplify
it prior to sending the output data to the data buffers. The row multiplexers
connect NWLs in the memory cell array to the write/read circuits during write
1 and read operations. The column multiplexers connect NBLs in the memory
cell array to the write circuits during write 0 operations; they also connect the
cell array to the sense amplifiers during read operations. All these operations
are controlled by the control circuit.

2.4 Electrical RRAM model

This section provides the electrical model for RRAM. Because some of the func-
tional blocks of the peripheral circuits are complex and they are not easily
represented by their electrical model, only the blocks used in the simulation
and analysis of the memory behavior are considered.

2.4.1 Memory cell

The design and configuration of memory cells are different depending the mem-
ory technologies used. In this thesis, memristors are considered as the RRAM
technology. A memristor is a two-terminal non-CMOS device whose resistance
changes in response to the applied voltage or current [58]. Interestingly, this
extremely thin non-CMOS device “memorizes” the last resistance when the sup-
ply voltage is turned off; thus, it is suitable to use memristors as non-volatile
memory cells.

Figure 2.5 depicts the electrical schematic of a single two-terminal RRAM
cell \( C \). One terminal is connection to a nanowire word line \( NWL \) and another
terminal to a nanowire bit line \( NBL \). \( NWL \) is connected to a CMOS bit line
\( CBL_R \) through short CNV \( SV \) and access transistor \( AT_R \) that is controlled by
CMOS word line \( CWL_R \). Conversely, \( NBL \) is connected to CMOS bit line \( CBL_C \)
through tall CNV \( TV \) and access transistor \( AT_C \) that is controlled CMOS word
line \( CWL_C \). Note that \( AT_R, CWL_R \) and \( CBL_R \) are part of the row multiplexer,
2.4. Electrical RRAM model

Figure 2.5: Electrical circuit of a single RRAM cell connected to its pass transistors

while $AT_C$, $CWL_C$ and $CBL_C$ are part of the column multiplexer.

**Write and read operations.** The selected cell is written and read by biasing sufficient voltages across the cell. Figure 2.6(a) shows a 2×2 cell array with the access transistors where a write 1 operation is applied to cell $C_{11}$; while Figure 2.6(b) and Figure 2.6(c) depict the biasing voltage required for write 0 and read operations, respectively. Note that the gray-colored cell is the selected cell, while the white-colored cells are the unselected cells; see the notation at the bottom-left part of Figure 2.6(a). The timing diagram for the operations is given in Figure 2.6(d).

For write 1 operation, the *Write/read enable* signal is set to high to initiate the write operation. During the write operation, the nanowire word line $NW_{sel}$ is set to $V_{dd}$ (by setting $CBL_{R1}$ to $V_{dd}$ and activating the access transistor $AT_{R1}$) and the nanowire bit line $NBL_{sel}$ to GND (by setting $CBL_{C1}$ to GND and activating the access transistor $AT_{C1}$) [59]. With this biasing condition, the write voltage is equal or larger than the threshold voltage of $C_{11}$, i.e., $V_{dd} \geq V_{th}$ enabling the write operation. At the same time, the unselected nanowire word line $NW_{unsel}$ and nanowire bit line $NBL_{unsel}$ are biased each with $V_{dd}/2$, preventing any voltage drop across unselected cells.

Write 0 operation is performed by biasing the selected cell with $V_{dd}$ from $NBL_{sel}$, while grounding $NW_{sel}$. At the same time, $NW_{unsel}$ and $NBL_{unsel}$ are biased with $V_{dd}/2$; see Figure 2.6(b).

For a read operation, the *Write/read enable* is set to low to initiate the read operation. During the read operation, the selected cell is biased with $-V_{dd}$ in the first half and $+V_{dd}$ in the second half from its $NW_{sel}$. At the same time, $NBL_{sel}$ is connected to the sense amplifier, and both $NW_{unsel}$ and $NBL_{unsel}$ are left floating. During the second half of the read operation, the *Sense amplifier enable* signal is activated to sense the read current, convert it to voltage, amplify it and send it to output data buffer.
Figure 2.6: Signal voltages for (a) write 1 operation (b) write 0 operation (c) read operation. (d) Timing diagram of write and read operations.
2.4. Peripheral circuits

Peripheral circuits such as address decoders, write/read circuits, sense amplifiers, row and column multiplexers, and data buffers are based on CMOS transistors. These circuits are briefly discussed as follows.

**Address decoders.** The address decoder is divided into row and column decoders in order to reduce its size and the length of word lines and bit lines. The task of the row decoder is to select a desired CMOS word line $CWL_R$ from a set of word lines in the memory. Note that the selected $CWL_R$ controls the access transistor $AT_R$ as mentioned in the previous section. Figure 2.7(a) shows a portion of a static decoder formed by a PMOS transistor $M_1$, a number of NMOS transistors $M_2$ to $M_{k+1}$, and an inverter formed by $Q_1$ and $Q_2$ [60]. The NMOS transistors are connected to address bits $A_0$ to $A_{k-1}$ where $k$ is an integer, while the PMOS is grounded. When a desired address bits (11...1) are asserted to the NMOS transistors, the inverter is pulled down activating $Q_1$; this in turn sets the selected word line to high. The undesired address bits cause the NMOS transistors to turn off; in this case the inverter is pulled up activating $Q_2$ and in turn sets the word line low. A similar circuit is used for the column decoder to select the desired CMOS word line $CWL_C$. Note that these line controls the access transistor $AT_C$ mentioned in the previous section.

Figure 2.7(b) shows a portion of a dynamic decoder that consists of the same components used in the static decoder, but with one extra NMOS transistor $M_{k+2}$ and a clock signal. The PMOS transistor $M_1$ and the NMOS transistor

![Figure 2.7: Schematic of (a) static decoder (b) dynamic decoder](image-url)
Chapter 2. RRAM Architecture

Write/read circuits. The write/read circuits supply the appropriate voltages to the memory cell array during write and read operations. As shown in Figure 2.8(a), the write circuit comprises a pair of NMOS and PMOS transistors \( M_1 \) and \( M_2 \) with input data \( \text{Din} \) signal, and an access transistor \( M_3 \) that is controlled by \( \text{Write/read} \) control signal [60]. When the write operation is activated (by setting \( \text{Write/read} \) to high), the \( \text{Din} \) value determines whether to connect \( \text{CBL}_R \) to \( +V_{dd} \) or \( \text{GND} \). If the \( \text{Din} \) signal is 1, then \( M_1 \) is activated connecting \( \text{CBL}_R \) to \( +V_{dd} \); otherwise if the \( \text{Din} \) signal is 0, \( M_2 \) is activated connecting \( \text{CBL}_R \) to \( \text{GND} \). These appropriate voltages will be supplied to the selected memory cells.

The read circuit is quite similar to the write circuit except the signal that controls \( M_1 \) and \( M_2 \) as well as the type of \( M_3 \); see Figure 2.8(b). In this circuit, \( M_1 \) and \( M_2 \) are controlled by the \( \text{Read} \) signal, and \( M_3 \) is an NMOS transistor. When read operation is activated (by setting \( \text{Write/read} \) to low), the \( \text{Read} \) signal will determine whether to connect \( \text{CBL}_R \) to \( +V_{dd} \) or \( -V_{dd} \). If the \( \text{Read} \) signal is 1, then \( M_1 \) is activated connecting \( \text{CBL}_R \) to \( +V_{dd} \); otherwise if the \( \text{Read} \) signal is 0, \( M_2 \) is activated connecting \( \text{CBL}_R \) to \( -V_{dd} \). During the read operation, this read circuit works together with the sense amplifier.

Sense amplifiers. The sense amplifiers receive (“sense”) either voltages or currents from the selected memory cells. The sensed voltage (or current), which corresponds to the stored logic value in the memory cell, is then amplified before being sent to the output data buffers. According to [61, 62, 63], a hybrid current-voltage mode sense amplifier (hybrid SA) that senses a read current and produces an output voltage is suitable for high-performance memory. Figure 2.9 illustrates the electrical circuit of a hybrid SA [63]. It consists of the cross-coupled latch of transistors \( M_1, M_2, M_3 \) and \( M_4 \); the clamp transistors \( M_5 \) and \( M_6 \); and the activation transistors \( M_7 \) and \( M_8 \).
The hybrid SA begins to operate when the Sense amplifier enable signal is set to high activating $M_8$ and providing power to the circuit. When the Clock signal is set to high, transistors $M_7$ is turned on forcing the output nodes to have equal potentials. In case of $I_{\text{read}}$ is greater than $I_{\text{ref}}$, then the current flows through $M_5$ will be larger than that of $M_6$. The differential current then appears at the output nodes. When the Clock signal is low, the differential current flows through $M_3$ and $M_4$ and charging the small equivalent capacitances at the drains of the transistors, respectively. A small differential voltage will then appear across the transistors’ drain and be amplified to CMOS-signal level $V_{\text{out}}$ and $\overline{V}_{\text{out}}$ by the cross-coupled latch and inverters.

**Row and column multiplexers.** The row and column multiplexers ensure that the selected signals from peripheral circuits are connected to the selected nanowire word lines and bit lines in the memory cell array (through their corresponding vias); see Figure 2.4. These multiplexers consist of either transmission gates as shown in Figure 2.10(a) or typical access transistors. Transmission gates are constructed by connecting the sources and drains of both NMOS and PMOS transistors together. Their gates are connected to $\text{CWL}_R$ and its complement in case of row multiplexer, or $\text{CWL}_C$ and its complement in case of column multiplexer. When compared to typical access transistors, transmission gates have the ability to pass through a full voltage high and voltage low at larger area overhead [22].

**Input and output data buffers.** The input data buffer holds the input data for the write operation, whereas the output data buffer holds the read data before being sent out to external circuits such as microprocessor. Figure 2.10(b) shows the simplified circuit of a combined input and output data buffer. It consists of the same circuit as the cross-coupled latch of the sense amplifier.
Figure 2.10: Schematic of (a) row and column multiplexer (b) input and output data buffer

(i.e., $M_1$ to $M_4$), along with two latch activation transistors $M_5$ and $M_6$, and an access activation transmission gate. When the Latch signal is activated, the data become available to the cross-coupled latch. The cross-coupled latch holds the written data as long as it is supplied with power (similar to SRAM cells). The input or output data is connected to the latch through the transmission gate, which is controlled by the Access signal.

2.5 Layout RRAM model

Figure 2.11(a) shows the generic layout model of an RRAM based on CMOL architecture [46, 47, 54]. The memory consists of three main parts, each of which is fabricated at different layers: these are non-CMOS memory cell array, CMOS to Non-CMOS Vias and CMOS peripheral circuit. The memory cell array formed by two sets of nanowires crossing in perpendicular creating the word lines $NWLS$ and bit lines $NBLs$. At each nanowires crosspoint, a bistable two-terminal nanoelectronic device such as memristor and organic molecules is fabricated. The middle layer is the CNVs made of metal such as copper and tungsten. The bottom layer consists of the peripheral circuits structured from CMOS devices.

Figure 2.11(b) illustrates the connection of a single memristor-based RRAM cell; see also Figure 2.5 for its electrical schematic circuit. A $D \leq 10nm$ memristor made of a transition-metal oxide film (e.g., titanium-oxide ($TiO_2$), nickel-oxide ($NiO$), etc.) is embedded between two nanowires. The memristor consists of two layers: (i) the doped layer (with thickness $w$) that has a slight depletion of negatively charged oxygen atoms, and (ii) the undoped layer (with thickness $D - w$) that has its original doping properties. The depleted negatively charged oxygen atoms in the doped layer create a positively charged oxygen vacancy; this vacancy acts as the donor of electrons that carry a current [12]. The bound-
2.6. Summary

This chapter described the architecture of memristor-based RRAM, which is the focus in this thesis. The main topics discussed are as follows.

- Description of the chosen architecture that build up the target RRAM. The architecture is based on a 3D nanoarchitecture that promises huge data density.
- Identification of memory modeling based on a top-down hierarchical approach. This modeling approach divides memory models into five levels starting with the behavioral model on top, thereafter the functional model, the logical model, the electrical model and finally the layout model at the bottom.
- Introduction of the behavioral memory model of memristor-based RRAM by treating it as a block box and its external signal pins. A timing diagram is also given to describe the operations of such memories.
- Discussion of the functional memory model of memristor-based RRAM where several functional blocks that form the memory are explained.
These include memory cell array, address decoders, control circuits, write/read circuits, input/output data buffers, and row and column multiplexers.

- Explanation of the electrical memory model of memristor-based RRAM, where the electrical composition of the functional blocks is explained.

- Description of the layout model of memristor-based RRAM including a brief introduction to memristor.
CHAPTER 3

TESTING AND FAULT TOLERANCE CONCEPTS

3.1 Key terminologies
3.2 Testing and fault tolerance in product manufacturing flow
3.3 Manufacturing testing
3.4 Design-for-Testability
3.5 Fault-tolerant schemes
3.6 Summary

As mentioned in the first chapter, massive defect densities and fault rates may overwhelm the advantages offered by emerging memories. The former problem is caused by process variations during manufacture resulting in quality problem. The latter problem is due to in-field disturbances during usage resulting in reliability problem. Even if they are manufactured with high quality, emerging memories are vulnerable to in-field disturbances as such novel circuits are fabricated using tiny devices in a very dense structure. Therefore, both quality and reliability are important characteristics in producing good emerging memories. High quality and reliability emerging memories require efficient testing; in addition, high reliability emerging memories require fault tolerance to prolong their usage lifetime.

In this chapter, the concept of memory testing and fault tolerance are presented. The aim is to provide readers a basic understanding of several schemes employed to improve the quality and reliability of emerging memories. Section 3.1 defines several key terminologies associated with testing and fault tolerance. Section 3.2 discusses the importance of testing and fault tolerance during design steps in order to achieve a specified quality and reliability requirement. Section 3.3 classifies and briefly defines several types of memory tests. Section 3.4 defines a test approach referred to as Design-for-Testability (DfT) and discusses several existing DfT schemes. Section 3.5 classifies and briefly defines several fault-tolerant schemes. Finally, Section 3.6 summarizes this chapter.
3.1 Key terminologies

This section gives the definition of several key terminologies associated with testing and fault tolerance as shown in Figure 3.1 [64, 65]. Assuming that each flaw is sufficient to produce its effect, a defect leads to faults, a fault leads to errors and an error leads to failures. Testing is carried out to capture defects and to ensure that the manufactured products have high quality. Fault tolerance is implemented to tolerate faults and errors, which in turn to ensure that the sold products have high reliability. A brief description of these key terminologies is given as follows.

**Defect.** A defect is the physical anomalous emerged from the manufacturing process that was not originally defined in the design circuit. Physical anomalous due to unintended particles, such as an extra or missing material caused by contamination, are referred to as hard defects. Besides hard defects, latent defects due to minor process variations are becoming crucial in nanotechnology. Defects due to source/drain dislocation, insufficient doping and via pinholes are examples of latent defects.

**Fault.** A fault is the manifestation of defects and external disturbances. External disturbances are due to environmental sources such as cosmic rays, alpha particles and temperature fluctuations. The effects of such defects and external disturbances can result in three types of faults including hard faults, intermittent faults and transient faults. Hard faults occur persistently and are caused by hard defects. Intermittent faults occur repetitively (appears, disappears and reappears repeatedly) and are caused by latent defects or aging components. Transient faults occur randomly (appears and disappears shortly) and are caused by external disturbances.

**Error.** An error is the fault effect at the output of a circuit. The effect is in the form of logical state, i.e., the affected logical state differs from the correct state. Hard errors are caused by hard faults and result in a determinate logic state. For example, extra material that bridges a line to power supply causes

![Figure 3.1: Key terminologies associated with testing and fault tolerance](image-url)
a stuck-at-1 fault, which in turn produces logic 1 throughout the operation of
the circuit (if no correction action is taken). Soft errors are caused by either
intermittent or transient faults and result in an indeterminate logic state. For
example, external disturbances cause coupling faults between adjacent lines,
which in turn produce either logic 1 and logic 0.

Failure. A failure is the inability of a circuit to perform its intended
function due to errors. Depending on the impact of errors and the capability
of the system to handle them, the failure can be manifested as either an incorrect
result, no result at all or a violation of design parameters (e.g., a correct result
that delays from the intended time).

Quality. Quality is the metric used to quantify the defective devices
that escape manufacturing tests but are returned back to the manufacturer
by customers. A low number of customer returns corresponds to high quality.
However, hard and latent defects induced during manufacturing processes
affect the quality.

Reliability. Reliability is the metric used to quantify the ability of
a system to operate correctly for a specified period of time under certain
conditions. The longer a system operates correctly corresponds to higher
reliability. However, faults due to latent defects and external disturbances
affect the reliability.

Testing. Testing is a process performed to ensure the correctness
of a product. This is achieved by distinguishing a good, bad and weak
product. Tests with high fault coverage capability are able to detect
defects that cause bad and weak products, thus improving quality and reliability.
However, tests with low fault coverage influence these two measurements.

Fault tolerance. Fault tolerance is the ability of a system to perform its
intended functions in the presence of faults. This is achieved by incorporating
fault-tolerant design to the original circuits. A fault-tolerant circuit is able to
mitigate faults during the operation, hence improving reliability.

3.2 Testing and fault tolerance in product manufacturing flow

A product manufacturing flow describes the process stages and activities per-
formed to produce products such as memories. Figure 3.2 shows a simplified
product manufacturing flow with four process stages: design, fabrication, pack-
aging and testing [66]. A brief description of these processes is as follows.
Design. This process translates the idea into memory model that satisfies the intended functions and specifications. For memories that require high reliability, fault-tolerant schemes can be implemented during the modeling phase. The designed memory model is then optimized and validated using simulation tools to ensure that the intended functions and specifications are fulfilled.

Fabrication. This process realizes the designed memory into a physical circuit on a piece of wafer. For example, in CMOS-based memory technologies, the fabrication subprocesses include pure silicon crystal growth, wafer slicing, thinning, impurity deposition, patterning, etching, etc. These subprocesses define the structure of memory components such as CMOS devices (gate, source, drain and channel) and interconnects on a piece of wafer.

Packaging. This process assembles the fabricated circuits (dies) into a finished product presentation, i.e., chip. Packaging process begins with cutting (dicing) the fabricated wafer into dies, each of which is then mounted onto the lead frame structure, bonded it to the package lead frames, encapsulated it into a plastic mold and sealed.
Testing. This process validates the functionality correctness of the fabricated and packaged memories. Each fabricated and packaged memory is tested using automatic test equipments, which apply measurement signals (i.e., test stimuli in a certain memory operation sequence) and perform test evaluation (i.e., fault detection and diagnosis). Stresses in the form of elevated temperature or voltage may also use during the testing to weed out weak memories, which are probable to fail during in-field operation.

The process stage considered in this thesis is testing and the process activity is the fault-tolerant scheme implementation. Testing is a method to achieve high-quality and reliable memories. As shown in Figure 3.3, there are two main steps involved in testing: (i) test generation that applies test stimuli to memory under test and (ii) test verification that validates the correctness of the test signatures from the memory under test. The complexity of existing and emerging memories demand a special feature to facilitate testing [67]. The process of incorporating a special circuit inside the memory is referred to as Design-for-Testability (DfT) [23]. DfT schemes ensure that the manufacturing testing achieves high test coverage at low test cost.

The fault-tolerant schemes are implemented during the design stage in order to achieve reliable memories. Due to the improvement in CMOS technology, fault-tolerant schemes were only implemented in applications that require high reliability, crucial computation and long operational life. However, because the

![Figure 3.3: Concept of testing and Design-for-Testability](image)

![Figure 3.4: Concept of fault-tolerant schemes](image)
CMOS fabrication process is becoming more imprecise and the nanotechnology fabrication process is still immature, the interest for fault tolerance is returning. The fundamental concept of fault tolerance is redundancy, which is the extra circuits or elements added to the original circuit as shown in Figure 3.4.

3.3 Manufacturing testing

Depending on the objectives, manufacturing testing can be classified into two types such as given in Figure 3.5 [64, 67]. Quality testing is a set of tests applied to distinguish bad memories from good ones. Tests that belong to this class apply appropriate electrical test stimuli to verify the correctness of the memories at the time of testing. Reliability testing, on the other hand, is a set of tests applied to distinguish weak memories from good ones. Tests that belong to this class apply test stimuli together with test stress such as high temperature, high voltage and high clock speed to detect the potential future defects, as well as to verify the present memory correctness. The following sections briefly review some of these tests.

3.3.1 Quality testing

The three tests that belong to quality testing including functional tests, structural tests and defect-oriented tests are described as follows [64, 67].

**Functional tests.** Functional tests aim to detect defects that cause a memory to function incorrectly, by treating the memory as a black box. Based on functional specifications given by the designers, 2^n test
stimuli in the form of digital values (logic 0 or 1) are applied exhaustively, where \( n \) is the number of input lines. Because memories are becoming more complex and denser, this testing type becomes more costly and time consuming.

**Structural tests.** Structural tests aim to detect defects that alter the *structural composition* of a memory with respect to the correct functional specification. Based on the structured fault models that imitate defects, the test stimuli in the form of digital values are applied. Because of economic considerations, it is often not practical to test for all possible defects in the memory. A trade-off between the economic and the type of fault model is usually taken into consideration in practice.

**Defect-oriented tests.** Defect-oriented tests aim to detect defects that affect *any measurable memory behavior* such as digital values, timing, current consumption, etc. Based on the realistic defect models, the appropriate test stimuli are applied and measured. The defects may be modeled as a spot of extra metal or missing metal in the layout. A trade-off between the economic and quality is usually taken into consideration in practice.

### 3.3.2 Reliability testing

The three tests that belong to reliability testing including *parametric tests*, *dynamic tests* and *burn-in tests* are described as follows [64, 67].

**Parametric tests.** Parametric tests aim to detect defects that alter the *external behavior* of a circuit, i.e., analog parameters measured at the input and output pins of a circuit. Dc parametric tests concern with time-independent analog parameters such as voltages and currents under steady-state condition. In contrast, ac parametric tests concern with time-dependent analog parameters such as rise time, fall time, setup time, hold time and delay of a signal.

**Dynamic tests.** Dynamic tests aim to detect defects associated with *internal behavior* and *time-dependent behavior* of a circuit. These tests are performed at various clock speeds depending on the target faults. Dynamic tests applied at high clock speed can verify slow components in a memory circuit, whereas dynamic tests applied at slow clock speed can verify leakage current from memory cell.

**Burn-in tests.** Burn-in tests aim to detect defects that cause weak devices. Test stress such as elevated-temperature or elevated-voltage is applied to accelerate the infant mortality period of the weak devices. Together with quality tests, burn-in can screen out the weak devices that might induce failures during operation.
3.4 Design-for-Testability

Design-for-Testability (DfT) aims to reduce the difficulty and cost of testing [23]. The main testing difficulty arises from the increment in circuit complexity that decreases the controllability and observability of a circuit. The original circuit design is modified to accommodate a DfT circuit, which can be activated for the test mode and deactivated for the normal operation mode. When the test mode is activated, the external test stimuli can be controlled and observed at the selected deep nodes through this DfT circuit.

Figure 3.6 depicts three DfT schemes proposed for testing SRAM stability faults memories such as weak write test mode [68], force voltage [69] and word line pulsing [70]. These DfT schemes are briefly described as follows.

**Weak write test mode.** This DfT scheme consists of weak write circuits to supply weak voltages to SRAM cells during the test [68]. The weak write circuits are connected to every column of an SRAM array and will try to weakly overwrite SRAM cells with the complementary value. When it is activated, a sequence of write solid $D$, weak write $\overline{D}$ and read $D$ is applied where $D$ can be logic 1 or 0. The unstable cells written with a solid $D$ will be flipped to $\overline{D}$ when it is overwritten by the weak write, while the stable cells remains storing $D$. When reading, the unstable cells that have been overwritten to the complementary logic value will be detected.

**Force voltage.** This DfT scheme consists of write circuits to supply intermediate voltages to the SRAM cells before the memorizing phase [69]. The intermediate voltages will set the unstable SRAM cells to an intermediate quiescent state and the stable cells to solid quiescent state. During the memorizing phase, the unstable cells will consume higher quiescent current $I_{ddq}$ than the stable cells. When reading, the unstable cells will be detected by measuring the $I_{ddq}$ value.
Word line pulsing. This DfT scheme consists of a reference cell at each SRAM array column [70]. When the DfT circuit is activated, the cell under test (CUT) is set to $D$, whereas the reference cell is set with $\overline{D}$. Thereafter, numerous pulses are applied to the reference cell word line, followed by activating the CUT word line and finally the read operation. A repetition of the word lines activations and read operations enables the detection of the unstable cells.

3.5 Fault-tolerant schemes

Fault-tolerant schemes are primarily based on redundancy, as well as special design approach as shown in Figure 3.7. Redundancy can be classified into three classes: hardware, time and information [71, 72, 73]. Hardware redundancy replicates the original hardware of a circuit. This scheme can tolerate permanent, intermittent and transient faults. Time redundancy repeats an intended function several times. This scheme can tolerate intermittent and transient faults, but not permanent faults. Information redundancy appends additional data to the original one. This scheme can tolerate permanent, intermittent and transient faults.

Special design schemes that modify design architecture or use dedicated components have been implemented in fault-tolerant systems [24, 74, 75, 76, 77]. The former scheme can tolerate permanent, intermittent and transient faults, while the latter scheme can tolerate intermittent and transient faults.

![Figure 3.7: Classification of fault-tolerant schemes](image-url)
3.5.1 Redundancy

The three design schemes that belong to redundancy including double modular redundancy, recomputing and error correction codes are described as follows.

**Double modular redundancy.** Double modular redundancy (DMR) is a hardware redundancy scheme where two identical modules operate in parallel and their corresponding results are compared, as depicted in Figure 3.8. Any disagreement from the comparison indicates that faults have occurred in one of the modules. DMR is a variant of N-modular redundancy (NMR) where N is usually an odd integer [71]. NMR has majority voter that will mask errors due to faults in some of its modules. This scheme can tolerate permanent, intermittent and transient faults. The advantages of DMR are high performance operation due to its parallel operation and simple hardware implementation. On the other hand, the primary drawback is that the scheme incurs high area overhead.

![Double modular redundancy scheme](Figure 3.8: Double modular redundancy scheme)

![Recomputing scheme](Figure 3.9: Recomputing scheme)
Recomputing. Recomputing is a time redundancy scheme where the same data is computed repeatedly and their corresponding results are compared, as shown in Figure 3.9. If the results show any discrepancy, then the computation can be executed again until the desired result is obtained. This fundamental time redundancy scheme can also be combined with other design approaches resulting in schemes such as recomputing with shift operand (RESO) and recomputing with swapped operand (RESWO) [72]. The fundamental recomputing scheme can tolerate intermittent and transient faults, but not permanent faults. On the other hand, RESO and RESWO can tolerate all three fault types. The advantages of recomputation are its small area overhead and simple hardware implementation. On the contrary, performance latency is the shortcoming of this scheme.

Error correction codes. Error correction codes (ECCs) are the information redundancy scheme where the original data is encoded into new representation [73]. The encoding process can be simply a replication of the original data or it can be in the form of a codeword. A group of codewords constructs an ECC. A codeword is a combination of a dataword and a checkword as illustrated in Figure 3.10. The dataword represents the original user data (input data), while the checkword consists of additional data that provides the detection and correction capability. In order to recover the original data, the codeword must be decoded. Decoding is essentially the reverse process of encoding to ensure that the input data is recovered at the output side (referred to as output data);

![Diagram of Error Detection and Correction Code Structure](image)

Figure 3.10: Error detection and correction code structure
Chapter 3. Testing and Fault Tolerance Concepts

It also detects and corrects errors in the codeword if any. ECCs can tolerate hard errors due to hard faults, and transient errors due to intermittent or transient faults. Depending on the distribution of the errors, either random errors or clustered (burst) errors, this scheme can be classified into two groups: bit-based ECCs and symbol-based ECCs. Bit-based ECCs such as Hamming, Parity, Hsiao, Duplication, Bose-Chaudhuri-Hocquenghem, Euclidean Geometry Low-Density Parity-Check codes are suitable to detect and correct random errors. In contrast, symbol-based ECCs such as Reed Solomon and Redundant Residue Number System (RRNS) codes are suitable for clustered errors. Because RRNS code is adapted in this research, the ECC will be explained in more detail in Chapter 6.

3.5.2 Special design

The two design schemes that belong to special design including interleaving and Muller C-gate are described as follows.

**Interleaving.** Interleaving is a design scheme that arrange data into a non-continuous arrangement [64]. In order to implement this scheme, an interleaver is used at the input side and de-interleaver at the output side, as depicted in Figure 3.11. The interleaver spreads adjacent data so that the it is carried by non-adjacent lines, while de-interleaver rearranges the data into the original organization. This design scheme effectively tolerate faults that impact a group of adjacent data.

**Muller C-gate.** Muller C-gate is a logic gate commonly employed in asynchronous applications [78]. Figure 3.12(a) and (b) shows its symbol and its truth table, respectively. This logic gate only changes its output, to be the same as the input logic value, when the states of all inputs match. On the other hand, the output remains in its previous state if the inputs are dissimilar. This rendezvous property is suitable to mask short period glitches due to intermittent and transient faults.

![Figure 3.11: Interleaving implementation](image-url)
3.6 Summary

This chapter provided the concept of memory testing and fault tolerance in general. The main topics discussed are as follows.

- Definition of several key terminologies associated with memory testing and fault tolerance.
- Discussion of testing and fault tolerance as the important step in manufacturing memories. These two steps enable the designed and manufactured memories to have high quality and reliability attributes.
- Classification of several memory tests into two groups based on the objectives either for quality or reliability. A brief description of each memory tests is also given in terms of the aims and the applied stimuli type. Defect-oriented test is adopted in improving the quality of emerging memories studied in this thesis.
- Identification of several Design-for-Testability schemes proposed for existing SRAM. These approaches are used to address stability faults in SRAM and are adopted in facilitating the defect-oriented test studied in this thesis.
- Classification of several fault-tolerant schemes into two groups based on how they are implemented, either redundancy or special design approaches. Although there are many more fault-tolerant schemes proposed so far, only the ones that are adopted in this thesis are described.
Part II:
Fault Modeling and Test for RRAM

This part presents the modeling of faults and development of test schemes for the target emerging memory, i.e., RRAM. Chapter 4 introduces a defect-oriented test framework, RRAM electrical model for defect simulation and fault models. Two unique fault models due to resistive open defects have been established; such faults demand a new test approach as they might escape from manufacturing tests. Chapter 5 proposes Design-for-Testability (DfT) schemes to detect the unique faults in RRAM. In addition to detecting the unique faults in RRAM, the proposed DfT schemes are programmable; this capability enables the proposed DfT scheme to track the unexpected changes (such as process variations) and prevent overkill.

Chapter 4: Defect-Oriented Test for RRAM
Chapter 5: Design-for-Testability Schemes for RRAM
CHAPTER 4

DEFECT-ORIENTED TEST FOR RRAM

4.1 RRAM defects classification and definition
4.2 Simulation model
4.3 Defect injection and circuit simulation
4.4 Summary

As aforementioned in Chapter 1, Resistive Random Access Memory (RRAM) is one of the primary candidates for use as universal memory in future computer systems. Universal memory is the one that possesses the combined attributes of the non-volatility of flash, density of DRAM and performance of SRAM. From the density perspective, RRAM outperforms the other emerging memory technologies due to its simple two-terminal cell element (without access transistor). Because of these potentials, the research on such a memory technology is growing. To this end, most of the work published so far focuses mainly on the modeling and design [55, 59, 79], fabrication techniques [11, 12, 13, 14, 80] and reliability improvement using fault tolerance schemes [35, 37, 39, 54, 46, 81, 82, 83]. However, research and published literatures on defect analysis, testing and Design-for-Testability for such devices are very limited [29, 30]. Understanding the faulty behavior of the memory devices in the presence of defects will enable the development of appropriate fault models and efficient test schemes; thus, improve the outgoing product quality.

In this chapter, a framework of defect-oriented test for RRAM based on electrical simulation is presented. As open defects occur most frequently in deep sub-micron circuits and it is expected to be the same for emerging memories, this defect type is the focus of the study. The resulting faulty RRAM cell behaviors from the simulation are translated into fault models to develop efficient memory tests. Section 4.1 discusses the classification and definition of possible defects that may occur in an RRAM. Section 4.2 introduces the electrical RRAM model used for defect injection and simulation. Section 4.3 provides the defect injection and simulation for open defects impacting a single RRAM cell including the defect-free and open defect simulation, as well as their analysis. Finally, Section 4.4 summarizes this chapter.
4.1 RRAM defects classification and definition

Defects in memory circuits are the physical structures that deviate from the intended layout design and caused by imperfection in the fabrication process. They introduce unintended disconnections or connections in the memory. Defects such as broken or missing metal lines and extra metal lines can be modeled at the electrical level using a resistor as follows [85]:

- **Open**: This is an unintended series resistance $R_{op}$ within a connection. The open resistance can be in the range of $0 \, \Omega < R_{op} \leq \infty \, \Omega$.

- **Bridge**: This is an unintended parallel resistance $R_{br}$ between two connections. The bridge resistance can be in the range of $0 \, \Omega < R_{br} \leq \infty \, \Omega$.

- **Short**: This is an unintended resistive path $R_{sh}$ between a node and supplied voltage $V_{DD}$, or ground $GND$. The short resistance can be in the range of $0 \, \Omega < R_{sh} \leq \infty \, \Omega$.

Table 4.1 gives the classification of defects in the three main parts of RRAMs. Opens, bridges and shorts may occur in peripheral circuits. However, only opens and bridges may occur in the memory cell array and CNVs; shorts will not occur because these memory parts have no direct connection to the supplied voltage or ground (see Figure 2.5 of Chapter 2). Of all three parts, only defects in the memory cell array and CNVs are discussed in this thesis. This is because these

<table>
<thead>
<tr>
<th>Part</th>
<th>Classification</th>
<th>Locations</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory cell array</td>
<td>Open</td>
<td>Within cells</td>
<td>OC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At nanowire bit lines (NBLs)</td>
<td>OB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At nanowire word lines (NWLs)</td>
<td>OW</td>
</tr>
<tr>
<td>Bridge</td>
<td></td>
<td>Between NBLs</td>
<td>BB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between NWLs</td>
<td>BW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between NBLs and NWLs</td>
<td>BBW</td>
</tr>
<tr>
<td>CMOS to non-CMOS Vias</td>
<td>Open</td>
<td>Within short CNVs</td>
<td>OSV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Within tall CNVs</td>
<td>OTV</td>
</tr>
<tr>
<td>Bridge</td>
<td></td>
<td>Between short CNVs</td>
<td>BSV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between tall CNVs</td>
<td>BTV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Between short and tall CNVs</td>
<td>BSTV</td>
</tr>
<tr>
<td>Peripheral circuits</td>
<td>Open</td>
<td>At gate, source and drain of transistor, interconnects</td>
<td>OPC</td>
</tr>
<tr>
<td></td>
<td>Bridge</td>
<td>Among gate, source, drain and substrate of transistors, interconnects</td>
<td>BPC</td>
</tr>
<tr>
<td></td>
<td>Short</td>
<td>At gate, source and drain of transistors, interconnects</td>
<td>SPC</td>
</tr>
</tbody>
</table>
4.1. RRAM defects classification and definition

Two parts are those that distinguish RRAMs from the existing semiconductor memories. Defects in the peripheral circuits are similar to those in conventional RAMs [86, 87, 88]. In the rest of this section, defects within the cell array and in CNVs will be discussed.

4.1.1 Defects in memory cell array

Defects in the memory cell array can be either opens or bridges as summarized in Table 4.1 and briefly explained in this section. Opens can exist within the memory cells (i.e., memristor), at nanowire bit lines (NBLs) and nanowire word lines (NWLS). Conversely, bridges can be present between NBLs, between NWLS, as well as between NBLs and NWLS. Although bridges can impact an arbitrary number of adjacent nanowire lines regardless of their distance, only the ones that connect two physically adjacent nanowires are considered as they are assumed to have the highest occurrence probability.

1. Opens within memory cells, at NBLs and at NWLS.

Opens create an increased connection resistivity of the affected locations. For example in Figure 4.1(a), the defective cell C_{11} may suffer from insufficient dopant [80, 89, 90] that will cause its resistance to be extremely high (modeled as OC); hence affecting the write and read operations. In contrast, an open defect in NBLs (OB) and NWLS (OW) may occur due to, e.g., missing material and broken nanowire as a consequence of excessive force of nanoimprint lithography process, doping variation in the nanowire, or device width fluctuation as a result of variation in nanoimprint lithography and patterning process [43, 91]. For example in Figure 4.1(a), an open defect at NBL_2 (OB) affects operations to be applied to cells along the NBL such as C_{12}. Likewise, an open defect at NWL_2...
(OW) affects operations to cells along the NWL such as \( C_{22} \). Consequently, depending on the severity of the induced resistance, write and read operations to one or more cells might be affected.

2. **Bridges between NBLs, between NWLs, or between NBLs and NWLs**

Bridges create an unintended connection between NBLs (BB), between NWLs (BW), or between NBLs and NWLs (BBW); see Figure 4.1(b). These defect types can occur due to, e.g., an unintended longer nanowire that is connected to its adjacent nanowire, etc. [43]. For example, a bridge defect between NBL1 and NBL2 (BB) may flip the content of cell \( C_{21} \) when an operation is performed to \( C_{22} \). Likewise, a bridge defect between NWL1 and NWL2 (BW) causes a fault in \( C_{22} \) when an operation is performed to \( C_{12} \). Similarly, BBW that bypasses \( C_{22} \) may affect the write and read operations to the cell.

### 4.1.2 Defects in CMOS to Non-CMOS Vias

Defects in CNVs can be either opens or bridges [48, 92, 110, 111].

1. **Opens in tall and short CNVs**

Opens in short CNVs (OSV) and in tall CNVs (OTV) may occur due to, e.g., broken or crack as the result of excessive polishing during fabrication, void after filling, misalignment, poor contact between the top-edge of short (tall) CNVs and NWL (NBL), etc. [56, 57, 92]. For example in Figure 4.2, an open defect at SV1 (OSV) may affect operations applied to \( C_{11} \) and/or \( C_{12} \). Likewise, an open defect at TV1 (OTV) in Figure 4.2 may affect operations applied to \( C_{11} \) and/or \( C_{21} \).

![Figure 4.2: Possible opens and bridges within CNVs](image-url)
2. Bridges between tall CNVs and between short CNVs

Bridges between short CNVs (BSV), between tall CNVs (BTV), and between short and tall CNVs (BSTV) may occur due to excessive misalignment as a result of, e.g., imprecise mask [56, 57]. For example in Figure 4.2, a bridge defect between between SV₁ and SV₂ (BSV) may cause a fault in C₂₁ when operations are performed to C₁₁. Likewise, a bridge defect between TV₁ and TV₂ (BTV) may cause a fault in C₂₂ when operations are performed to C₂₁, etc.

4.2 Simulation model

This section introduces the three-dimensional (3D) RRAM model developed for defect injection and simulation using HSPICE and Verilog-AMS. First, the functional model of SPICE-based RRAM model will be described. Then, the RRAM cell electrical model based on memristor will be explained. Next, the logic state definition of memristor for memory application will be presented. Finally, the write/read timing requirements for memristor-based RRAM will be discussed.

Figure 4.3: Functional model of 3D RRAM
4.2.1 RRAM functional model

Figure 4.3 depicts the RRAM functional model of the 3D RRAM used in the simulation of this thesis [55]. The memory comprises three main parts: (i) CMOS peripheral circuits, (ii) non-CMOS cell array, and (iii) CNVs. The peripheral circuits consist of the functional units that are similar to those used in the existing semiconductor memories. Both row and column decoders operate together to access the selected memory cells in the memory cell array. The write/read circuits supply appropriate voltage levels for write and read operations. The row multiplexers connect the write/read circuits to the memory cell array during write 1 and read operations. The sense amplifiers sense read current, convert it into voltage and amplify it prior to sending the read value to data buffers. The column multiplexers connect the memory cell array to the sense amplifiers during read operations; they also connect the write circuits and memory cell array during write 0 operations. All these operations are controlled by the control circuit. Note that the nanowires in the cell array and CNVs are assumed to be simple interconnects where the resistance is negligible.

4.2.2 RRAM cell electrical model

A memristor is a two-terminal, non-linear nanodevice that can be used to represent logic data [58, 79]. Interestingly, its resistance (referred to as memristance) is retained even when power is switched off; thus, memristor can be used as a non-volatile storage element for memory applications. Figure 4.4(a) shows the symbol of a memristor where the bottom terminal is usually connected to the negative terminal, whereas the top terminal to the positive terminal.

Based on the theoretical formulation by L. Chua in 1971, a group of Hewlett-Packard (HP) researchers introduced the memristor prototype and the electrical model in 2008 [58]. The HP’s memristor is fabricated using a thin film of

![Figure 4.4: Memristor (a) symbol (b) diagram (c) electrical equivalent circuit](image-url)
4.2. Simulation model

titanium dioxide with different oxygen atom doping rates: doped and undoped, as shown in Figure 4.4(b). The doped layer has a thickness of $w$ and corresponds to a low-memristance $R_{\text{ON}}$, while the undoped layer has a thickness of $D-w$ and corresponds to a high-memristance $R_{\text{OFF}}$ where $D$ the total thickness of the memristor.

When a positive voltage is applied to the positive terminal of the memristor (the top terminal in Figure 4.4) while grounding the negative terminal (the bottom terminal in Figure 4.4), the dopants drift toward the undoped layer; thereby reducing the memristance. In contrast, when the polarity of the supply voltage is reversed, the dopants drift toward the doped layer; thereby increasing the memristance. When the voltage supply is removed, the dopants remain at their last position; thereby setting the memristance at a specific value. Figure 4.3(c) shows the equivalent electrical circuit of a memristor modeled as a coupled variable resistor where each resistor is coupled with its respective doped and undoped layer thickness (i.e., normalized to total memristor thickness) [58]. For example, if the doped (undoped) layer dominates the memristor, then the memristance becomes equal to $R_{\text{ON}}$ ($R_{\text{OFF}}$).

The current moving through a memristor is proportional to the flux of the magnetic field flowing across the device. Depending on the biasing sources, either current or voltage source, memristor can be controlled by charge or flux [55]. Because memory typically uses voltage at the source, a flux controlled memristor is considered and modeled. The voltage-current relation of memristor is expressed as follows [58].

$$v(t) = i(t)M(t)$$

$$v(t) = i(t)\left(R_{\text{ON}}\frac{w(t)}{D} + R_{\text{OFF}}\left(1 - \frac{w(t)}{D}\right)\right) \quad (4.1)$$

where $M(t)$ is the total memristance and $w(t)$ is the doped layer thickness of a memristor that changes with time. In order to satisfy the flux controlled model, $w(t)$ is normalized to $D$ [90]; it is referred to as the memristor internal state. Given $\Phi(t) = \int_0^T V(t)\,dt$ is the injected flux for $T$ period, $\mu_v$ is the average dopant mobility and $\beta = \frac{R_{\text{OFF}}}{R_{\text{ON}}}$, the memristor internal state is is expressed as follows.

$$\frac{w(t)}{D} = 1 - \sqrt{1 - \frac{2\mu_v\Phi(t)}{\beta D^2}} \quad (4.2)$$
Chapter 4. Defect-Oriented Test for RRAM

4.2.3 Logic state definition

The memristor internal state $\frac{w(t)}{D}$ is the metric used to define the logic value held by a memristor [58]. The value of $\frac{w(t)}{D}$ represents the change in the physical properties (dopant drift) of the memristor and is inversely proportional to its memristance. Because RRAM operates in binary logic value, yet memristors used as RRAM cells are an analog device, the logic state definition as proposed in [55] is used. As illustrated in Figure 4.5, logic 1 is defined for $0.6 \leq \frac{w(t)}{D} \leq 1$; whereas logic 0 is defined for $0 \leq \frac{w(t)}{D} \leq 0.4$. A safety margin $0.4 < \frac{w(t)}{D} < 0.6$ is set as undefined logic state. This undefined logic state should be alleviated so that the cell can operate optimally.

4.2.4 Memory operation time duration

In order to ensure that an ideal logic state is written, the write operation requires an ample write time $T_{\text{write}}$. For a given write voltage $V_{\text{write}}$ value, the required time for an appropriate write operation is expressed as follows [55].

$$T_{\text{write}} = \left| \frac{\phi(t)}{V_{\text{write}}R_{\text{OFF}}^2} \times (R^2_{\text{OFF}} - R^2_{\text{ON}}) \right|$$  \hfill (4.3)

where $\phi(t)$ is the effective flux for $0 < D \leq 1$ and is expressed as follows.

$$\phi(t) = \frac{(\beta D)^2}{2\mu_v(\beta - 1)}$$  \hfill (4.4)
Based on the memristor model specifications such as $R_{ON} = 100\, \Omega$, $R_{OFF} = 100\, k\Omega$, $D = 3\, nm$, $\mu_v = 3 \times 10^{-8} m^2 V^{-1} s^{-1}$ and $V_{\text{write}} = 1.5\, V$ [55, 90], the write time $T_{\text{write}} = 100\, ns$ is obtained. On the other hand, the read time is set to $20\, ns$; this short time is set to alleviate too much change in the memristor internal state that might lead to soft errors [55].

Figure 4.6(d) shows the timing control signal for write 1, read 1, write 0 and read 0 operations. The associate signals for each operation are as follows.

- For write 1 operation, the Write/read enable $\overline{\text{WRE}}$ signal is set to high for the duration of $100\, ns$ to initiate a write operation; at the same time, Input data of logic 1 sets the nanowire word line NWL voltage to $1.5\, V$ and the nanowire bit line NBL voltage to $0\, V$.

- For read 1 operation, the $\text{WRE}$ signal is set to low for the duration of $20\, ns$; at the same time, $-1.5\, V$ is applied to NWL for the first $10\, ns$ followed by $1.5\, V$ for the second $10\, ns$. Note that the pulse width for this negative and positive read voltage must be equal to have zero net flux effect [55]. At the same time, NBL is connected to sense amplifier. During the second half of the read operation, the Sense amplifier enable SE signal is activated.

- For write 0 operation, Input data of logic 0 sets the nanowire word line NWL voltage to $0\, V$ and the nanowire bit line NBL voltage to $1.5\, V$. 

Figure 4.6: Timing diagram of RRAM operation
For read 0 operation, $-1.5 \, V$ is applied to NWL for the first $10 \, ns$ followed by $1.5 \, V$ for the second $10 \, ns$. During this period, NBL is connected to sense amplifier.

4.3 Defect injection and circuit simulation

This section presents the electrical simulation results of open defects that impact the behavior of a single cell. Open defects are of interest because this defect type occurs most frequently in deep sub-micron circuits [93], and it is expected that the situation will be the same for emerging memories such as RRAM as well. In the rest of this section, first the simulation methodology will be explained, then the simulation results for defect-free memory will be presented. Thereafter, the simulation results of open defects and their analysis will be provided.

4.3.1 Simulation methodology

In order to ensure that the electrical RRAM model works correctly for defect injection and circuit simulation, it must be verified. Therefore, a defect-free simulation has been performed using the memory operations $S = w_1, r_1, w_0, r_0$; $w_1$ ($w_0$) denotes a write 1 (0) to the cell under test, while $r_1$ ($r_0$) denotes a read operation with the expected value 1 (0). The sequence $S$ is applied after initializing the cell to 0. It is worth noting that although the electrical memory model can operate multiple cells, only a single cell is considered in this work. Evaluation of multiple RRAM cells will be considered in the future.

![Simulation methodology](image)

Figure 4.7: Simulation methodology and the considered opens
After verifying that the model works correctly in the absence of defect, defect injection and simulation is performed. In order to perform this defect simulation, the same methodologies for conventional random access memories as illustrated in Figure 4.7 [84] are used; they are (i) defects injection to the target location and (ii) the functional fault analysis. For the first step, the considered defects are modeled as an additional resistance. The second step is performed by analyzing the deviation of the observed from expected cell behavior when the cell is sensitized with the appropriate memory operations $S$.

Open defects considered for this simulation are: (i) an open within memory cell $OC$, (ii) an open at nanowire word line $OW$, (iii) an open at nanowire bit line $OB$, (iv) an open within short via $OSV$, and (v) an open within tall via $OTV$. These defects are assumed to occur at one location at one time and are modeled by injecting a series resistor at the impacted location; e.g., $OW$ is modeled as $R_{op}$ between a nanowire word line and a memory cell. The resistance within the range of $10\Omega \leq R_{op} \leq 100\, M\Omega$ is increased in logarithmic scale. A smaller scale than logarithmic is used to when a specific behavior is observed such as the resistance that sets the memristor internal state $\frac{w}{D}$ to 0.4 and 0.6 (the undefined state boundaries).

Each time a single open defect is injected in the SPICE memory model, the following sequences are performed.

- $0w1$ – write 1 to a cell initialized to 0.
- $1w0$ – write 0 to a cell initialized to 1.
- $1r1$ – read an expected value 1 from a cell.
- $0r0$ – read an expected value 0 from a cell.

Finally, the functional fault analysis is performed where the behavior of the defective cell is observed for proper functionality. For this analysis, any difference between the observed and expected cell behavior is represented in the form of a fault primitive (FP), which is denoted as $<S/F/R>$. $S$ refers to the sensitizing operation as mentioned before, $F \in \{0,1\}$ describes the behavior or logic value of the faulty cell and $R \in \{0,1,-\}$ describes the logic output value for a read operation. For $R$, ‘-’ denotes no read operation is performed (thus no expected logic output value) for certain FPs [84]. For example, the FP $<0w1/0/->$ indicates that a cell containing logic 0 value is written with logic 1; however, it remains at initial logic 0 state and at the same time no logic output value is read out. The resulting faulty behavior is analyzed and the fault model is identified. If the faulty behaviors are similar to those of existing memories, then they are mapped into the established fault models. If the faulty behaviors are unique, a new fault models will be established. These fault models are then used to derive the appropriate test algorithms and schemes.
4.3.2 Defect-free simulation result

Figure 4.8 shows the defect-free simulation results for the sequence $S = w_1, r_1, w_0, r_0$. The figure consists of two graphs: (i) the top graph describes the memristor internal state of the cell (memristor) $\frac{w(t)}{D}$, which corresponds to the logic value being written or read, and (ii) the bottom graph shows the output data value sent to data buffer, which corresponds to the current sensed from the cell. Note that the control signals given in Figure 4.6 are used for the simulation. The resulting internal state and output data signals are as follows.

- For write 1 operation, $\frac{w(t)}{D}$ increases slowly (almost linearly) starting from 0. However, once $\frac{w(t)}{D}$ approaches 0.6, it increases quickly (exponentially) to reach 1. This is due to a slower ionic drift at smaller $\frac{w(t)}{D}$ values [58].

- For read 1 operation, $\frac{w(t)}{D}$ decreases from 1 to 0.64 before increasing back to 1. During second read period, the read current is sensed, is converted to voltage and sent to output data buffer at $t = 120\,\text{ns}$. Note that the first phase of the read operation is a destructive operation; therefore, the second phase is needed to get the cell to its initial state $\frac{w(t)}{D} = 1$.

- For write 0 operation, $\frac{w(t)}{D}$ decreases from 1 to 0 in a symmetric shape to that of write 1 operation.

![Figure 4.8: Simulation results of defect-free write and read operations](image-url)
4.3. Defect injection and circuit simulation

- For read 0 operation, \( w(t)/D \) remains at 0. The read value is sent to output data buffer at \( t = 240\,\text{ns} \).

4.3.3 Open defect simulation results

The simulation results for defect within the cell OC will be provided. Thereafter, a brief description for the other open defects will be given.

1. Open defect within the cell (OC)

Figure 4.9 shows the simulation results of 0w1 sequence for OC with different \( R_{op} \) values injected within the cell initially set to logic 0. It shows that the maximum value of the internal state at the end of write operation decreases as \( R_{op} \) increases; it even enters an undefined state for \( 19\,\text{k}\Omega \leq R_{op} \leq 56\,\text{k}\Omega \) (since \( 0.6 < \frac{w_D}{R_{op}} < 0.4 \)). For \( R_{op} > 56\,\text{k}\Omega \), \( \frac{w_D}{R_{op}} \leq 0.4 \) meaning that the write operation fails to set the defective cell to logic 1. Figure 4.10 shows the simulation results of 1w0 sequence for different \( R_{op} \) values injected within the cell initially set to logic 0. It clearly shows that depending on the defect value, the cell may fail to undergo a down write transition or enter the undefined state. Figure 4.11 shows the simulation results of 1r1 sequence for different \( R_{op} \) values injected. The figure shows that the defective cell returns an incorrect logic value while keeping the correct stored data when \( R_{op} > 21\,\text{k}\Omega \). The sequence 0r0 has also been performed for both opens within the cell; the results show no faulty behavior.

![Figure 4.9: Simulation result of OC for S = 0w1](image)

Figure 4.9: Simulation result of OC for \( S = 0w1 \)
2. Open defect for other locations (OW, OB, OSV, OTV)

The simulation results for an open defect in NWL (OW) exhibits the same faulty cell behaviors at slightly different open defect values. The faulty cell enters an undefined state when an open defect with values $14k\Omega \leq R_{op} \leq 49k\Omega$ is injected and sequence $w_1$ is performed; the same behavior is observed for sequence $1w_0$. When open defect value $R_{op} > 49k\Omega$ is injected and sequence
4.3. Defect injection and circuit simulation

When \( R_{\text{op}} > 17k\Omega \) is injected and sequence 1r1 is performed, the cell returns an incorrect logic value while keeping the correct stored data. For sequence 0r0, no faulty behavior is observed. Note that due to the symmetrical structure of the memory cell (see Figure 4.7), it is not required to perform the open defect simulation for all locations to derive the faulty behavior [85]. An open at NWL has a similar faulty behavior to that of an open at NBL, as well as at tall and short CNVs.

4.3.4 Simulation analysis

Table 4.2 summarizes the observed faulty behaviors of the defective RRAM cell including their corresponding simulated sequences, open resistance values and fault models. Depending on the defect values, two types of faults can occur:

- **Traditional faults**: these consists of e.g., Transition Faults (\( TF_0 \) and \( TF_1 \)), Stuck at Faults (\( SAF_0 \) and \( SAF_1 \)) and Incorrect Read Faults (\( IRF_1 \)) [94, 95]. These faults can be detected by existing quality tests.

- **Unique faults**: these consist of e.g., Undefined State Faults (\( USF_0 \) and \( USF_1 \)). Detecting these faults cannot be guaranteed with existing quality tests because the faults cause a random logic value to be read from the defective RRAM cells [29], while quality tests (march tests) deal with defined logic values (i.e., logic 0 or logic 1). Therefore, a special Design-for-Testability (DfT) scheme is required; this will be given in Chapter 5.

Table 4.2: Observed fault models for open defects within the memory cell (OC) and others (OW, OB, OSV and OTV)

<table>
<thead>
<tr>
<th>Simulated sequences</th>
<th>Defect values (OC, others)</th>
<th>Faulty behaviors</th>
<th>Fault models</th>
</tr>
</thead>
<tbody>
<tr>
<td>0w1, 1w0</td>
<td>( 19k\Omega \leq R_{\text{op}} \leq 56k\Omega, ) ( 14k\Omega \leq R_{\text{op}} \leq 49k\Omega )</td>
<td>The cell is set to an undefined state by a write 0 operation</td>
<td>USF_0</td>
</tr>
<tr>
<td></td>
<td>( R_{\text{op}} &gt; 56k\Omega, ) ( R_{\text{op}} &gt; 49k\Omega )</td>
<td>The cell is set to an undefined state by a write 1 operation</td>
<td>USF_1</td>
</tr>
<tr>
<td></td>
<td>( R_{\text{op}} &gt; 10M\Omega, ) ( R_{\text{op}} &gt; 10M\Omega )</td>
<td>The cell fails to undergo an down-transition when write 1 is performed</td>
<td>( TF_0 )</td>
</tr>
<tr>
<td></td>
<td>( R_{\text{op}} &gt; 10M\Omega, ) ( R_{\text{op}} &gt; 10M\Omega )</td>
<td>The cell fails to undergo an up-transition when write 0 is performed</td>
<td>( TF_1 )</td>
</tr>
<tr>
<td>1r1</td>
<td>( R_{\text{op}} &gt; 21k\Omega ) ( R_{\text{op}} &gt; 17k\Omega )</td>
<td>The cell returns a incorrect logic value 0 while it keeps its correct logic 1</td>
<td>( IRF_1 )</td>
</tr>
</tbody>
</table>
4.4 Summary

This chapter presented the framework of defect-oriented test for memristor-based RRAM. Several defects that are possible to occur in the emerging memories have been identified. As a case study, resistive open defects that impact a single RRAM cell have been simulated. The simulation results show that the resistive open defect that impact a single RRAM cell induces not only conventional semiconductor memories faults, but also new two faults. The new faults cause the impacted cell to enter an undefined state, which returns a random logic value when read operation is performed on the cell. Because it is difficult for quality test to detect these faults, a new test approach (e.g., Design-for-Testability) is required; this will be covered in the next chapter.

The main topics discussed are as follows.

- Classification and definition of possible defects in emerging memristor-based RRAMs. Among the three main units of this emerging memory, only defects in memory cell array and CMOS to Non-CMOS Vias are considered. Defects in the peripheral circuits of RRAM are not covered as this memory part is similar to that of conventional memories.

- Introduction of the SPICE simulation model of RRAM used to perform the defect injection and fault analysis in this chapter.

- Theoretical presentation of memristor devices employed as the memory cells in RRAM.

- Description of the methodology used for defect simulation and fault analysis.

- Simulation and analysis of a defect-free RRAM cell.

- Simulation and analysis of open defects that impacts a single RRAM cell.

- Establishment of new unique faults that correspond to the open defects in the RRAM cell.
CHAPTER 5

DESIGN-FOR-TESTABILITY SCHEMES FOR RRAM

5.1 DfT concept
5.2 Short Write Time based DfT
5.3 Low Write Voltage based DfT
5.4 Programmable DfT concept
5.5 Programmable Short Write Time based DfT
5.6 Programmable Low Write Voltage based DfT
5.7 Comparison of DfT schemes
5.8 Summary

The previous chapter evaluated open defects in the memory cell array of RRAM. Besides conventional memory faults, simulation results also showed the occurrence of unique faults. RRAM cells impacted by the unique faults will return random values (i.e., somewhere between $GND$ and $V_{dd}$) when they are read. As a consequence, manufacturing quality tests that are based on defined logic values (i.e., $V_{test} = GND$ or $V_{test} = V_{dd}$) cannot guarantee to detect the faults and in turn reduces the fault coverage. Hence, Design-for-Testability (DfT) schemes, which apply appropriate test stresses (e.g., $V_{test} < V_{dd}$), is required to facilitate manufacturing quality tests in detecting the unique faults and improve the fault coverage. Nevertheless, as they are designed at the pre-fabrication stage, the DfT scheme may not effective for use in the post-fabrication stage due to unexpected changes (e.g., process variations). Therefore, the DfT circuit has to be programmable to track the unexpected changes and prevent overkill.

In this chapter, the concept and development of DfT schemes to detect the two unique faults are presented. Section 5.1 describes the concept of the DfT schemes. Section 5.2 discusses the first DfT scheme including its circuit and simulation results, while Section 5.3 gives the second DfT scheme. Section 5.4 introduces the extended DfT schemes, which can be programmed for use in the post-fabrication stage. Section 5.5 discusses the design methodology, circuit and simulation results for the extended version of the first DfT scheme, while Section 5.6 gives the extended version of the second DfT scheme. Section 5.7 compares the DfT schemes in terms of fault detection capability, test time and practical implementation. Finally, Section 5.8 summarizes this chapter.
5.1 DfT concept

As mentioned in the previous chapter, open defects may cause RRAM cells to enter an undefined state and retain the state. Using a sequence of write and read operations of quality tests (i.e., march tests) cannot guarantee the detection of such faults as read operations will return random values. A possible way to improve the fault coverage of march tests and detect the Undefined State Faults is to stress the cell in such a way that:

- If the cell is faulty (i.e., undefined state), then stressing has to shift the cell’s state from an undefined state to an incorrect state. By performing a read operation after stressing the cell, the fault will be detected.
- If the cell is fault-free, then it has to remain in its correct state. Otherwise, the stress may lead to overkill and yield loss.

RRAM operations rely mainly on the duration of access time and the voltage value applied to the terminals of the cells [55, 58, 90]. Changing any of these parameters results in a test stress. One can exploit these two parameters to develop two DfT schemes as follows.

Duration of access time. Every normal write operation requires a specific write access time; for example, it is 100 ns as shown in Figure 4.8 of Chapter 4. If the access time is reduced, the cell will not have enough time to change its state from logic 0 (i.e., \( \frac{w(t)}{D} = 0 \)) to logic 1 (i.e., \( \frac{w(t)}{D} = 1 \)) or vice-versa; see Figure 4.8. However, if the cell was already in the undefined state (i.e., \( 0.4 < \frac{w(t)}{D} < 0.6 \)), then applying a write operation with a reduced write time will push the cell to shift from the undefined state to the defined state. This scheme is referred to as Short Write Time (SWT) and is illustrated in Figure 5.1(a). During a normal mode, a write operation (say \( w1 \)) is applied by setting the \( NWL_{sel} \) to \( V_{dd} \) for the normal access time \( T_{write} \); see also Figure 4.6. Thereafter, the DfT mode is enabled and a write operation, referred to as weak write (say \( \hat{w}0 \)), is applied by setting the \( NBL_{sel} \) to \( V_{dd} \) for a shorter access time \( T_{SWT} \). The DfT mode is then set-off and a normal read operation is applied (say \( r1 \)). If the cell suffers from an open defect, then the \( w1 \) operation will put the cell in an undefined state and the \( \hat{w}0 \) operation will shift the cell’s state to 0. The \( r1 \) operation will return an incorrect value 0 instead of 1 and the fault will be detected. If the cell is fault-free, then the weak write will not change the cell’s state and the read operation will return a correct value.

Applied voltage value. Every normal write operation requires a specific write voltage value; this is set to 1.5 V in the simulation. If the voltage supply is reduced, the induced electric field will not be sufficient to change the cell’s state from logic 0 (i.e., \( \frac{w(t)}{D} = 0 \)) to logic 1 (i.e., \( \frac{w(t)}{D} = 1 \)) or vice-versa; see Figure 4.8. However, if the cell was already in the undefined state (i.e.,
5.1. DfT concept

0.4 < \( \frac{w(t)}{D} < 0.6 \), then applying a write operation with a reduced write voltage will push the cell to shift from the undefined to the defined state. This scheme is referred to as Low Write Voltage (LWV) and is illustrated in Figure 5.1(b). During a normal mode, a write operation (say \( w_1 \)) is applied by setting the \( NWL_{sel} \) to \( V_{dd} \) for the normal access time \( T_{write} \); see also Figure 4.6. Thereafter, the DfT mode is enabled and a write operation, referred to as weak write (say \( \hat{w}_0 \)), is applied by putting the \( NBL_{sel} \) to a reduced supply voltage \( V_{LWV} \). The DfT mode is then set-off and a normal read operation is applied (say \( r_1 \)). If the cell suffers from an open defect, then the \( w_1 \) operation will put the cell in an undefined state and the \( \hat{w}_0 \) operation will shift the cell’s state to 0. The \( r_1 \) operation will return an incorrect value 0 instead of 1 and the fault will be detected. If the cell is fault-free, then the weak write will not change the cell’s state and the read operation will return a correct value.

In order to detect open defects within RRAM cells one can apply an appropriate test algorithm while incorporating one of the above DfT schemes, as it is done in [68] for SRAMs. The following test algorithm can be applied.

1. Initialize memory cells to 0.
2. Write 1 to memory cells.
3. Activate DfT (SWT or LWV) scheme and apply weak write 0; only defective cells will flip to 0.
4. Deactivate the DfT scheme and read 1 from the cells. If the read value is 0, then faults are detected.
5. Initialize the cells to 1.
6. Write 0 to the cells.
7. Activate the DfT scheme and apply weak write 1; only defective cells will flip to 1.
8. Deactivate the DfT scheme and read 0 from the cell. If the read value is 1, then faults are detected.

A detailed description of the two DfT schemes mentioned above will be given in the next two sections.

5.2 Short Write Time based DfT

As already mentioned, this scheme is based on reducing the write access time $T_{SWT}$ while maintaining the normal supply voltage $V_{dd}$. Identifying the duration SWT of weak write is crucial; it has to detect faulty cells but at the same time prevent overkill. In order to accomplish this, three steps are utilized as follows.

1. Set the targeted $w(t)$ boundaries of the undefined state that the DfT scheme has to shift to a defined state. In order to target cells impacted with Undefined State Faults, the upper and lower boundaries are set to $B_{high} = 0.6$ and $B_{low} = 0.4$, respectively.

2. Search for the critical open defect $R_{cr}$ value that causes $w(t)$ to enter the targeted boundary state. For this step, a defect injection and simulation are performed. For example, in Figure 5.2 when performing $0w1$ sequence in the presence of a defect, the $R_{cr}$ will cause $w(t)$ to be at the boundary of the correct state 1 and the undefined state; this corresponds to $\frac{w(t)}{B} = 0.6 = B_{high}$. The simulation results for different $R_{op}$ values for $0w1$ operation shows that the $R_{cr}$ is $R_{cr↓} = R_3 = 19k\Omega$; see also Figure 4.9 in Chapter 4. Figure 5.3 shows the simulation results for the $1w0$ operation where $R_{cr↑} = R_2 = 19k\Omega$; i.e., the value that cause the $\frac{w(t)}{B}$ to be at the boundary of the correct state 0 and the undefined state $\frac{w(t)}{B} = 0.4 = B_{low}$; see also Figure 4.10 in Chapter 4.
5.2. Short Write Time based DfT

Figure 5.2: Simulation results to determine $T_{SWT\downarrow}$

Figure 5.3: Simulation results to determine $T_{SWT\uparrow}$

3. Use the critical defect value $R_{cr}$ to determine the weak write duration that will cause the cell to shift from $B_{high}$ to $B_{low}$ and vice-versa. For example, for $R_{cr\downarrow} = R_3$ in Figure 5.2, the $\tilde{w}0$ will shift the cell from a state with $\frac{w(t)}{D} = 0.6$ to a state with $\frac{w(t)}{D} = 0.4$. This weak write operation will cause all defective cells with undefined state to shift to state 0, keeping healthy cells initially with $0.8 < \frac{w(t)}{D} \leq 1$ to remain in state 1, and weak cells with $0.6 \leq \frac{w(t)}{D} \leq 0.8$ to shift to undefined state. Therefore, the DfT will guarantee the detection of all defective cells that initially were put in
an undefined state. It may also detect some weak cells initially in a state with \(0.6 \leq \frac{\text{w}(t)}{D} \leq 0.8\). For this considered case, the required duration that shifts the \(\frac{\text{w}(t)}{D}\) from \(B_{\text{high}}\) to \(B_{\text{low}}\) for \(0\text{w}1\text{w}0\) operations is \(T_{\text{SWT}_\downarrow} = 24\text{ns}\); see Figure 5.2. A similar explanation can be given when identifying the \(T_{\text{SWT}_\uparrow}\) of the \(\hat{\text{w}}1\) operation required to shift the \(\frac{\text{w}(t)}{D}\) from \(B_{\text{low}}\) to \(B_{\text{high}}\) for \(1\text{w}0\text{w}1\) operations. Figure 5.3 shows that the required duration of \(\hat{\text{w}}1\) is \(T_{\text{SWT}_\uparrow} = 43\text{ns}\). Note that \(T_{\text{SWT}_\downarrow}\) requires shorter duration than \(T_{\text{SWT}_\uparrow}\) because the RRAM cell \(\frac{\text{w}(t)}{D}\) changes quickly at larger values but more slowly at smaller values; i.e., larger \(\frac{\text{w}(t)}{D}\) allows faster ionic drift and thus higher current [58].

### 5.2.1 DfT circuit

Two SWT circuits are required to perform \(\hat{\text{w}}0\) and \(\hat{\text{w}}1\) operations. The SWT circuits are embedded inside the write circuit shown in Figure 4.3. As illustrated in Figure 5.4, the concept of the SWT circuit that provide the required control signals for normal and \(\hat{\text{w}}0\) operations is given; the circuit consists of a Timer and three transistors. The NMOS transistor \(M_1\) and the PMOS transistor \(M_2\) are used to switch between the normal mode and DfT mode. When the DfTE signal is set to low, Timer will activate the \(T_{\text{write}}\) signal for a normal write operation (i.e., \(T_{\text{write}} = 100\text{ns}\)). However, when the DfTE signal is set to high, Timer will activate the \(T_{\text{write}}\) for the \(\hat{\text{w}}0\) operation (i.e., \(T_{\text{write}} = T_{\text{SWT}_\downarrow} = 24\text{ns}\) in the considered case). Subsequently, the \(T_{\text{write}} = T_{\text{SWT}_\downarrow}\) signal switches transistor \(M_3\) on allowing \(\hat{\text{w}}0\) to be supplied to the RRAM cell array. With the exception of different write duration (i.e., \(T_{\text{SWT}_\uparrow} = 43\text{ns}\)) set for Timer, the same components form the SWT circuit that supplies \(\hat{\text{w}}1\).

![Figure 5.4: Schematic of SWT circuit](image-url)
5.2.2 Simulation results

In order to evaluate the correctness of the proposed scheme, a simulation has been performed using three open resistance values: (i) $R_{op} = 0\, \Omega$, (ii) $R_{op} = 19\, k\, \Omega$, and (ii) $R_{op} = 33\, k\, \Omega$. For each injected open resistance, the test algorithm mentioned in the previous section is performed.

Figure 5.5 shows the result for the SWT scheme that performs $1w0\tilde{w}1r0$ sequence. Based on the figure, the following analyses can be drawn:

- $R_{op} = 33\, k\, \Omega$: the $\tilde{w}1$ operation will shift the cell’s state from the undefined state (at $t = 100\, ns$) to state 1 (at $t = 143\, ns$). Hence, a read 0 operation will easily detect the fault.

- $R_{op} = 19\, k\, \Omega$: this is the critical value. The $\tilde{w}1$ operation will shift the cell’s state from $B_{low}$ (at $t = 100\, ns$) to $B_{high}$ (at $t=143\, ns$). Although the read operation returns an incorrect value in this experiment, typically the read value can be considered random as the cell’s state is at the boundary.

- $R_{op} = 0\, \Omega$: the healthy cell keeps its state 0; hence the cell passes the test.

Therefore, the DfT scheme can guarantee the detection of any open defect with a value for $R_{op} > 19\, k\, \Omega$.

Figure 5.6 shows the result for the SWT scheme that performs $0w1\tilde{w}0r1$ operations. Based on the figure, the following analyses can be drawn:

- $R_{op} = 33\, k\, \Omega$: the $\tilde{w}0$ operation will shift the cell’s state from the undefined
Chapter 5. Design-for-Testability Schemes for RRAM

Figure 5.6: Simulation results of the SWT scheme for \(0\overline{w}1\overline{w}0\overline{r}1\) sequence; \(\overline{w}0\) is based on \(V_{dd}\) supplied for \(T_{SWT}\).

Based on the simulation results for the SWT scheme, one can conclude that the scheme is more suitable to be used with \(1w0\overline{w}1r0\) sequence.

5.3 Low Write Voltage based DfT

This scheme is based on reducing the level the write voltage \(V_{LWV}\) while maintaining the normal write access time \(T_{\text{write}}\). Similar steps as for the SWT scheme are used to determine the appropriate write voltage. Each time an open defect
5.3. Low Write Voltage based DfT

is injected, a different $V_{LWV}$ value is supplied to the RRAM cell under test and the sequences $\hat{w}_0$ and $\hat{w}_1$ are performed. Note that the weak write operations (i.e., $\hat{w}_0$ and $\hat{w}_1$) use a lower voltage than $V_{dd} = 1.5V$. The resulting cell’s state $\frac{w(t)}{D}$ curve is analyzed at the end of the weak write operation. If the $\frac{w(t)}{D}$ is shifted to the desired boundary ($B_{\text{high}}$ or $B_{\text{low}}$), then the required write voltage has been determined; if not, a different write voltage is supplied. As shown in Figure 5.7, to shift $\frac{w(t)}{D}$ from $B_{\text{high}}$ to $B_{\text{low}}$ after $100ns$, the required write voltage is $V_{LWV\downarrow} = 0.36V$. Conversely, to shift $\frac{w(t)}{D}$ from $B_{\text{low}}$ to $B_{\text{high}}$ after $T_{\text{write}} = 100ns$, the required write voltage is $V_{LWV\uparrow} = 0.64V$ as shown in Figure 5.8.
5.3.1 DfT circuit

As in the SWT scheme, two LWV circuits are required to perform \( \hat{w}0 \) and \( \hat{w}1 \) operations. Figure 5.9 shows the concept of the LWV circuits that provide the control signals for normal and \( \hat{w}0 \) operations. The LWV circuit consists of two transistors and two resistors. The NMOS transistors \( M_1 \) and PMOS transistor \( M_2 \) are used to switch between the normal mode and DfT mode. When the DfTE signal is set to low, \( V_{dd} \) is supplied for normal write operation. However, when the DfTE signal is set to high, a reduced voltage \( V_{LWV\downarrow} = \frac{V_{dd} \times R_2}{R_1 + R_2} \) is supplied for weak write operations. For instance, for the considered case where \( V_{LWV\downarrow} = 0.36 \) V, the resistors are set to \( R_1 = 1k\Omega \) and \( R_2 = 316\Omega \). With the exception of different value of the resistor values, the same components form the SWT circuit that supplies \( \hat{w}1 \).

5.3.2 Simulation results

The same simulation setup for the SWT scheme is used to simulate the LWV scheme. Figure 5.10 depicts the simulation result for \( 1w0\hat{w}1r0 \) sequence. Based on the figure, the following analyses can be made:

- \( R_{op}=33k\Omega \): the \( \hat{w}1 \) will shift the cell’s state from the undefined state (at \( t = 100ns \)) to state 1 (at \( t = 200ns \)). Hence, a read 0 operation will easily detect the fault.
- \( R_{op} = 19k\Omega \): this is the critical value. The \( \hat{w}1 \) will shift the cell’s state from \( B_{low} \) (at \( t = 100ns \)) to \( B_{high} \) (at \( t = 200ns \)). Although the read operation returns an incorrect value in this experiment, typically the read value can be considered random as the cell’s state is at the boundary.
- \( R_{op} = 0\Omega \): the healthy cell keeps its state 0; hence the cell passes the test.

Therefore, the DfT scheme can guarantee the detection of any open defect with a value of \( R_{op} > 19k\Omega \).
5.3. Low Write Voltage based DfT

Figure 5.10: Simulation results of the LWV scheme for 1w0\hat{w}1r0 operation; \( \hat{w}1 \) is based on \( V_{LWV} \) supplied for \( T_{write} \)

Figure 5.11: Simulation results of the LWV scheme for 0w1\hat{w}0r1 operation; \( \hat{w}0 \) is based on \( V_{LWV} \) supplied for \( T_{write} \)

Figure 5.11 shows the simulation results for the LWV scheme that performs 0w1\hat{w}0r1 sequence. Based on the figure, the following analyses can be made:

- \( R_{op} = 33k\Omega \): the \( \hat{w}0 \) will shift the cell’s state from the undefined state (at \( t = 100ns \)) to state 0 (at \( t = 200ns \)). Hence, a read 1 operation will easily detect the fault.

- \( R_{op} = 19k\Omega \): this is the critical value. The \( \hat{w}0 \) will shift the cell’s state
from $B_{\text{high}}$ (at $t = 100\text{ns}$) to $B_{\text{low}}$ (at $t = 200\text{ns}$). Although the read operation returns an incorrect value in this experiment, typically the read value can be considered random as the cell’s state is at the boundary.

- $R_{\text{op}} = 0\Omega$: the cell is supposed to keep its state $1$. However, the simulation results reveal that the state of the healthy cell is also shifted to the undefined state, which may cause overkill and yield loss. To prevent this drawback, the DfT scheme has to be adjustable to different voltage levels that suit the quality aim; hence a programmable DfT scheme is needed, this will be introduced in Section 5.4.

### 5.4 Programmable DfT concept

Even if they can detect the targeted open defects, both SWT and LWV schemes are designed at the pre-fabrication stage. Because of process variation, it is difficult to define accurate critical defect values, which in turn will be used to set the appropriate access time duration and voltage levels for the SWT and LWV schemes, respectively. As these DfT schemes apply weak write operations at a fixed strength, which is determined based on the best available pre-fabrication data, they might either understress or overstress the cells leading to test quality and cost issues. Thus, there are two major limitations of the DfT schemes:

1. Not possible to deal with parametric variation.
2. Possibility of yield loss due to overstress as shown in the previous section.

It is worth noting that redesign cannot solve this problem. To solve these limitations, both SWT and LWV DfT schemes must be modified so that they can be adjusted for different settings that suit the targets. The programmable SWT and LWV schemes are introduced next.

### 5.5 Programmable Short Write Time based DfT

If the SWT scheme is extended to have multiple access time durations that can be programmed, one (or more) weak write operations can be used to shift the cell to a desired defined state. Figure 5.12(a) shows the control signals of the PSWT scheme for multiple \( \hat{w}0 \) operations where a normal write voltage $V_{\text{dd}}$ is applied for different time durations $T_{\text{SWT}}$, when the DfT mode is activated; $p$ is an integer. The selection of the appropriate $T_{\text{SWT}}$ depends on the $p$-bit Selection signals.

In order to find these $p$ different write times, the design steps for the SWT scheme mentioned in Section 5.2 are used with a slight modification; they are as follows.
5.5. Programmable Short Write Time based DfT

DfT enable

Selection

NBL_sel

\[ V_{SWT} = V_{dd} \]

\[ -V_{dd} \]

\[ T_{write} \]

\[ T_{SWT} \]

\[ T_{read} \]

(a)

(b)

Figure 5.12: Control signals for (a) PSWT scheme (b) PLWV scheme

1. Set the \( p \) cell state boundaries that the PSWT scheme has to shift to a defined state. As a case study, \( p = 2^4 = 16 \) is used to have a sufficient precision at an acceptable area overhead cost. In addition, only memory sequence \( 1w0\hat{w}1r0 \) is considered. The low state boundary \( B_{low} \) can be represented by 16 possible \( \frac{w(t)}{D} \) points between \( B_0 = 0.45 \) and \( B_{15} = 0.3 \) with 0.01 steps.

2. Search for the 16 critical open defect \( R_{op} \) values that produce each targeted boundary state. For this step, open defects with various resistance values \( (1\Omega \leq R_{op} \leq 100k\Omega) \) are injected within the RRAM cell and \( 1w0 \) sequence is performed.

3. Use each of the critical \( R_{op} \) value determined in Step 2, and perform the simulation using the sequence \( 1w0\hat{w}1 \) in order to identify the required short time of \( \hat{w}1 \) that will cause the cell to reach boundary \( B_{high} = 0.6 \).

**PSWT circuit.** Figure 5.13 shows the PSWT circuit that consists of a 4-to-16 decoder, a Timer and a transistor. When the DfTE signal is set to low, the decoder is deactivated; during this duration, the normal operation is performed. When the DfTE signal is set to high, the decoder is activated. Depending on the selection signals \( S_1 \) to \( S_4 \), the decoder will set Timer to activate one of the \( T_{SWT} \) durations. For example, when \( S_1S_2S_3S_4 = 0000 \), the decoder sets Timer to activate the \( T_{write} \) signal for \( T_{SWT} \); when \( S_1S_2S_3S_4 = 0001 \), the \( T_{write} \) signal is activated for \( T_{SWT} \) and so on. Subsequently, the selected \( T_{SWT} \) signal then switches transistor \( M_1 \) on allowing the \( \hat{w}0 \) voltage
Figure 5.13: Schematic of PSWT circuit

Figure 5.14: Simulation results of PSWT scheme for $T_{SWT_6}$

to be supplied.

**PSWT simulation.** Figure 5.14 shows the simulation results for the PSWT scheme when it is programmed to one of the 16 weak write durations, i.e., based on $R_{c_6} = 17.93\, \Omega$ that requires $T_{SWT_6} = 45\, \text{ns}$. The figure shows that when using the appropriate $T_{SWT_p}$, an additional fault coverage can be guaranteed. For instance, for $R_{op} = 19k\Omega$, the cell’s state is shifted to the incorrect logic 1 state instead of the undefined state as it is the case for the SWT scheme. Hence, the fault will be detected.
5.6 Programmable Low Write Voltage based DfT

If the LWV scheme is extended to have multiple voltage values that can be programmed, one (or more) of the resulting electrical field will shift the cell to a desired defined state. Hence, using multiple supply voltage values (weak write settings) enables cells with different defect values to be shifted to the defined state when appropriate settings are applied. Figure 5.12(b) shows the control signals of the PLWV scheme for multiple \( w_0 \) operations where different reduced write voltages \( V_{\text{LWV}} \) are applied for a normal time duration \( T_{\text{write}} \) when the DfT mode is activated. The selection of the appropriate \( T_{\text{SWT}} \) depends on the \( p \)-bit Selection signals.

In order to find the different \( V_{\text{LWV}} \) values for the PLWV scheme, a similar methodology as that of PSWT is used. However, for each of the \( R_{\text{op}} \) values injected, different \( V_{\text{LWV}} \) were supplied during \( w_1 \) (the second write in the \( 1w0w1 \) sequence). The \( V_{\text{LWV}} \) that cause the faulty cell to shift to \( B_{\text{high}} = 0.6 \) are selected.

**PLWV circuit.** The PLWV circuit shown in Figure 5.15 comprises the same decoder as in the PSWT circuit, and a voltage divider circuit formed by transistors and parallel resistors. When the \( \text{DfTE} \) signal is set to low, \( M_{17} \) is switched on and the normal operation is performed. At the same time, the decoder is deactivated and \( M_{16} \) is switched off. When the \( \text{DfTE} \) signal is set to high, the PLWV operation is activated where the four selection signals \( S_1 \) to \( S_4 \) will select the appropriate transistors \( M_0 \) to \( M_{15} \). At the same time, transistor \( M_{16} \) is switched on to supply \( V_{\text{dd}} \) to the voltage divider circuit.

![Schematic of PLWV circuit](image-url)

**Figure 5.15:** Schematic of PLWV circuit
activated transistor (one of transistors $M_0$ to $M_{15}$) allows its corresponding voltage divider circuit to supply a reduced voltage $V_{LWV}$. For example, when $S_1S_2S_3S_4 = 1111$, the $Wk_{15}$ signal is activated to switch transistor $M_{15}$ on; this in turn supplies $V_{LWV_{15}} = V_{dd} \times \frac{R_{30}}{R_{30} + R_{31}}$ for the weak write operation.

**PLWV simulation.** Figure 5.16 shows the simulation results for the PLWV scheme when it is programmed to one of the 16 weak write durations, i.e., based on $R_{c6} = 17.93k\Omega$ that requires $V_{PLWV_6} = 0.67V$. The figure shows that for $R_{op} = 19k\Omega$, the cell’s state is shifted to the incorrect logic 1 state instead of the undefined state as it is for the LWV scheme.

It is worth noting that the main advantage of using one of the two programmable schemes is that it provides the possibility to adjust the stress (weak write setting) to prevent overkill and improve the fault coverage.

### 5.7 Comparison of DfT schemes

The proposed DfT and Programmable DfT schemes have their own advantages and disadvantages. In this section, the schemes are compared (in a relative manner) in terms of their capability to detect faults in the post-fabrication stage, the required testing time and their implementation; see Table 5.1. A brief discussion of the comparison is given next.

- **Fault detection capability:** The PSWT and PLWV schemes have a high fault detection capability due to their multiple weak write settings...
and programmability attributes. In contrast, the SWT and LWV scheme have a low fault detection capability due to their fixed weak write setting.

- **Testing time**: The SWT scheme requires the shortest time duration followed by the LWV scheme; these schemes apply a fixed weak write operation. In contrast, the PSWT scheme requires a long time duration and the PLWV scheme requires an even longer duration; this is because these PDfT schemes apply several weak write settings.

- **Practical implementation**: The SWT and PSWT schemes are easier to implement than the LWV and PLWV schemes. This is because the circuit of the first two schemes is easy to be tuned as their write time duration are quite high (i.e., $T_{\text{write}} \leq 35\text{ns}$). Furthermore, as the SWT and PSWT are based on time, their circuits can be realized using digital circuits such as counters. In contrast, to realize the LWV and PLWV schemes, data from fabrication process is required to set the appropriate voltage values. Moreover, the voltage divider circuits used in these schemes are an analog design, which demands more design effort than a digital design as for the SWT and PSWT schemes.

## 5.8 Summary

This chapter presented a test approach that combined march tests and DfT schemes to detect Undefined State Faults (USFs) and improve fault coverage of open defects in RRAM cell. The proposed DfT schemes are developed based on the access time and the voltage supply of RRAM cells. In addition to detecting USFs and improving the fault coverage, their circuits require an insignificant modification of the memory circuit. The schemes are also made programmable in order to improve fault coverage even further and prevent overkill, which arises due to process variations. Among the four DfT schemes proposed in this paper, Programmable Short Write Time scheme is the best scheme as it provides high fault detection capability (hence, high fault coverage and/or prevent overkill) and easy to implement in practice.
The main topics discussed are as follows.

- Introduction of two Design-for-Testability schemes used to detect resistive open defects in memristor-based RRAM.
- Description of the concept of the proposed DfT schemes that exploit the operation properties (time and voltage) of memristors.
- Establishment of the methodology employed to design the DfT schemes.
- Discussion of the DfT schemes including their design methodologies and circuits.
- Evaluation of the proposed DfT schemes that address undefined state faults in RRAM cells.
- Description of the extended version of the DfT schemes, which are programmable in dealing with unexpected changes to the circuit parameters due to process variations, as well as yield and quality/reliability trade-off.
Part III:
Fault-tolerant Architecture for Emerging Memories

This part presents the schemes used to develop a fault-tolerant architecture of the entire emerging memory system. Chapter 6 introduces two error correction codes (ECCs) that are modified from an existing symbol-based ECC; these modified ECCs tolerate faults in the memory cell array. Chapter 7 proposes a fault-tolerant scheme that is developed by combining one of the modified ECCs with a hardware redundancy scheme; this scheme tolerate faults in the memory cell array and decoders. Chapter 8 presents a fault-tolerance scheme that is developed by combining the existing symbol-based ECC with interleaving; this scheme tolerate faults in the memory cell array and vias.

Chapter 6: Tolerating Faults in Memory Cell Array
Chapter 7: Tolerating Faults in Memory Cell Array and Decoder
Chapter 8: Tolerating Faults in Memory Cell Array and Vias
6.1 Error correction codes
6.2 Conventional RRNS code
6.3 Six-Moduli RRNS code
6.4 Double Three-Residue code
6.5 Experimental evaluation
6.6 Summary

A memory cell array is one of the main units in a memory system in which data is temporarily stored for future retrieval. During operation, this unit is subject to faults causing erroneous data to be read from the memory system. In order to ensure that reliable data is read, error correction code (ECC) is typically used. Because it is expected that emerging memories will suffer from both clustered and multiple random errors, symbol-based ECC such as Redundant Residue Number System (RRNS) is required. However, the conventional RRNS (C-RRNS) code employed in the existing applications is subject to high area overhead and performance penalty. Consequently, if C-RRNS is applied without any modification, the implementation overheads might overwhelm the reliability benefits.

In this chapter, two modified versions of C-RRNS code are proposed to tolerate faults in memory cell array, each of which at reduced area and performance impact. Section 6.1 provides an overview of the RRNS circuit in an emerging memory. Section 6.2 presents the theory of C-RRNS code. Section 6.3 introduces the first modified version of C-RRNS referred to as Six-Moduli RRNS (6M-RRNS); this code aims to reduce the area overhead for a fixed memory word size while keeping the error correction capability at a competitive level. Section 6.4 proposes the second modified version of C-RRNS referred to as Double Three-Residue (D3R); this code aims to enhance the memory performance while maintaining the error correction capability at a competitive level. Section 6.5 gives the simulation results and analysis of the proposed 6M-RRNS and D3R codes. Finally, Section 6.6 summarizes this chapter.
6.1 Error correction codes

Considerable research work has applied ECCs to deal with faults in emerging memories, as well as existing memories [3, 81, 82, 83, 96, 97, 103]. ECCs possess the ability to correct errors concurrently at lower cost than the other fault tolerance schemes. These advantages enable ECCs to detect and correct faults on-the-fly during normal memory operation, and thereby improve the reliability. In order to have such fault tolerance capability, the memory must be equipped with error correction circuit.

Figure 6.1 illustrates the block diagram of an emerging memory that consists of memory cell array, and RRNS encoder and decoder; the former memory part is fabricated from non-CMOS devices, whereas the latter two memory parts are formed by CMOS devices. The encoder and decoder that operate RRNS code are added into a memory system at the input side and at the output side, respectively. The RRNS circuits and memory cell array are connected through the interconnects placed between them. These interconnects allow bidirectional data to be transferred between the RRNS circuit and memory cell array. For simplicity, the other peripheral circuits commonly required by a memory system such as sense amplifiers, write/read circuits, etc. are not considered in this block diagram.

At the input side of the memory, the RRNS encoder is used to transform the \( d \)-bit input data \( \text{Data-in} \) into an RRNS codeword \( b_s \) where \( 1 \leq s \leq n \) and \( n \) is an

![Figure 6.1: ECC scheme in an emerging memory](image-url)
integer. The RRNS codeword is then stored in the memory cell array. During the course of this storage phase, faults might impact the stored codewords. For example, transient and intermittent faults cause the stored codewords to flip; this phenomenon results in soft errors.

At the output side of the memory, the RRNS decoder is used to transform the retrieved codewords into the output data $Data_{-out}$. The decoding process will ensure that $Data_{-out}$ is free from faults (identical to $Data_{-in}$), provided that: (i) the retrieved data is inherently free from faults and (ii) the fault counts are still within the error correction capability of the RRNS code. If the retrieved codeword has an error count larger than the error correction capability of the RRNS code, then an uncorrectable signal $Valid$ will be set to low to indicate the problem to the other blocks in a computer system. This indication will acknowledge the computer system to initiate a higher level of fault tolerance or to ignore $Data_{-out}$.

### 6.2 Conventional RRNS code

This section discusses the concept of conventional RRNS code including its encoding and decoding procedures. The term *conventional* is used to distinguish between the typical RRNS code and the two modified RRNS codes proposed in this thesis. Examples of both RRNS encoding and decoding procedures are given in Appendix A1 and A2.

#### 6.2.1 RRNS code

An RRNS codeword is constructed from a set of encoded binary digits referred to as *residues* (or symbols). Figure 6.2 depicts the structure of an $n$-residue RRNS codeword representing a $d$-bit input data. Note that RRNS code belongs to symbol-based ECC as mentioned in Chapter 3.

The RRNS codeword is divided into two sets of residues: (i) *non-redundant residues* $x_i$ consisting of $k$-residue dataword and (ii) *redundant residues* $x_j$ consisting of $(n-k)$-residue checkword (parity), where $1 \leq i \leq k$ and $k+1 \leq j \leq n$. The bit size of $k$-residue may be larger than that of $d$-bit input data ($k \geq d$) in

<table>
<thead>
<tr>
<th>RRNS codeword</th>
<th>Checkword</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$ $x_2$ $x_3$ $\cdots$ $x_k$ $x_{k+1}$ $x_{k+2}$ $x_{k+3}$ $\cdots$ $x_n$</td>
<td></td>
</tr>
</tbody>
</table>

Non-redundant residues | Redundant residues

Figure 6.2: Structure of RRNS code
order to provide the residue number system representation of the \(d\) bits. The RRNS code is able to detect up to \(h = (n-k)\) residues and correct up to \(t = \frac{(n-k)\cdot 2}{2}\) residues. Note that RRNS code has a similarity in terms of the structure as well as the detection and correction capability to that of Reed-Solomon code [102].

The residues \(x_i\) and \(x_j\) are generated by performing modulo operation on the input data \(X\) to a set of non-redundant moduli \(m_i\) and redundant moduli \(m_j\), respectively; \(X\) represents the integer value of \(d\)-bit input data. These operations can be mathematically represented with the following equations [98].

\[
x_i = |X|_{m_i} = X \mod m_i
\]  
(6.1)

\[
x_j = |X|_{m_j} = X \mod m_j
\]  
(6.2)

The bit length of \(x_i\) is \(b_i = \lfloor \log_2(m_i-1) + 1 \rfloor\) bits and that of \(x_j\) is \(b_j = \lfloor \log_2(m_j-1) + 1 \rfloor\) bits. For example, for \(m_i = 8\), the bit length of \(x_i\) is \(\lfloor \log_2(8-1) + 1 \rfloor = 3\) bits. This is true because for \(m_i = 8\), \(0 \leq x_i \leq 7\) each of which is represented by 3 bits. Summing \(b_i\) and \(b_j\) will result in the total bit size of the RRNS codeword \(b_s\).

For an RRNS codeword to be consistently decoded (i.e., to prevent a codeword to be decoded into more than one output data), \(m_i\) must satisfy three requirements [98, 99, 100]:

1. A pair of any two moduli, say \(m_a\) and \(m_b\) with \(a \neq b\), must be relatively prime positive integers such that their greatest common divisor \(\gcd(m_a, m_b) = 1\).
2. The integer value of succeeding modulus is greater than the preceding modulus, i.e., \(m_1 < \ldots < m_k < m_{k+1} < \ldots < m_n\).
3. The product of non-redundant moduli \(M_i = \prod_{j=1}^{k} m_j\) is sufficient to represent all numbers in the operating legitimate range of input data \(M_{op} = [0, 2^d - 1]\) for \(d\)-bit input data (memory word size); this means that \(M_i \geq M_{op}\).

In addition to the three requirements mentioned above, \(m_j\) are chosen arbitrarily such that:

1. They ensure the desired error correction capability.
2. Their product \(M_j = \prod_{j=k+1}^{n} m_j\) is sufficient to represent all the numbers in \(M_{op}\); this means that \(M_j \geq M_{op}\).
6.2. Conventional RRNS code

The conventional RRNS (C-RRNS) code comprises nine residues, where there are (i) three restricted non-redundant moduli and (ii) six unrestricted redundant moduli. The restricted non-redundant moduli set is based on \{2^p-1, 2^p, 2^p+1\} where \(p\) is a positive integer; this moduli set is adopted to realize simple and fast hardware implementation [103, 104]. Unrestricted redundant moduli are commonly appended to the non-redundant moduli in order to provide the fault detection and the correction capability. The restricted non-redundant moduli are selected in such a way that, besides satisfying the RRNS rules, the product of the selected moduli should be as close as possible to the operating legitimate range so as to minimize the area overhead and performance penalty. This selection is accomplished by choosing the minimum value of \(p\). On the other hand, the number of the unrestricted redundant moduli \(q_j\) (where \(1 \leq j \leq 2t\)) can be any integer larger than the value of the chosen non-redundant moduli as long they conform the RRNS rules. Prime integers larger than those of the restricted non-redundant moduli are chosen to be the redundant moduli.

For 16-bit memory word, the integer value for the restricted non-redundant moduli \(\{2^p-1, 2^p, 2^p+1\}\) must be selected so that their product is at least equal to the operating legitimate range \(M_{op} = 2^{16} - 1 = 65535\). In order to satisfy this criterion, the minimum value of \(p = 6\) is chosen. This results in non-redundant moduli \(m_3 = \{63, 64, 65\}\); note that their product \(M_j = 63 \times 64 \times 65 = 262080 > M_{op}\). The chosen redundant moduli are \(m_j = \{67, 71, 73, 79, 83, 89\}\); their product is obviously larger than \(M_{op}\). In a similar way, \(p = 11\) is chosen for 32-bit and \(p = 22\) is chosen for 64-bit. The six-residue checkword is needed to protect the three-residue dataword, i.e., \(t = 3\). The codeword size is \(b_s = \sum_{i=1}^{k} \left\lfloor \log_2 (m_i - 1) + 1 \right\rfloor + \sum_{j=k+1}^{n} \left\lfloor \log_2 (m_j - 1) + 1 \right\rfloor = 61\) bits.

6.2.2 RRNS encoding process

RRNS encoding is based on modulo operation of the input data to a moduli set as explained in Section 6.2.1. The residues resulting from the modulo operations are obtained simultaneously; i.e., the operations are executed in parallel by the corresponding modulo circuits. These generated residues are then concatenated to create an RRNS codeword before being stored in the memory cell array. An example of this scheme is given in Appendix A1. Alternatively, the redundant residues can be generated from the non-redundant residues using a scheme known as Base Extension (BEX) [98]. However, this scheme requires an iterative calculation, which makes the encoding slower.

6.2.3 RRNS decoding process

RRNS decoding begins by validating the codeword read from the memory array. The codeword is regarded as valid if its decoded value is within \(M_{op}\); if it is valid, no error correction is needed. On the other hand, it is regarded as invalid if its value is equal to or larger than \(M_{op}\); if this is the case, a correction process
is needed. During the correction process, an iterative calculation is executed. In each iteration, \( t \) number of residues will be discarded. Subsequently, the decoding procedure will calculate the decoded value using the remaining \( (n - t) \) residues. This exhaustive calculation will be performed until the valid data is recovered for maximum \( C_n^t = \frac{n!}{t!(n-t)!} \) times of iterations, where \( n \) is the number of residues and \( t \) is the error correction capability; see Section 6.2.1.

Ideally, whenever the erroneous residues are discarded, the correct data will be recovered and read out. However, when no data less than \( M_{op} \) is recovered after the maximum iteration, the decoding cannot correct the erroneous residues in the RRNS codeword. Subsequently, an uncorrectable output signal will be invoked (e.g., Valid signal is set low) so that the output data can be ignored or can be further processed by a system equipped with multilevel fault tolerance.

### 6.2.4 Decoding algorithms

Two algorithms can be used in the decoding process, either (i) **Chinese Remainder Theorem (CRT)** or (ii) **Mixed-Radix Conversion (MRC)** [98]. CRT is the classical approach to convert residue number system into binary or decimal. This conversion approach executes in parallel but its large legitimate operating range results in complex circuits. A decimal number \( X \) can be obtained from a set of residue representation \( \{x_1, x_2, \ldots, x_n\} \) using CRT approach as follows [98],

\[
X = \left( \frac{M_1|x_1 M_1^{-1}|m_1}{} + \left( \frac{M_2|x_2 M_2^{-1}|m_2}{M} \right) + \left( \frac{M_3|x_3 M_3^{-1}|m_3}{M} \right) + \cdots + \left( \frac{M_n|x_n M_n^{-1}|m_n}{M} \right) \right]
\]

\[
X = \left[ \sum_{s=1}^{n} M_s|x_s M_s^{-1}|m_s \right]_M
\]  

(6.3)

where for \( s = 1 \) to \( n \), \( x_s \) are the residues, \( M = \prod_{s=1}^{n} m_s \) is theoretical RRNS legitimate range, \( M_s = \frac{M}{m_s} \) are modular multiplicatives and \( |M_s^{-1}|m_s \) are modular multiplicative inverse of \( M_s \) with respect to \( m_s \) satisfying \( |M_s \times M_s^{-1}|m_s = 1 \).

MRC is another conversion approach used to convert a residue number system into a decimal (binary in computer systems). MRC is based on mixed radix number system that executes sequentially. Although MRC operates sequentially, the resulting circuit is simpler and is easier to optimize than that of CRT. A decimal number \( X \) can be obtained from a set of residue representation \( \{x_1, x_2, \ldots, x_n\} \) using MRC approach as follows [98],

\[
X = v_1 + (v_2 m_1) + (v_3 m_1 m_2) + \cdots + \left( v_{n-1} \prod_{s=1}^{n-1} m_s \right)
\]

(6.4)
where $m_s$ are the moduli and $v_s$ are mixed radix digits calculated as follows.

\begin{align*}
v_1 &= x_1 \\
v_2 &= \left( x_2 - v_1 \right) g_{12} \pmod{m_2} \\
v_3 &= \left( \left( x_3 - v_1 \right) g_{13} \right) - v_2 g_{23} \pmod{m_3} \\
&\vdots \\
v_s &= \left( \left( x_s - v_1 \right) g_{1s} \right) \ldots - v_{(s-1)} g_{(s-1)s} \pmod{m_s}
\end{align*}

where $x_s = |X|_{m_s}$ and $g_{(s-u)s}$ are the multiplicative inverses of $m_{(s-u)}$ with respect to $m_s$ satisfying $|m_{(s-u)} \times g_{(s-u)s}|_{m_s} = 1$; $2 \leq s \leq n$ and $1 \leq u \leq n-1$. An example of these two algorithms is given in Appendix A2.

### 6.3 Six-Moduli RRNS code

Due to its varied moduli length, C-RRNS produces a longer codeword size as the memory word size increases. This results in either large area overhead for a fixed memory word length or reduced capacity of useful stored data for a fixed memory chip size. Nevertheless, C-RRNS has better error correction capability than the other symbol-based ECCs such as Reed-Solomon code. Therefore, it is beneficial to modify C-RRNS to reduce the length of C-RRNS codeword, while retaining its competitive correction capability. In order to achieve this aim, Six Moduli RRNS (6M-RRNS) is introduced, which is based on the following strategies [32].

**For non-redundant moduli.** The number of non-redundant moduli is minimized as long as their product is larger than the operating legitimate range. This results in a shorter checkword, hence a shorter codeword. For example, Table 6.1 shows that C-RRNS code uses three non-redundant moduli $(2^p-1, 2^p, 2^p+1)$ to encode the dataword, while only two are needed $(2^p+1, 2^p)$ for 6M-RRNS code; see Figure 6.1. The appropriate $p$ values given in the
Table 6.1: Moduli and $p$ values for C-RRNS, 6M-RRNS and D3R codes

<table>
<thead>
<tr>
<th>ECC types</th>
<th>Moduli set</th>
<th>$p$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>16-bit</td>
</tr>
<tr>
<td>C-RRNS</td>
<td>$2^p - 1, 2^p, 2^p + 1, q_1, q_2, q_3, q_4, q_5, q_6$</td>
<td>6</td>
</tr>
<tr>
<td>6M-RRNS</td>
<td>$2^p, 2^p + 1, 2^{p-1} - 1, 2^{p-2} - 1, 2^{p-3} - 1, 2^{p-4} + 1$</td>
<td>8</td>
</tr>
<tr>
<td>D3R</td>
<td>$2^p - 1, 2^{p+1} - 1, 2^{p+1}, 2^p - 1, 2^{p+1} - 1, 2^{p+1}$</td>
<td>8</td>
</tr>
</tbody>
</table>

The table ensures that the considered moduli product is larger than the operating legitimate range. Thus, to correct its three-residue dataword, C-RRNS code requires six residues as the checkword; while 6M-RRNS code only requires four residues. More analysis will be given in Section 6.5.

For redundant moduli. The redundant moduli with smaller integer values than those of the non-redundant moduli are chosen; yet the chosen moduli have product larger than the operating legitimate range. Table 6.1 shows that C-RRNS code uses redundant moduli $(q_1, q_2, q_3, q_4, q_5, q_6)$ with values that are larger than those of the non-redundant moduli. In contrast, 6M-RRNS code uses redundant moduli with values that are smaller than those of non-redundant moduli. The appropriate $p$ values given in the table ensure that the product of the chosen redundant moduli is larger than the operating legitimate range. Note that this strategy violates the second requirement of RRNS coding as mentioned in Section 6.2.1. Such violation may cause an inconsistency during decoding where a single read codeword might be decoded into more than one (ambiguous) output data. In order to solve this problem, maximum likelihood decoding is introduced.

Maximum likelihood decoding (MLD). This method is used to determine the authentic output data from the ambiguous decoded data. More details on this method will be given in Section 6.3.1.

The 6M-RRNS code consists of six residues, where there are (i) two non-redundant moduli and (ii) four redundant moduli [35]. Based on the proposed modification strategies mentioned above, the non-redundant moduli are $m_1 = \{2^p, 2^p+1\}$, while redundant moduli are $m_f = \{2^{p-1} - 1, 2^{p-2} - 1, 2^{p-3} - 1, 2^{p-4} + 1\}$ where $p$ is an integer. Note that both moduli sets of 6M-RRNS are low-cost moduli, while this is only the case for the non-redundant moduli for C-RRNS. These low-cost moduli sets will result in a simpler hardware implementation than that of C-RRNS [98].

The second row of Table 6.1 shows the generic moduli set for 6M-RRNS and the appropriate $p$ values that fulfill four of the five RRNS coding requirements. For 16-bit memory word, the smallest $p$ value satisfying the requirement that
the product of non-redundant moduli is larger than $M_{op} = 2^{16} - 1$ is $p = 8$.
This results in $m_i = \{257, 256\}$, $M_i = 65792$, $m_j = \{127, 63, 31, 17\}$ and $M_j = 4216527$. Although $m_j$ consist of smaller integer values than those of $m_i$, their product $M_j$ is clearly bigger than $M_{op} = 2^{16} - 1$. In a similar way, $p = 16$ is chosen for 32-bit and $p = 32$ is chosen for 64-bit memory word.

6.3.1 Maximum likelihood decoding

As mentioned before, the second modification strategy intentionally violates one of the RRNS requirements; this results in some of the read codewords might be decoded into ambiguous decoded data. In order to solve this ambiguity, a maximum likelihood decoding (MLD) scheme, adopted from [101], is employed. The concept behind the scheme is to find the smallest difference (Hamming distance) between the residues of the ambiguous decoded data and the read codeword that causes the ambiguity. First, each ambiguous decoded data is encoded resulting in a new codeword. Then, the Hamming distance between each new codeword and the read codeword (i.e., the codeword that produce the ambiguous data) is calculated. Finally, the new codeword (of the ambiguous decoded data) that has the smallest Hamming distance with respect to the read codeword will be regarded as the authentic output data. Furthermore, as the moduli set used for 6M-RRNS satisfies the first and third RRNS requirement, MLD ensures the decoding correctness. An example of this scheme is given in Appendix A3.

6.3.2 6M-RRNS encoding and decoding process

Although 6M-RRNS is modified from C-RRNS, the modification is only on the number of moduli sets required and their values. Therefore, 6M-RRNS is still based on the same encoding and decoding steps of C-RRNS. The 6M-RRNS encoding uses modulo operation that operates in parallel. The 6M-RRNS decoding uses the MRC conversion algorithm to convert the codeword into a decimal prior to the detection and correction phase.

6.4 Double Three-Residue code

Similar to the other symbol-based ECCs such as Reed-Solomon code, C-RRNS code provides better correction capability than bit-based ECC such as Hamming code [40, 102]. However, this advantage comes with high computation latency (especially for decoding) leading to performance penalty. Therefore, it is advantageous to modify C-RRNS in such a way that it performs fast decoding while retaining its competitive correction capability. In order to achieve this aim, Double Three-Residue (D3R) code is introduced; it is based on the following strategies [32].
Chapter 6. Tolerating Faults in Memory Cell Array

Concept. D3R only performs detection during decoding process. This concept is different from that of C-RRNS (and 6M-RRNS), which performs detection followed by correction (if errors are detected) during decoding.

Code structure. D3R code consists of two copies of codewords, each of which is formed by a two-residue dataword and one-residue checkword. Figure 6.3 illustrates that a D3R code consists of an RRNS codeword ($C = x_1, x_2, x_3$) and its duplicate ($C' = x_1', x_2', x_3'$). This code is different from C-RRNS code, which comprises $n$-residue dataword and $2n$-residue checkword where $n$ is an integer.

Decoding process. D3R code swaps the residues between its codeword copies and compares their values to the operating legitimate range simultaneously. This decoding process is different from that of C-RRNS where one residue is discarded in each decoding iteration and the decoding is performed iteratively.

6.4.1 D3R encoding process

A D3R code consists of an RRNS codeword ($C = DW + CW$) and its duplicate ($C' = DW' + CW'$) as depicted in Figure 6.3. The D3R code is encoded based on the moduli set $m = \{2^p-1, 2^{p+1}-1, 2^{p+1}\}$ [36] where $p$ is half of the memory word size. Such a moduli set is the low-cost moduli, which result in faster performance and lower area overhead [98]. The first two moduli are used to generate the two-residue dataword $DW = x_1, x_2$ and its duplicate $DW' = x_1', x_2'$; while the third modulus is used to produce the single-residue checkword $CW = x_1$ and its duplicate $CW' = x_1'$. The bit size of the D3R codeword is $b_{\text{D3R}} = 2 \times (\lceil \log_2(m_1 - 1) + 1 \rceil + \lceil \log_2(m_2 - 1) + 1 \rceil + \lceil \log_2(m_3 - 1) + 1 \rceil)$ where $m_1, m_2$ and $m_3$ are the moduli. An example of this encoding is given in Appendix A4.

Three moduli are used to ensure the detection of a single erroneous residue, i.e., $h = n-k = 3-2 = 1$. This implies that each D3R codeword part ($C$ and $C'$) is able to detect a single erroneous residue. In case of faults impact only one of the D3R parts (yet another one is fault-free), this code can tolerate up to three erroneous residues. Such capability is better than that of the error correction capability possessed by RRNS and Reed-Solomon [35], when assuming that the codes are composed of an equal number of residues. For example, let us assume...
6.4. Double Three-Residue code

An RRNS code consists of six residues \((n = 6)\), where two residues represent the dataword \((k = 2)\) and four residues represent the checkword \((n-k = 4)\). With this setting, RRNS can only correct up to \(\frac{n-k}{2} = \frac{6-2}{2} = 2\) residues [98, 103].

6.4.2 D3R decoding process

As mentioned in Section 6.2.3, RRNS decoding requires an iterative correction when errors are detected; thus, it impacts the performance of the decoder. However, this is not the case for D3R as it does not discard any residues during the correction phase but swaps between the two codeword parts. Both D3R codeword parts are then decoded simultaneously. The maximum number of swapping is equal to half of the number of residues in a codeword, i.e., \(\frac{n}{2}\) (as compared to \(C^n\) for RRNS). Hence, due to its structure and decoding nature, D3R decoding is faster than that of RRNS. The following steps describe the D3R decoding.

1. Convert \(C = x_1, x_2, x_3\) and \(C' = x'_1, x'_2, x'_3\) into binary data \(B_1\) and \(B'_1\), each.
2. Compare \(B\) and \(B'\) to the operation legitimate range \(M_{op} = 2^d - 1\) where \(d\) is the memory word size.
3. If \((B = B') \leq M_{op}\), read out \(B\) or \(B'\); Break
   else if \(B \leq M_{op}\), read out \(B\); Break
   else if \(B' \leq M_{op}\), read out \(B'\); Break
   else go to Step 4.
4. Check if the maximum number of swapping has been reached. If “yes” go to Step 6, else go to Step 5.
5. Swap one of the residues between \(C\) and \(C'\). Go to Step 1.
6. Ignore the output data and invoke a flag indicating that the data is uncorrectable.

Step 1 converts D3R code into binary data based on the MRC algorithm. As mentioned in Section 6.2.4, MRC is preferable to the Chinese Remainder Theorem algorithm because the former algorithm requires a simpler circuit, which is easier to be optimized; it also speeds up the simulation work.

Step 5 swaps the residues between the two D3R codeword parts. Table 6.2 gives the swapped residue combinations for each iteration. For example, the first residues between \(C\) and \(C'\) are swapped during the first iteration in such a way that \(x_1\) becomes a part of \(C'\) and \(x'_1\) a part of \(C\). Similar steps are performed to the second and third iterations. The residue to be swapped is selected at random, i.e., no fixed rule is set. An example of this decoding is given in Appendix A5.
Table 6.2: Residues set for each iteration of D3R correction

<table>
<thead>
<tr>
<th>Number of Iteration</th>
<th>Residues</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original part (C)</td>
</tr>
<tr>
<td></td>
<td>Replicated part (C')</td>
</tr>
<tr>
<td>1</td>
<td>$x_1'$, $x_2$, $x_3$</td>
</tr>
<tr>
<td></td>
<td>$x_1$, $x_2'$, $x_3'$</td>
</tr>
<tr>
<td>2</td>
<td>$x_1$, $x_2$, $x_3$</td>
</tr>
<tr>
<td></td>
<td>$x_1'$, $x_2$, $x_3'$</td>
</tr>
<tr>
<td>3</td>
<td>$x_1$, $x_2$, $x_3'$</td>
</tr>
<tr>
<td></td>
<td>$x_1'$, $x_2'$, $x_3$</td>
</tr>
</tbody>
</table>

6.5 Experimental evaluation

This section presents the experimental evaluation and analysis of the modified versions of RRNS code. The modified versions are compared with existing ECCs such as C-RRNS and Reed-Solomon (RS). Three attributes are compared among the ECCs: fault tolerance capability, area overhead and decoding time performance.

6.5.1 Experimental setup

The simulation model consists of $4K \times 64$-bit memory system, which is described using Matlab script. The parameters for 6M-RRNS and D3R code are as described before; while that for the existing symbol-based ECCs are as follows:

- **C-RRNS** [103]: This ECC always consists of three residues representing the dataword. For the correction capability of these three residues, C-RRNS has to make use of six residues representing the checkword. Therefore, the moduli set of $\{2^p-1, 2^p, 2^p+1, q_1, q_2, ..., q_6\}$ will be used; where $p$ is the minimum integer satisfying the RRNS requirements and $q_j$ is a prime number larger than $2^p+1$ [103]. The first three moduli are used to encode the dataword and the last six moduli to encode the checkword. For the experiment, the $p$ values are 6, 11, and 22 for 16, 32, and 64-bit memory word, respectively.

- **RS** [102]: This ECC consists of two symbols representing the dataword. For the correction of these two symbols, RS has to make use of four symbols representing the checkword. Galois Field of degree $w$ denoted as $GF(2^w)$ is used where each symbol consists of a $w$-bit data satisfying RS requirement [102]. For the experiment, the $w$ values are 8, 16, and 32 for 16, 32, and 64-bit memory word, respectively.

Faults were randomly injected into the memory cell array with fault rates of 1% to 10%. The faults flip a number of adjacent bits (representing the clustered and multiple random errors) that form the codewords of the considered ECCs.
6.5.2 Fault tolerance capability

Figure 6.4 shows the capability of the considered ECCs to tolerate faults that induce clustered and multiple random errors in the memory cell array. This capability is represented by the percentage of corrected memory words. Regardless of the fault rates, C-RRNS provides the best correction capability among the considered ECCs followed by D3R, 6M-RRNS and RS. However, the difference between C-RRNS and D3R is negligible, e.g., it is only 0.2% at 10% fault rate. The difference becomes even smaller for memories with larger words [35].

The fault tolerance capability for each ECC relies on its error correction capability or error detection capability, which is set at initial design phase. In this thesis, the set parameters for the considered ECCs result in the error correction capability as follows.

- **D3R**: This code ensures the correction of up to three erroneous residues if faults do not affect a residue and its duplicate at the same time; i.e., if at least one of the codeword shown in Table 6.2 is not affected, then the data can be recovered. However, if a residue and its duplicate, let say $x_i$ and $x'_i$ are corrupted, D3R cannot correct them even though the other four residues are fault-free. In general, D3R is able to tolerate up to $t+1 = \frac{n-k}{2} + 1$ residues.

- **C-RRNS**: This code always ensures the correction of up to three erroneous residues regardless of the erroneous residue combination. In general, C-RRNS is able to tolerate up to $t = \frac{n-k}{2}$ residues.
6M-RRNS: This code always ensures the correction of up to two erroneous residues regardless of the erroneous residue combination [35]. In general, 6M-RRNS is able to tolerate up to $t = \frac{n-k}{2}$ residues.

RS: This code always ensures the correction of up to two erroneous symbols regardless of the erroneous symbol combination. In general, RS is able to tolerate up to $t = \frac{n-k}{2}$ symbols.

Moreover, both D3R and 6M-RRNS codes ensure that the ratio of correctable bits over the generated codeword $T_{B_{ECC}}$ is better than that of C-RRNS and RS codes. For this analysis, the ratio is calculated as $T_{B_{ECC}} = \frac{\max(b_r)}{b_s}$ where $b_r$ is the maximum number of correctable bits in a codeword, and $b_s$ is the generated codeword size. For example, for 16-bit input data encoded into D3R code, the maximum number of erroneous bits this code can correct is $b_r = 27$ out of the generated codeword size $b_s = 52$ bits. Based on the set parameter for such memory word size, the ratio of correctable bits over the generated codeword size is $T_{D3R} = \frac{27}{52} = 51.9\%$. Using the same calculation, 6M-RRNS code possesses $T_{6M-RRNS} = \frac{17}{40} = 42.5\%$, RS code possesses $T_{RS} = \frac{16}{48} = 33.3\%$ and C-RRNS code possesses only $T_{C-RRNS} = \frac{19}{61} = 31.1\%$. Although the differences decrease as the input data width becomes larger, they are still significant.

One can conclude that the D3R and 6M-RRNS codes provide correction capability that is as competitive as that of C-RRNS and RS codes at a lower overall codeword size; the latter attribute will be discussed next.

### 6.5.3 Area overhead and useful data capacity

In essence, introducing fault tolerance to the original data comes with a cost of a longer codeword size. This overhead can be analyzed from two perspectives: (i) the memory cell area for a fixed memory word size and (ii) the useful data capacity for a fixed memory area size. Note that useful data capacity can be defined as the number of codewords that represent a set of input data.

The first perspective is analyzed by calculating the codeword size (in terms of the number of bits) generated by the considered ECCs. Figure 6.5 depicts the codeword size of the considered ECCs normalized to C-RRNS. The figure shows that 6M-RRNS code has a shorter codeword size than that of C-RRNS and RS codes for memory word size lower than 256 bits; e.g., for a 64-bit memory, the difference is 13% as compared to C-RRNS and 8% to RS. For very large memory word size (> 512-bit), the difference in the codeword size becomes negligible. The same trend is obtained for D3R when compared to C-RRNS. This analysis indicates that both 6M-RRNS and D3R consume a comparable area overhead as compared to C-RRNS and RS.

The second perspective is analyzed by calculating the number of codewords that can be stored in a 1Tbit per square unit area of memory chip. Note that this memory size can store the entire $1T = 2^{40}$ bits data if the data is not encoded.
6.5. Experimental evaluation

Figure 6.5: Codeword size of the considered ECCs

to any ECC. However, when the data in encoded into the considered ECCs, the useful data becomes $\text{UsefulData}_{\text{ECC}} = \frac{2^{b_s}}{b_s}$ where $b_s$ is the generated codeword size. For example, for 64-bit input data encoded into 6M-RRNS code with codeword size $b_s = 184$ bits, its useful data capacity $\text{UsefulData}_{6\text{M}-\text{RRNS}} = \frac{2^{184}}{184} = 6 \times 10^9$ words. Using the same calculation, D3R code produces $\text{UsefulData}_{\text{D3R}} = \frac{2^{196}}{196} = 5.6 \times 10^9$ words, RS code produces $\text{UsefulData}_{\text{RS}} = \frac{2^{192}}{192} = 5.7 \times 10^9$ words and C-RRNS code produces $\text{UsefulData}_{\text{C-RRNS}} = \frac{2^{205}}{205} = 5.3 \times 10^9$ words. These numbers indicate that 6M-RRNS code allows 4.3% and 11.4% larger useful data capacity than RS and C-RRNS codes, respectively. Similarly, D3R code allows 4.6% bigger and 0.5% less useful data capacity than C-RRNS and RS codes, respectively.

6.5.4 Decoding performance

The symbol-based ECCs considered in this thesis require an iterative correction process during decoding; this process might result in performance penalty to the memory system. Based on the number of erroneous residues/symbols the ECCs intend to correct, the maximum iteration they require is given as follows.

- **D3R**: This code ensures the correction of three residues by performing up to $C_3^1 = \frac{3!}{1!(3-1)!} = 3$ iterations. Note that although D3R code has six residues, the correction process executes in parallel. The iteration stops when the fault has been detected and the correct value is determined. In general, D3R code iterates for a maximum of $C_n^2$ times.

- **C-RRNS**: This code ensures the correction of three residues by perform-
ing up to $C_3^9 = \frac{9!}{3!(9-3)!} = 84$ iterations [103]. As with D3R code, C-RRNS code stops iterate when it has recovered the correct output data. In general, C-RRNS code iterates for a maximum of $C_7^6$ times.

- **RS:** This code ensures the correction of two symbols by performing up to $C_2^6 = \frac{6!}{2!(6-2)!} = 15$ iterations [102]. RS code stops iterate when it has recovered the correct output data. In general, RS code iterates for a maximum of $C_3^5$ times.

- **6M-RRNS:** This code ensures the correction of three residues by performing up to $C_3^6 = \frac{6!}{3!(6-3)!} = 15$ iterations. Moreover, it must go through all its iterations and an additional likelihood decoding step [35]. In general, 6M-RRNS code iterates for a maximum of $C_4^6 + 1$ times.

Therefore, the decoding performance of D3R code is much better than the other three ECCs; i.e., it is $\frac{84}{28} \approx 28 \times$ faster than C-RRNS code, and $5 \times$ faster than RS and 6M-RRNS codes. Furthermore, these differences increase for higher error correction capability.

### 6.6 Summary

This chapter provided two error correction codes, modified from Redundant Residue Number System (RRNS) code, to tolerate faults in the memory cell array of emerging memories. The evaluation shows that D3R code is the best among the conventional RRNS and Reed-Solomon codes as well as another modified version of RRNS code, i.e., 6M-RRNS code. D3R code enables a high-performance decoding together with a competitive fault tolerance and a comparable area overhead. Therefore, this code is selected to introduce a fault-tolerant architecture for the memory system, which will be presented in the next chapter.

The main topics discussed in this chapter are as follows.

- Overview of conventional Redundant Residue Number System (RRNS) code including the coding requirement, encoding procedures and decoding procedures.

- Introduction of Six-Moduli RRNS (6M-RRNS) code, which reduces the codeword size and in turn memory area overhead while maintaining the fault tolerance capability.

- Introduction of Double-Three Residue (D3R) code, which improves the decoding latency and in turn memory performance, while maintaining the fault tolerance capability.

- Comparison between both 6M-RRNS and D3R codes and conventional RRNS and Reed-Solomon codes in terms of fault tolerance capability, area overhead of the memory cell array and decoding performance.
The previous chapter introduced two error correction codes to tolerate faults in the memory cell array. In that chapter, the decoder that operates the error correction codes is assumed to be free from faults; this is the case for most of the published work for emerging memories. Nevertheless, as the decoder is formed using nanometer scaled CMOS devices, it is subject to aging and soft error failures [26, 105, 106, 107]. Aging mechanisms such as negative bias temperature instability [105] and hot carrier injection [106], when stimulated by disturbances such as temperature or elevated voltage, will cause intermittent faults. Moreover, as the signal-to-noise ratio of scaled CMOS reduces, the decoder circuit is no longer resistant to transient faults when struck by cosmic particles [26, 107]. No matter how effective the fault-tolerant scheme applied to the memory cell array is, a fault in the decoder causes the memory to produce an incorrect output data; this in turn impacts the reliability of the entire memory system. Therefore, designing reliable emerging memories requires not only protecting the memory cell array, but the decoder as well.

In this chapter, a fault-tolerant technique – that combines Double Three-Residue (D3R) code (introduced in Chapter 6) and an on-line masking scheme – is proposed to tolerate faults in the memory cell array and decoder. Section 7.1 describes the emerging memory architecture equipped with the basic D3R decoder, which is referred to as D3R-based architecture. Section 7.2 introduces the improved version of the architecture incorporated with the on-line masking scheme and new decoding process; this architecture is referred to as Optimized D3R-based architecture. Section 7.3 provides the evaluation of the two architectures and comparison to a related work. Section 7.4 summarizes the chapter.
Chapter 7. Tolerating Faults in Memory Cell Array and Decoder

7.1 D3R-based architecture

This section describes the D3R-based architecture developed to tolerate faults in the memory cell array while assuming that no faults occur in the decoder. As shown in Figure 7.1(a), D3R-based architecture comprises:

- **CMOS encoder**: This unit encodes the input data into the D3R codewords.
- **CMOS decoder**: This unit decodes the D3R codewords into the output data.
- **Non-CMOS memory cell array**: This unit stores D3R codewords \((x_1, x_2, x_3)\) and their duplicate \((x'_1, x'_2, x'_3)\).

During operation, faults might impact all memory parts such as the encoder, memory cell array and decoder. The faults that occur in the encoder induce glitches that might corrupt the data being encoded producing an erroneous codeword; likewise, faults in the decoder cause erroneous decoded data. Additionally, the faults might also flip the stored codeword in the memory cell array. The D3R-based architecture will correct the erroneously stored codeword by means of D3R code. The error-free data is then read out from the memory.

In the rest of this section, first the circuit of D3R code will be discussed; thereafter, the decoding process and the circuit will be covered. Note that the encoding and decoding process of D3R code has been explained in Section 6.4 of Chapter 6.

7.1.1 Encoding circuit

The bottom-left part of Figure 7.1(a) shows the block diagram of the D3R encoder constructed of three modulo circuits. These three modulo circuits receive a \(d\)-bit input word and generate the corresponding residues simultaneously. \(\text{Modulo}_1\) and \(\text{Modulo}_2\) generate the two-residue dataword \(x_1\) and \(x_2\), respectively, which then form \(DW = x_1, x_2\); while \(\text{Modulo}_3\) generates a single-residue checkword \(CW = x_3\). Thereafter, the original codeword \(C = DW + CW\) is duplicated producing \(C' = DW' + CW'\). Then both \(C\) and \(C'\) are combined creating a D3R codeword. Finally, the D3R codeword with size of \(b_{D3R} = 2 \times \left\lceil \log_2(m_1 - 1) + 1 \right\rceil + \left\lceil \log_2(m_2 - 1) + 1 \right\rceil + \left\lceil \log_2(m_3 - 1) + 1 \right\rceil\) bits (where \(m_1, m_2\) and \(m_3\) are the moduli used) is stored in the memory cell array. The duplicate dataword and checkword parts are simply generated by wires and buffers; see Figure 7.1(a).

Figure 7.1(b) illustrates the functional units of \(\text{Modulo}_1\) circuit that generates the first residue \(x_1\) based on \(\text{Modulus}_1\), \(m_1 = 2^{p_1} - 1\), where \(p_1\) is the bit length of \(m_1\). The \(\text{Modulo}_1\) circuit consists of a buffer, a carry-save adder (CSA) with end-around carry (EAC), a two’s complement adder (TCA) and a multiplexer.
7.1. D3R-based architecture

Figure 7.1: (a) Block diagram of D3R-based architecture (b) functional units of modulo circuit for $m_1 = 2^{p_1} - 1$ (c) functional units of modulo circuit for $m_2 = 2^{p_2} - 1$ and (d) functional units of modulo circuit for $m_3 = 2^{p_3}$
Chapter 7. Tolerating Faults in Memory Cell Array and Decoder

[108]. The buffer splits the input data $Data_{in}$ into a number of intermediate data groups $G_f$, where $f$ is a positive integer. These $G_f$ are added together producing $Sum$; any resulting carry bit from this summation must be added again. Then, the $Sum$ value and $Modulus_1 = 2^{p_1} - 1$ are added in two’s complement way (representing a subtraction of $Modulus_1$ from $Sum$) producing $Sum’$ and its $Sign$ value. If $Sign = 1$, then the multiplexer will take the $Sum$ value as the encoded data $x_1$ (the first residue). Otherwise, if $Sign = 0$, then the multiplexer will take the $Sum’$ value as the encoded data $x_1$.

The $Modulo_2$ circuit consists of functional units that are similar to those of $Modulo_1$ circuit. The difference is that the $Modulo_2$ circuit generates the second residue $x_2$ based on $Modulus_2 = 2^{p_2} - 1$, where $p_2$ is the bit length of $m_2$; see Figure 7.1(c). Figure 7.1(d) depicts the functional units of $Modulo_3$ that generates $x_3$ based on $Modulus_3 = 2^{p_3}$, where $p_3$ is the bit length of $m_3$. The $Modulo_3$ circuit comprises a buffer that takes the $p_3$-bit least significant bits of the input data as the third residue $x_3$. The remaining $d - p_3$ most significant bits of the input data are left unconnected.

7.1.2 Decoding circuit

The bottom-right part of Figure 7.2(a) depicts the block diagram of the D3R-based decoder, which comprises two detectors and a multiplexer with control logic. As illustrated in Figure 7.2(b), each D3R-based detector is formed by an RRNS-to-binary converter and a comparator. The RRNS-to-binary converter converts a D3R codeword (represented by a set of residues) into binary data by using the Mixed-Radix Conversion algorithm. The comparator checks the validity of the D3R codeword by comparing the converted binary data with the predefined operating legitimate range $LR = 2^d - 1$, where $d$ is memory word size. The input signals of the detectors are the D3R codeword stored in the memory cell array. The output signals of both detectors are multiplexed prior to sending them out of the memory; the selection is controlled by the $Correct$ and $Correct’$ signals.

During detection, $Converter$ converts $C = x_1, x_2, x_3$ into a binary word $B$, while $Converter’$ converts $C’ = x_1’, x_2’, x_3’$ into a binary word $B’$. The binary words $B$ and $B’$ are then fit into the comparators; they are compared to the operation legitimate range $LR = 2^d - 1$ (where $d$ is memory word size) to determine the validity of the read codeword. If no errors are detected, then both $B$ and $B’$ will be within the operation legitimate range setting $Correct$ and $Correct’$ signals to high. This in turn sets the $Valid$ signal to high indicating that the output of the multiplexer is a valid read data. However, if faults affect $B$ ($B’$) while $B’$ ($B$) is fault-free, then the $Correct$ ($Correct’$) signal is set to high so that the correct data $B’$ ($B$) is forwarded to $Data_{out}$ signal; note that in this case the $Valid$ signal is set to high indicating that the $Data_{out}$ signal is valid.

When both $B$ and $B’$ are faulty (i.e., beyond the legitimate range), then a correction action is initiated; the converters of Figure 7.2(b) will then convert
7.1. D3R-based architecture

Figure 7.2: (a) D3R-based architecture (b) units inside the decoder (c) units inside the detector
Table 7.1: Multiplicative inverses for D3R-based decoder

<table>
<thead>
<tr>
<th>Memory words, (d)</th>
<th>Moduli</th>
<th>Multiplicative inverses</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>(m_1 = 255) (m_2 = 511)</td>
<td>(g_{12} = 509)</td>
</tr>
<tr>
<td></td>
<td>(m_2 = 255) (m_3 = 512)</td>
<td>(g_{13} = 255)</td>
</tr>
<tr>
<td></td>
<td>(m_3 = 511) (m_3 = 512)</td>
<td>(g_{23} = 511)</td>
</tr>
<tr>
<td>32</td>
<td>(m_1 = 65535) (m_2 = 131071)</td>
<td>(g_{12} = 131069)</td>
</tr>
<tr>
<td></td>
<td>(m_2 = 65535) (m_3 = 131072)</td>
<td>(g_{13} = 65535)</td>
</tr>
<tr>
<td></td>
<td>(m_3 = 131071) (m_3 = 131072)</td>
<td>(g_{23} = 131071)</td>
</tr>
<tr>
<td>64</td>
<td>(m_1 = 4294967295) (m_2 = 8589934591)</td>
<td>(g_{12} = 8589934589)</td>
</tr>
<tr>
<td></td>
<td>(m_2 = 4294967295) (m_3 = 8589934592)</td>
<td>(g_{13} = 4294967295)</td>
</tr>
<tr>
<td></td>
<td>(m_3 = 8589934591) (m_3 = 8589934592)</td>
<td>(g_{23} = 8589934591)</td>
</tr>
</tbody>
</table>

As mentioned in Section 6.4.2, the D3R decoding is based on the Mixed-Radix Conversion (MRC) algorithm. MRC executes a D3R codeword starting from the most significant residue \(x_1\), and ending with the least significant residue \(x_3\). Besides these residues, other parameters are required including the moduli \(m_s\) and modular multiplicative inverses \(g_{(s-u)s}\), where \(i\) and \(u\) are integers; see Section 6.2.4 of Chapter 6. These parameters can be pre-calculated and are given in Table 7.1 for different memory words. For the two mixed radix digit units in the converter \(g_{(s-u)s}\) is the input; see Figure 7.2(c). The \(g_{(s-u)s}\) values increase proportionally to the memory word sizes; for example, \(g_{12} = 2^{2^d+1} - 3\), \(g_{13} = 2^{2^d} - 1\) and \(g_{23} = 2^{2^d+1} - 1\) where \(d\) is the memory word sizes.

Figure 7.2(c) shows the functional units of the converters that execute the MRC algorithm. The converters consist of two mixed-radix units (formed by two’s complement adders (TCA) and multipliers), two multipliers and an adder. Each mixed-radix unit gets input from the corresponding residues, modular multiplicative inverses and/or mixed-radix digit. For example, the first mixed-radix unit gets input from the first residue \(x_1\), second residues \(x_2\) and multiplicative inverse \(g_{12}\) to produce the second mixed-radix digit \(v_2\). The second mixed-radix unit gets input from the first residue \(x_1\), third residue \(x_3\), second mixed-radix digit \(v_2\), and multiplicative inverses \(g_{13}\) and \(g_{23}\) to produce the third mixed-radix digit \(v_3\). Note that the first mixed-radix digit \(v_1\) is equal to the first residue \(x_1\); thus only wires are needed.
7.2 Optimized D3R-based architecture

The D3R-based architecture described in the previous section tolerates both random and clustered faults in the memory cell array. In this section, the architecture will be extended so that it can tolerate faults in the decoder as well [38]. According to [109], the combinational logic nodes closer to the output pins have more impact to transient faults than other circuit parts. Inspecting Figure 7.3(a) reveals that the decoder/multiplexer is the closest unit to the output pins. Therefore, making this unit tolerant from faults will improve the overall reliability of the memory architecture. The bottom-right part of Figure 7.3(a) illustrates the block diagram of the Optimized D3R-based decoder comprising two modified detectors and a group of Muller C-gates; the latter is used as an on-line masking scheme. The circuit and the process of this decoder are optimized to realize smaller area overhead. In the rest of this section, first the optimized decoding circuit will be discussed; then, the optimized decoding process and finally, the on-line masking circuit formed using Muller C-gates.

7.2.1 Optimized decoding circuit

The decoding process is based on: (a) the multiplication of residues $x_s$ and multiplicative inverses $g_{(s-u)s}$ where $s$ and $u$ are integers, (b) the subtraction, and (c) the summation; see Section 6.4. Making the $g_{(s-u)s}$ values smaller reduces the multiplication complexity. Moreover, there is no need for multiplication if the $g_{(s-u)s}$ values are made equal to 1, and there are only shift operations if the $g_{(s-u)s}$ values are power of 2. These are the basic ideas used to develop the new decoder. The new modular multiplicative inverses $g_{(s-u)s}^{(new)}$ will be referred to as $g_{(s-u)s}^{(new)}$.

In order to explain how the $g_{(s-u)s}^{(new)}$ values are generated, let us assume the moduli set $m_s$ for 16-bit memory word (see Table 7.1 for $d=16$); herein, $m_s = \{255, 511, 512\}$ where 255 is the most significant moduli (MSM) and 512 is the least significant moduli (LSM). In the new design, however, these moduli are put in the opposite sequence where 512 is the MSM and 255 is the LSM; hence $m_s^{(new)} = \{512, 511, 255\}$. With these values, the new multiplicative inverses will be calculated using the equation $|m_{(s-u)s}^{(new)} \times g_{(s-u)s}^{(new)}|_{m_{(s-u)s}^{(new)}=1} = 1$ [98]. Note that this calculation is based on a trial and error approach [98], i.e., increasing the $g_{(s-u)s}^{(new)}$ value by one in each attempt. Substituting $m_1^{(new)} = 512$, $m_2^{(new)} = 511$ and $g_{12}^{(new)} = 1$ into the equation produce $|512 \times 1|_{511} = 1$, meaning that $g_{12}^{(new)}=1$ is the desired value. This is also the case for $m_2^{(new)} = 511$ and $m_3^{(new)} = 255$, where they produce $g_{23}^{(new)} = 1$. However, substituting $m_1^{(new)}=512$ and $m_3^{(new)}=255$ with $g_{13}^{(new)} = 1$ produce $|512 \times 1|_{255} = 2$; in this case $g_{13}^{(new)}=1$ does not satisfy the RRNS uniqueness requirement [98]. Yet, after several attempts the desired multiplicative inverse is obtained, i.e., $g_{13}^{(new)} = 128$, which is the power of 2.

It is worth noting that the optimized decoding process realizes two new
Figure 7.3: (a) Optimized D3R(OD3R)-based architecture (b) units inside the decoder (c) units inside the detector
Table 7.2: Multiplicative inverses for Optimized D3R-based decoder

<table>
<thead>
<tr>
<th>Memory word, $d$</th>
<th>Moduli</th>
<th>Multiplicative inverses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m_1(new) = 512$</td>
<td>$m_2(new) = 511$</td>
</tr>
<tr>
<td></td>
<td>$m_1(new) = 512$</td>
<td>$m_3(new) = 255$</td>
</tr>
<tr>
<td></td>
<td>$m_2(new) = 511$</td>
<td>$m_3(new) = 255$</td>
</tr>
<tr>
<td>16</td>
<td>$m_1(new) = 131072$</td>
<td>$m_2(new) = 131071$</td>
</tr>
<tr>
<td></td>
<td>$m_1(new) = 131072$</td>
<td>$m_3(new) = 55535$</td>
</tr>
<tr>
<td></td>
<td>$m_2(new) = 131071$</td>
<td>$m_3(new) = 55535$</td>
</tr>
<tr>
<td>32</td>
<td>$m_1(new) = 8589934592$</td>
<td>$m_2(new) = 8589934591$</td>
</tr>
<tr>
<td></td>
<td>$m_1(new) = 8589934592$</td>
<td>$m_3(new) = 4294967295$</td>
</tr>
<tr>
<td></td>
<td>$m_2(new) = 8589934591$</td>
<td>$m_3(new) = 4294967295$</td>
</tr>
</tbody>
</table>

Modular multiplicative inverses $g_{12(new)} = g_{23(new)} = 1$ regardless of the memory word size, and $g_{13(new)} = 2^{d-1}$ where $d$ is the memory word size. This significantly reduces the decoding implementation as compared to the conventional design. Table 7.2 gives the new modular multiplicative inverses for different memory word sizes. Figure 7.3(c) shows that the top-right mixed-radix unit consists of only a two’s complement adder (without multiplier since $g_{12(new)} = 1$) and the other unit comprises a shifter (instead of multiplier since $g_{13(new)}$ is power of 2 and $g_{23(new)} = 1$) and two’s complement adders. Thus, the Optimized D3R-based decoder requires a simpler circuit than the D3R-based decoder; see Figure 7.2(c).

7.2.2 Optimized decoding process

In addition to simplifying the decoding circuit, the decoding process has been optimized in order to improve the overall memory reliability. The concept of time redundancy is used in this new decoding process; see Section 3.5 of Chapter 3. In contrast to only up to three iterations to recover the read data if faults are detected in D3R-based architecture (see Figure 7.2(a)), the Optimized D3R-based architecture will continue up to six iterations. Each Optimized D3R-based detector will execute the RRNS-to-binary conversion for all six residue sets shown in Table 6.2 of Chapter 6. Note that the execution of a next iteration is needed only if the obtained converted data is larger than the legitimate range. It is clear that the new decoder requires up to twice as many iterations as the former decoder, which in turn can impact performance. In order to compensate for this performance loss, the decoding process is modified and made cost-efficient using a reverse decoding process.
Chapter 7. Tolerating Faults in Memory Cell Array and Decoder

As mentioned before, the new decoding circuit reverses the MSM and LSM of D3R-based architecture. Therefore, the decoding process has to be reversed as well to satisfy the MRC algorithm used in the decoding process [98]. For each D3R codeword part (see the residue sets in Table 6.2 of Chapter 6), the conversion to binary starts at the residue farthest to the right followed by the next residue to the left. For example, for the original codeword part C the conversion starts first at $x_3$, then $x_2$ and finally $x_1$. This is exactly the reverse order of the operation performed by the decoder of D3R-based architecture, which execute, e.g., first $x_1$, then $x_2$ and finally $x_3$. In order to provide more insight into the modified decoding procedure, an example of encoding and decoding for 16-bit input data is given in Appendices A4 and A5.

7.2.3 On-line masking circuit

The C-gates are used to replace the multiplexer in D3R-based architecture, as shown in Figure 7.3(a) and (b). The inputs of the C-gates are connected to the outputs of the two detectors. The reason for using C-gates instead of the multiplexer is that these asynchronous logic gates are able to mask short-period glitches (due to intermittent and transient faults) produced by the detectors. In practice, the probability of two glitches for occurring simultaneously at two pins carrying the same signal is very low. For example, the probability that glitches occur simultaneously at one output of a 64-bit Detector and one output of a 64-bit Detector’ is $(\frac{1}{64})^2 = 2.44 \times 10^{-4}$. This is where C-gates show their superiority because even if there are many short-period glitches, as long as they occur at different times and/or at different pins, the output data is still unchanged.

7.3 Experimental evaluation

This section gives the evaluation of the Optimized D3R-based architecture. First, a comparison to the D3R-based architecture in terms of the area overhead, performance penalty and system reliability. Thereafter, a comparison to the related work will be given in terms of correctable bit over codeword size ratio and codeword size.

7.3.1 Experimental setup

As mentioned before, the D3R-based architecture is based on the conventional MRC algorithm and the decoding parameters given in Table 7.1; see also Figure 7.2 for the block diagram. In contrast, the Optimized D3R-based decoder is based on the modified MRC algorithm and the decoding parameters presented in Table 7.2; see also Figure 7.3 for the block diagram. The implementation of these decoders was done using VHDL on Xilinx ISE and Synopsys Design Compiler tools based on 90nm CMOS technology.
The reliability evaluation was carried out using Matlab simulation. Faults were randomly injected into the memory cell array with fault rates ranging from 1% to 10%. The faults flip a number of adjacent bits (representing the clustered errors) that form the codewords of the considered ECCs. Besides injecting faults into the memory cell array, the faults were also injected into the decoders with the ratio of 1:10 (decoder:memory cell array). This ratio is set based on the soft error rates between SRAM bit and logic for 90nm CMOS technology reported in [26].

### 7.3.2 Area overhead

Figure 7.4 illustrates the area overhead for both proposed decoders normalized to the D3R-based decoder. The figure clearly shows that the Optimized D3R-based decoder requires smaller area overhead than is required by the D3R-based decoder regardless of the memory word sizes; the reduction becomes significant for larger memory words. This benefit is achieved because no multiplier is needed in one mixed-radix unit, and because a shift register is used instead of a multiplier in another mixed radix unit; see Figure 7.3(c). For example, for a word size of 16 bits, the Optimized D3R-based decoder consumes 10% less area than the D3R-based decoder. For a word size of 128 bits, the difference becomes 25%; hence, it is expected that this benefit will be more noticeable for larger memory word size.
Chapter 7. Tolerating Faults in Memory Cell Array and Decoder

7.3.3 Performance penalty

Figure 7.5 depicts the performance penalty for both proposed decoders normalized to the D3R-based decoder. The figure indicates that for smaller word size (≤ 64 bits), the performance of the Optimized D3R-based decoder is slightly slower than the D3R-based decoder (≤ 5%). However, as the word size increases, the performance becomes close to and eventually faster than that of the D3R-based decoder. It is anticipated that for large memory word, the Optimized D3R-based decoder will be even faster than the D3R-based decoder. This is mainly because the D3R-based architecture multiplier (a unit in the D3R-based architecture decoder) requires longer time to execute large memory word than that of Optimized D3R-based architecture shifter (unit in the Optimized D3R-based architecture decoder).

7.3.4 Memory system reliability

Figure 7.6 shows the simulation results of memory system reliability for the two proposed architectures for a 64-bit memory word. Clearly, the Optimized D3R-based decoder provides better system reliability than the D3R-based decoder irrespective of the fault rate. The difference becomes larger at higher fault rate, e.g., 10× greater at 10% fault rate as compared to 1% fault rate. Therefore, the Optimized D3R-based decoder is able to improve the memory system reliability especially at higher fault rates.
7.3. Experimental evaluation

7.3.5 Comparison to related work

To the best knowledge of the authors, no published work has addressed the same problem for emerging nanoelectronic memories as considered in this thesis except H. Naeimi et al. [83]. However, the error correction scheme used in [83] targets only random errors using Euclidean Geometry Low-Density Parity-Check (EG-LDPC) code. Contrarily, the ECC proposed in this work targets both random and clustered errors using Double Three-Residue (D3R). Despite this difference, the results from [83] will be used to compare the following attributes: correctable bits over codeword size ratio and area overhead of the memory cell array.

Table 7.3 summarizes the parameters used to develop both EG-LDPC and D3R codes. The parameters \( n, k, t \) denote the codeword size, the dataword (memory word) size and the error correction capability, respectively. Note that the memory sizes for D3R are chosen to be the nearest to the ones reported in [83] in order to perform a fair comparison.

The third column of Table 7.3 gives the ratio of the corrected bits in each generated codeword \( T_{\text{ECC}} \) of the two ECCs. For example, for 32-bit memory, D3R can correct up to \( T_{\text{D3R}} = \frac{n}{t} = \frac{50}{100} = 50\% \) of bits that form the six-residue codeword, while EG-LDPC is reported to correct only \( T_{\text{EG-LDPC}} = \frac{n}{t} = \frac{4}{63} = 6\% \) [83]; this results shows than D3R is \( 8.3 \times \) better than EG-LDPC. The improvement is even more significant for larger memory words.

The fourth column of Table 7.3 presents the ratio of the codeword size of
Table 7.3: Comparison of ECC proposed in this thesis and in [83]

<table>
<thead>
<tr>
<th>D3R</th>
<th>EG-LDPC [83]</th>
<th>Differences in Correctable bits over codeword size ratio</th>
<th>Required codeword size</th>
</tr>
</thead>
<tbody>
<tr>
<td>((n_1, k_1, t_1))</td>
<td>((n_2, k_2, t_2))</td>
<td>((\frac{t_1}{n_1}, \frac{t_2}{n_2}))</td>
<td>(m_1/\hat{m}_1)</td>
</tr>
<tr>
<td>100,32,50</td>
<td>63,37,4</td>
<td>((\frac{50}{100}, \frac{4}{63}))</td>
<td>8.3</td>
</tr>
<tr>
<td>388,128,194</td>
<td>255,175,8</td>
<td>((\frac{194}{388}, \frac{8}{255}))</td>
<td>16.7</td>
</tr>
</tbody>
</table>

D3R and EG-LDPC; it clearly shows that the improvement in terms of error correction capability of D3R comes at the cost of 50% larger codeword size as compared to EG-LDPC. This means that D3R requires bigger memory cell area overhead than that of EG-LDPC.

7.4 Summary

This chapter presented a fault-tolerant memory architecture to tolerate fault in the memory cell array and decoder of an emerging memory. The fault-tolerant memory architecture combines the D3R code introduced in Chapter 6 and asynchronous gates. In addition, the D3R decoding process is reversed for design optimization. The evaluation shows that the fault-tolerant memory architecture provides better memory system reliability at lower decoder overhead as compared to the basic memory architecture incorporated only with the D3R code and the typical decoding process.

The main topics discussed are as follows.

■ Description of the emerging memory architecture that addresses faults only in memory cell array while assuming the decoder is free from faults.

■ Introduction of an optimized fault-tolerant architecture that addresses faults both in the memory cell array and the decoder.

■ Adaptation of a conventional design technique with one of the proposed error correction techniques to address faults in the memory cell array and decoder.

■ Establishment of the optimized decoding procedure that realizes smaller decoder area overhead.

■ Comparison between the two architectures and a related work.
CHAPTER 8

TOLERATING FAULTS IN MEMORY CELL ARRAY AND VIAS

8.1 CMOS to Non-CMOS Vias
8.2 Fault-tolerant architectures
8.3 Experimental evaluation
8.4 Summary

A new approach that increases the performance and density of existing integrated circuits involves stacking multiple circuit layers vertically; this realizes a three-dimensional circuit. The circuits stacked at different layers are connected using special interface pins referred to as Through Silicon Vias. Likewise, three-dimensional (3D) emerging memories comprising CMOS peripheral circuit at the bottom and non-CMOS memory cell array at the top are realized by using sharp tip vertical interface pins; these pins are referred to as CMOS to Non-CMOS Vias (CNVs) [46, 47]. Because these CNVs are very tiny and are placed close to each other, it is difficult to fabricate them without any defect; furthermore, they also subject to coupling and interference. When this happens, the connection between the peripheral circuits and the memory cell array will be affected; in the worst case these two memory parts are completely disconnected where data cannot be stored and retrieved. Therefore, these CNVs have a large influence on the quality and reliability of the memory.

In this chapter, a fault-tolerant technique – that combines RRNS code and interleaving technique – is devised to address faults in the memory cell array and CNVs. Section 8.1 describes the CNVs concept in the three-dimensional emerging memory and their defect cases. Then, Section 8.2 presents two fault-tolerant architectures: (i) the conventional architecture that addresses faults in the memory cell array, and (ii) the improved architecture that addresses faults in both memory parts. Thereafter, Section 8.3 gives the simulation results of the two fault-tolerant architectures and their hardware implementation. Finally, Section 8.4 summarizes the chapter.
Chapter 8. Tolerating Faults in Memory Cell Array and Vias

8.1 CMOS to Non-CMOS Vias

This section describes the concept of CMOS to Non-CMOS Vias used in the 3D emerging memory, and the defect scenarios that are considered in the study case.

8.1.1 CMOS to Non-CMOS Vias concept

As mentioned before, the memory cell array of 3D emerging memories is stacked above the peripheral circuits. The memory cell array consists of two sets of nanowires crossing in perpendicular and molecular devices embedded at each of the nanowire crosspoints. This structure is commonly referred to as a crossbar memory cell array. Figure 8.1 depicts a portion of the 3D emerging memory where the crossbar memory cell array at the top is connected to the peripheral circuits at the bottom through two groups of CNVs [3, 4, 110].

In order to increase the density, two design approaches are introduced in the memory: (i) the memory cell array is rotated by $H^\circ$ with respect to the peripheral circuits, and (ii) the CNVs are equally distributed across the peripheral circuits. In terms of hardware structure, these approaches ensure that each nanowire is connected to exactly one CNV; thus, they enable individual access to a single memory cell. A pair of CNVs, each with a different height, is fabricated at each memory relay cell. The tall (red-color) CNVs connect the peripheral circuits to nearly-vertical nanowires (top nanowires), whereas the short (blue-color) CNVs to nearly horizontal nanowires (bottom nanowires). A detailed explanation of this memory structure and operation can be referred to [3, 4, 110].

Figure 8.2(a) and (b) illustrate the connection of a group of memory cells $C_1$.

![Figure 8.1: CMOS to Non-CMOS Vias distributions in a 3D emerging memory [110]](image)
Figure 8.2: A single memory word in 3D emerging memory: (a) side view schematic (b) top view schematic (c) electrical equivalent circuit
to $C_n$, which represent a single memory word in the 3D emerging memory from side and top views, respectively. These memory cells are sandwiched between a single top nanowire $NW_{T1}$ and $n$ numbers of bottom nanowires $NW_{B1}$ to $NW_{Bn}$ where $n$ is an integer. In order to access the memory cells from the peripheral circuits through CMOS lines $CB_{T1}$, $CB_{S1}$ to $CB_{Sn}$, one tall CNV $V_{L1}$ and a number of short CNVs $V_{S1}$ to $V_{Sn}$ are used. For example, to access $C_1$, signals at $CB_{S1}$ and $CB_{T1}$ are activated making a complete path from $V_{L1}$ and $NW_{T1}$, crossing $C_1$ toward $NW_{S1}$ and $V_{B1}$.

Figure 8.2(c) depicts the electrical equivalent circuit of the single memory word. The nanowire crossbars (top and bottom nanowires) hold a group of molecular non-CMOS nanodevices (symbolized as resistive memory cells) representing a single memory word. The nanowires are connected to their corresponding CMOS lines through the respective CNVs.

### 8.1.2 Case study: Defective CMOS to Non-CMOS Vias

As mentioned in Chapter 4, defects in CNVs are due to several failure mechanisms as follows [48, 110, 111].

- Variations in CNVs height that cause some nanowires to be unconnected introducing open defects. Because half of the CNVs must be taller than the other half, complicated fabrication steps are required, which introduce more possible defects.

- Misalignment of CNVs that cause nanowires to be unconnected introducing open defects; this problem also results in one CNV to be connected to more than one nanowires causing short defects.

- Process variabilities such as excessive polishing, overdone etching and imprecise via formation; these process variabilities induce defects such as cracks, pinholes and voids within the CNVs causing delay faults.

- Electromigration due to high current density and high temperature for metal-based CNVs; this also causes delay faults.

In the worst scenarios, there are three possible defective CNV cases as follows [39].

- If one short CNV is defective, then the corresponding memory cell cannot be accessed. For example in Figure 8.2, if the most left short CNV $V_{S1}$ is broken, memory cell $C_1$ cannot be accessed.

- If more than one short CNV is defective, then the corresponding memory cells cannot be accessed. For example in Figure 8.2, if the two most left short CNVs $V_{S1}$ and $V_{S2}$ are broken, memory cells $C_1$ and $C_2$ cannot be accessed.
If one long CNV is defective, then the entire memory cell connected by the bottom nanowire will be affected. For example in Figure 8.2, if $V_{L1}$ is broken, memory cells $C_1, C_2, ..., C_{n-1}$ and $C_n$ cannot be accessed. Thus, the long CNVs are more critical than the short CNVs.

### 8.2 Fault-tolerant architectures

This section describes two fault-tolerant architectures: (i) the typical architecture that tolerates faults in the memory cell array, and (ii) the modified architecture that tolerates faults both in the memory cell array and in the vias.

#### 8.2.1 RRNS-based architecture

Figure 8.3 illustrates the architecture of the typical 3D emerging memory that stores RRNS codewords; this architecture is referred to as RRNS-based architecture [3]. For simplicity of illustration and analysis, only four memory banks that store four RRNS codewords are shown. These four RRNS codewords, i.e., $A_s, B_s, C_s$, and $D_s$ where $1 \leq s \leq 4$, are accessed simultaneously by asserting the appropriate address. Each codeword consists of two non-redundant residues (i.e., $k = 2$) and two redundant residues (i.e., $(n-k) = 2$); hence, the RRNS code is capable of correcting a single erroneous residue (i.e., $t = \frac{n-k}{2} = 1$) [103]. The codewords are written through the RRNS encoder and read through the RRNS decoder. The encoder and decoder (part of the peripheral circuits) are connected to the memory cell array through the CNVs.

The shortcoming of the RRNS-based architecture is its low error correction

Figure 8.3: RRNS-based architecture
capability for faults caused by defective CNVs. For example, let us assume that faults occur in two short CNVs carrying residues $A_1$ and $A_2$. In this case, the value of the read residues will be incorrect and in turn produces an erroneous output data. Since the RRNS code in this example can only correct one residue, the correct codeword cannot be recovered. The situation becomes worse if a long CNV is defective where the entire codeword $A_i$ is in erroneous state.

### 8.2.2 Interleaved RRNS-based architecture

Figure 8.4 shows the proposed fault-tolerant architecture for 3D emerging memory that stores interleaved RRNS codewords; this architecture is referred to as *Interleaved RRNS-based architecture*. For simplicity of illustration and analysis, only the interleaved connections for the first codeword, i.e., $A_1$ are shown. Interleaver and de-interleaver units are added after the encoder and before the decoder, respectively. This implies that the encoded data (RRNS codewords) are interleaved prior to being stored in the memory banks. Similarly, the RRNS codewords (stored data) are de-interleaved before decoding them.

Because the size of each bank is fixed and is equal to each other, the interleaved residues must fit within the word size of the bank. Moreover, each RRNS residue might have different size from the others depending on the modulus used [103]. Therefore, in order to use the same bank size as in the RRNS-based ar-

![Interleaved RRNS-based architecture](image)

Figure 8.4: Interleaved RRNS-based architecture
architecture while providing an improved fault tolerance, the interleaved pattern shown in Figure 8.4 is chosen. Only the first residues of the four codewords are stored in their original memory banks, while the other three residues are distributed into different banks. For example, \( A_1 \) is stored in Bank A, \( A_2 \) in Bank D, \( A_3 \) in Bank C, \( A_4 \) in Bank B. Although the codeword size of the interleaved RRNS is \( B_{\text{IRRNS}} = \log_2 A_s + \log_2 B_s + \log_2 C_s + \log_2 D_s \), this size is equal to \( B_{\text{BankA}} \) since \( \log_2 A_2 = \log_2 B_2, \log_2 A_3 = \log_2 C_3 \) and \( \log_2 A_4 = \log_2 D_4 \).

The interleaved residues can be represented in matrix form as follows. Without interleaving, the stored codewords are in the form of matrix \( \text{RRNS} \). On the other hand, by using the proposed interleaving pattern, the stored codewords are in the form of \( \text{IRRNS} \). Note that each row represents each memory bank that stores the respective four RRNS residues. For example, the residues in the first row of \( \text{IRRNS} \) are the interleaved RRNS codewords stored in Bank A.

\[
\text{RRNS} = \begin{pmatrix}
A_1 & A_2 & A_3 & A_4 \\
B_1 & B_2 & B_3 & B_4 \\
C_1 & C_2 & C_3 & C_4 \\
D_1 & D_2 & D_3 & D_4
\end{pmatrix}
\]

\[
\text{IRRNS} = \begin{pmatrix}
A_1 & B_2 & C_3 & D_4 \\
B_1 & C_2 & D_3 & A_4 \\
C_1 & D_2 & A_3 & B_4 \\
D_1 & A_2 & B_3 & C_4
\end{pmatrix}
\]

With the interleaving scheme, up to four erroneous residues can be corrected if they occur in a single \( \text{IRRNS} \) codeword while the other three \( \text{IRRNS} \) codewords are error-free. For example, let us assume that the entire first interleaved codeword \( \text{IRRNS}_1 \) (i.e., \( A_1, B_2, C_3, D_4 \)) is corrupted. Before decoding, the \( \text{IRRNS}_1 \) residues are de-interleaved into \( \text{RRNS} \) organization. This de-interleaving process enables the four erroneous residues that formed \( \text{IRRNS}_1 \) codeword becoming a single erroneous residue in each \( \text{RRNS} \) codeword; for example, only \( A_1 \) is erroneous in \( \text{RRNS}_1 \) (Bank A). Therefore, they are still within the error correction capability of the considered RRNS code.

The proposed architecture has the advantage of being able to improve the error correction capability even for clustered faults, which are expected to be more prevalent for nanoscale circuits such as emerging memories [24, 25]. For example, if one or more short CNVs in Bank A are defective, then the proposed architecture will still be able to provide the correct read data, as explained above. This is because each defective short CNV can impact at most one residue of a codeword, which is still within the error correction capability of the considered RRNS code. Even if one of the long CNV in Bank A is defective, the architecture will still recover the correct data. Again, this is because in the proposed architecture, only one residue is shared per memory bank.
In addition to clustered faults, the proposed architecture can also mitigate random faults as long as the defective CNVs do not impact two residues belonging to the same codeword. For example in Figure 8.4, if the defective CNVs impact residues \(A_1\) in \(BankA\) and residue \(C_1\) in \(BankB\), the read codeword \(A_s\) and \(C_s\) where \(1 \leq s \leq 4\) are still correctable. This is because each codeword has only one erroneous residue, which is within the correction capability of considered RRNS.

8.3 Experimental evaluation

This section presents the evaluation of Interleaved RRNS architecture. A comparison to RRNS architecture is performed in terms of fault tolerance capability, area overhead and performance penalty.

8.3.1 Experimental setup

All designs including the encoder and decoder of the RRNS code, interleaver, de-interleaver, memories and fault injection were described using Matlab script. The RRNS code is based on the moduli set \(\{2^{p+1} - 1, 2^{p+1}, 2^{p+1} + 1, 2^{p+2} + 1\}\) where \(p = 32\) for 64-bit memory word [32, 33]. In the experiment, defective CNVs were assumed to be completely open. Hence, the cells associated with the defective CNVs were assumed to be faulty. Faults were randomly injected with the fault rate of up to 10% of the memory. The experiment is performed for two cases:

- **Case 1**: Defective short CNVs leading to the bottom layer of nanowires.
- **Case 2**: Defective tall CNVs leading to the top layer of nanowires.

In order to estimate the area and speed of the encoder and decoder of both RRNS and IRRNS architectures, the circuits were designed and synthesized at 90\(\text{nm}\) CMOS technology using Xilinx and Synopsys design tools. The synthesized circuits are further estimated based on 32\(\text{nm}\) CMOS technology [81].

8.3.2 Fault tolerance capability

Figure 8.5 shows the simulation results of the fault tolerance capability of RRNS-based and IRRNS-based architectures for the two cases mentioned above. Overall, IRRNS-based architecture outperforms RRNS-based architectures for both defective CNV cases. Specifically for the defective short CNVs (i.e., Case 1), IRRNS-based architecture ensures 100% fault tolerance capability for all fault rates. However, this is not the case for RRNS-based architectures; their fault tolerance capability decreases linearly with the fault rate approaching 90% at 10%
8.3. Experimental evaluation

For defective long CNVs (i.e., Case 2), IRRNS-based architecture is still better than RRNS-based architectures. Interestingly, the difference between the two architectures becomes more significant as the fault rate increases; e.g., IRRNS-based architecture performs about 1% better than RRNS-based architectures at 5% fault rate, while it is about 2.5% at 10% fault rate.

For both cases, IRRNS provides better fault tolerance capability as compared to RRNS for both cases because:

- **Case 1**: Defective short CNVs may impact single, double, triple or even all four residues that form IRRNS and RRNS codewords. During decoding the erroneous IRRNS residues belong to different memory word; hence, they can be corrected by the IRRNS decoder. However, the erroneous RRNS residues belong to the same memory word; hence the decoder fails to correct when there are two or more erroneous RRNS residues as they are beyond the error correction capability of the RRNS code.

- **Case 2**: Defective tall CNVs may impact all four residues of IRRNS codewords; however, the residues belong to different memory words. As in the first case, the IRRNS decoder can correct these erroneous residues, but the RRNS decoder cannot.
Table 8.1: Overhead for 64-bit encoder/decoder for both RRNS and IRRNS

<table>
<thead>
<tr>
<th>CMOS Technology (Technology)</th>
<th>Circuit</th>
<th>Area ($\mu m^2$)</th>
<th>Speed (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>90nm (Synthesized)</td>
<td>Encoder</td>
<td>3172.38</td>
<td>1.61</td>
</tr>
<tr>
<td></td>
<td>Decoder</td>
<td>20527.32</td>
<td>4.93</td>
</tr>
<tr>
<td>32nm (Estimated)</td>
<td>Encoder</td>
<td>198.27</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>Decoder</td>
<td>1282.96</td>
<td>0.49</td>
</tr>
</tbody>
</table>

8.3.3 Area overhead and performance penalty

Similar results are obtained for both RRNS and IRRNS as shown in Table 8.1. The table shows that the total area overhead of the encoder and decoder is much less than 1% of 1cm² of an emerging memory [54]. In addition, their speed realized by both encoder and decoder is faster than the typical access time of the emerging memory, which can be 30ns [54]. It is worth noting that although an interconnection delay might be incurred in IRRNS architecture, the delay is insignificant and is therefore ignored.

8.4 Summary

This chapter provided a modified architecture that tolerates faults in CMOS to Non-CMOS Vias of a 3D emerging memory. Simulation results show that the modified architecture improves the fault tolerance capability as compared to the conventional architecture at the same area overhead and performance penalty. The main topics discussed are as follows.

- Overview of the structure and defect cases of CMOS to Non-CMOS Vias in a 3D emerging memory. These advanced interfaces allow the memory cell array to be fabricated above the peripheral circuits.
- Description of the conventional 3D memory incorporated with error correction code to tolerate fault in the memory cell array. The architecture has a direct interface between the memory cell array, CMOS to Non-CMOS Vias and peripheral circuits.
- Introduction of an improved 3D emerging memory incorporated with a combined scheme of an error correction code and interleaving that tolerates faults in the memory cell array and in CMOS to Non-CMOS Vias. The architecture has an interleaved interface between the memory cell array, CMOS to Non-CMOS Vias and peripheral circuits.
- Evaluation of the two memory architectures in terms of their fault tolerance capability and hardware implementation.
CHAPTER 9

CONCLUSIONS AND RECOMMENDATIONS

9.1 Conclusion of the chapters
9.2 Thesis contributions
9.3 Recommendations for future research

In order that high quality emerging memories are shipped-out from the manufacturing plants, memory testing is indispensable. Furthermore, in order to ensure that they can operate reliably during their use, fault-tolerant schemes should be employed. This thesis studied these two crucial topics encompassing defect analysis, fault modeling and testing; and schemes that address the quality and reliability-related problems.

In this chapter, the overall accomplishment of the study presented in this thesis is summarized. Section 9.1 concludes the chapters of the thesis. Section 9.2 discusses the contribution of this thesis to scientific knowledge. Finally, Section 9.3 lists the recommendations for future research of this topic.

9.1 Conclusion of the chapters

The conclusion of the chapters are given as follows.

Chapter 1 – Introduction – presents the concept of memory devices and the motivations of the memory technology considered in this thesis. The chapter begins with the concept of a memory device, its operation and its generic schematic diagram with the associate interfaces. Then, a brief description of several memory technologies is given, which are classified into two distinct groups: conventional memories and emerging memories. Conventional memories are devices such as SRAM, DRAM and flash; these memories are widely used in the current computer and other electronic systems. In contrast, emerging memories are such as FRAM, MRAM, STTRAM, PCRAM, RRAM and ORAM; these memories are expected to replace the conventional memories in the future. The
description of the memories includes their electrical structure, their operation, as well as their advantages and disadvantages. The chapter moves on to identify RRAM as the focus memory technology in the thesis. A number of potentials and challenges of RRAM is discussed. Thereafter, the chapter lists the outcomes of this study published in international conferences and journals to the scientific community. Finally, the organization of the thesis is discussed.

Chapter 2 – Emerging Memory Architecture – discusses the RRAM model using a hierarchical modeling approach commonly employed to evaluate complex electronic systems. The chapter begins with the definition of the modeling approach divided into five levels. Thereafter, a detailed description of each modeling level is given where each level concerns only to certain relevant information of the memory. Three of the five modeling levels are used in the evaluation of the work of this thesis. The electrical model, which illustrates the electrical components and connections, is used to model the RRAM for fault analysis. The behavioral model, which treats the memory as a single black box, is used to evaluate the fault tolerance capability for an emerging memory. The functional model, which describes the internal blocks that compose the behavioral model, is used to evaluate the hardware implementation of the proposed fault-tolerant schemes.

Chapter 3 – Testing and Fault Tolerance Concept – gives an overview of different types of memory testing and fault tolerance schemes. The chapter begins with the definition of key terminologies associated with testing and fault tolerance. It also discusses the importance of testing and fault tolerance during the manufacturing product flow. A classification of the memory testing types is given based on the objectives for quality or reliability improvement, together with a brief description of each type. Then, several Design-for-Testability schemes that support memory tests for SRAM stability faults are also discussed. Thereafter, a classification of the fault tolerance schemes is provided based on either redundancy or special design approach, along with a brief description of each scheme. Because some of the testing types and fault tolerance schemes are adopted in the study, this chapter establishes the required basic knowledge for understanding the contribution of this thesis.

Chapter 4 – Defect-Oriented Test for RRAM – establishes a framework of defect-oriented testing for RRAM based on electrical simulation. The chapter begins with the classification and definition of the possible defects that will occur in RRAM. Because the peripheral circuits (e.g., decoder, encoder, sense amplifier, etc.) of RRAM are based on CMOS devices, the same defects that occur in conventional memories (e.g., SRAM, DRAM and flash) are also expected; therefore, the defect classification and definition for RRAM only considers its memory cell array and CMOS to Non-CMOS Vias. For the purpose of realistic defect injection and simulation, an electrical RRAM model is introduced. The evaluation and analysis of open defects that impact a single RRAM cell using the electrical RRAM model is provided. From the analysis, two new unique fault models are discovered that cause the RRAM cell to enter an undefined
logic state. Such logic states correspond to a random logic value, either logic 1 or 0, which is difficult to detect during manufacturing tests.

**Chapter 5 – Design-for-Testability Schemes for RRAM** – proposes two Design-for-Testability (DfT) schemes to solve the problems induced by the two new unique faults analyzed in Chapter 4. By testing the defective RRAM at a lower voltage or shorter time, instead of at a nominal voltage or time as in common manufacturing tests, the proposed DfT schemes are able to capture the two new unique faults. The chapter begins with the concept of DfT schemes that are based on the access time and supply voltage of the RRAM cell. Then, the methodology used in designing two DfT schemes are explained. Thereafter, the experimental evaluation of the proposed schemes and their circuit are also discussed. In addition, the proposed DfT schemes are also extended to have their programmable versions for use in the post-silicon stage. The concept, design methodology and circuits of the programmable DfT schemes are also presented.

**Chapter 6 – Tolerating Faults in Memory Cell Array** – introduces two error correction schemes to tolerate faults in the memory cell array of emerging memories. The error correction schemes adopt, modify and combine several established fault-tolerant schemes. The chapter begins with a description of the architecture of an emerging memory incorporated with the error correction scheme circuit. Next, it explains the conventional RRNS error correction codes (ECC) used in the architecture. Thereafter, each of the error correction scheme modified from RRNS code is introduced along with its concept, structure, and its encoding and decoding process. Then, the modified error correction schemes are evaluated by comparing them to conventional, well-known ECCs such as Reed-Solomon and RRNS. The evaluation shows that in addition to providing a competitive fault tolerance (thus reliability), the first proposed error correction scheme incurs smaller area overhead as compared to the conventional ECCs. Similarly, the second proposed error correction scheme provides a comparable reliability and higher performance than the conventional ECCs.

**Chapter 7 – Tolerating Faults in Memory Cell Array and Decoder** – presents the fault-tolerant scheme used to tolerate faults in the memory cell array and decoder of emerging memories. The second error correction scheme introduced in Chapter 6 is utilized and optimized to realize the fault-tolerant decoder. The chapter begins with a description of the architecture of an emerging memory incorporated with basic decoder (without fault tolerance and cost-effective implementation). Thereafter, it introduces the proposed architecture consisting of a fault-tolerant decoder; it also describes the modified decoding process for the fault-tolerant decoder. Then, both architectures are compared in terms of their fault tolerance capability and the implementation cost. The evaluation shows that the proposed fault-tolerant, cost-effective architecture provides better memory system reliability, faster decoding performance and smaller area overhead (for large memory word) as compared to the emerging memory architecture incorporated with basic decoder.
Chapter 8 – Tolerating Faults in Memory Cell Array and Vias – proposes the fault-tolerant scheme used to tolerate faults in the memory cell array and CMOS to Non-CMOS Vias (CNVs) of emerging memories. The chapter begins with a description of the CNVs concept in a three-dimensional memory architecture and their defect cases that might occur. Then, it presents the memory architecture incorporated with a conventional RRNS code, which tolerates faults in the memory cell array. After that, the chapter introduces the modified memory architecture incorporated with the fault-tolerant scheme, which tolerates faults in the memory cell array and CNVs. Then, both memory architectures are compared in terms of their fault tolerance capability and their implementation cost. The evaluation shows that the proposed fault-tolerant scheme provides better reliability at a comparable area overhead and performance penalty as compared to the memory architecture without fault-tolerant scheme.

9.2 Thesis contributions

The specific contributions of this thesis are given as follows.

1. A taxonomy of defects that may occur in RRAM.

A theoretical framework of possible defects in RRAM cell array and vias has been established. The definition, the failure mechanisms and the impacts on the defective cell have been discussed.

2. An analysis of resistive open defects in RRAM.

A case study of resistive open defects that impact a single-cell RRAM cell has been performed using an electrical model and simulator. The resistive open defects mimic the variabilities introduced in the RRAM cell, nanowires and vias during the fabrication process. Two unique fault models have been derived from the faulty behaviors of the defective cell.

3. A development of two Design-for-Testability schemes that facilitate memory tests to detect the unique faults in RRAM.

Based on the behavior of the two unique faults, it is deduced that the typical manufacturing tests cannot be used. Two DfT schemes are introduced to facilitate the manufacturing tests to detect the faults. The DfT schemes are developed in such a way that they exploit the required voltage and time for the RRAM cell to set its resistance to the state corresponding to logic value of 1 or 0. Apart from being able to detect the unique faults, the DfT circuits requires insignificant circuit modification.
4. An introduction of an area-efficient error correction scheme that corrects clustered and multiple random errors in the memory array of emerging memories.

The conventional RRNS code used in the current applications provides higher fault tolerance capability at bigger area overhead (due to its longer codeword size) as compared to the well-known RS code. In order to reduce the area overhead while maintaining the fault tolerance capability, the conventional RRNS is modified by intentionally violating one of the coding properties and devising an extra likelihood calculation. The evaluation shows that the modified RRNS code requires smaller overhead and provides competitive fault tolerance capability as compared to conventional RRNS and RS codes.

5. An establishment of a high-performance error correction scheme that tolerates clustered errors in the memory array of emerging memories.

Because the conventional RRNS decoding is an iterative operation, the speed of the entire memory operation to some extent is affected. Thus, another modification has been proposed that combines the C-RRNS code with a double modular redundancy scheme. The decoding process is also altered from the conventional one. The resulting error correction scheme provides high-performance decoding while maintaining the fault tolerance capability at a competitive level. The altered RRNS decoding procedure enables a reduced area overhead of the decoding circuit for large memory word size as compared to that of conventional RRNS decoding.

6. An assessment of a masking scheme that is combined with the high-performance error correction scheme to tolerate errors in the peripheral circuits of emerging memories.

The decoder of the proposed high-performance error correction scheme is designed so that it can tolerate transient and intermittent faults. The short-period glitches caused by the faults are masked by asynchronous gates that form one of the units in the RRNS decoder; this fault-tolerant scheme ensures the glitches do not affect the logic value sent out of the decoder (and in turn the memory). The employed asynchronous gates incur an insignificant area overhead to the overall decoder design.

7. An evaluation of interleaving scheme that is combined with an error correction scheme to tolerate errors in the vias of emerging memories.

The CMOS to Non-CMOS Vias (CNVs) are the crucial units that transfer data between the memory cell array stacked on the top and the peripheral circuits. Faults that occur in CNVs caused by coupling are tolerated by
interleaving the data transferred between the two memory parts. An interleaver and de-interleaver are introduced to perform this task. In addition to the RRNS code used to protect the memory cell array, interleaving ensures that erroneous read data due to faults in CNVs can be corrected during decoding. An insignificant overhead is generated from the introduction of the scheme.

Besides the specific contribution mentioned above, three literature reviews on matters leading to the research in this thesis have been published in international conferences as follows.

1. **An identification of the limitations of CMOS technology that leads to the end of its use.**

Several challenges faced by CMOS technology have been identified in the early phase of the research [2]. The challenges are classified into five categories including physical, material, power-thermal, technological and economical. A conclusion from this discussion is that new technologies are required to complement and replace CMOS in future.

2. **A survey of numerous promising nanodevices to replace CMOS devices.**

Alternative devices that are potentially able to overcome the limitation in CMOS technology have been surveyed [112]. The devices are classified according to the physical phenomena driving their operations such as electrical, magnetic and mechanical. Detailed analysis and comparison of the different classes are also presented, including their structures, advantages, disadvantages and potential applications. The main candidate to replace CMOS devices is also proposed.

3. **An overview of several potential nanoarchitectures for future electronic circuits.**

Different hybrid CMOS/nanodevice nanoarchitectures structured using crossbar arrays have been reviewed [51]. The hybrid architectures harness the advantage of CMOS maturity and the potentials of nanodevice (e.g. high density, low power consumption, reduced manufacturing costs, new functionality, etc.), to have better scalability, functionality and performance than the existing CMOS-based architectures. The discussion includes the concept of crossbar nanoarchitecture, as well as a classification and a detailed description of the considered nanoarchitectures. The most promising crossbar-based hybrid nanoarchitecture is suggested.
9.3 Recommendations for future research

The topics that can be considered for future research concerning the quality and reliability field of emerging memories are as follows.

1. **A fault analysis and test for multiple RRAM cells.**

Chapter 4 presented a single-cell fault modeling and analysis using an electrical RRAM model. Detection conditions and analysis of the faulty behavior only focuses on a single cell while ignoring the adjacent cells. However, because such emerging memories are expected to suffer from high defect densities, fault analysis can be extended to two or more cells. Subsequently, the faulty behaviors of multiple cells can be examined in order to derive the realistic fault models. Then, the efficient memory tests could be generated from the derived fault models.

2. **A fault analysis and test for bridge defects and coupling effects.**

The single-cell fault analysis performed in this thesis focuses only on open defects. Bridge defects are also expected to occur in the RRAM cell array as mentioned in Section 4.1 of Chapter 4. Moreover, because the nanowires in the RRAM cell array are extremely close to each other, coupling effect between neighboring nanowires will impact the cell behavior. Therefore, in addition to the open defects addressed in this thesis, the appropriate detection conditions, fault modeling, fault analysis and test development can be extended to bridges and coupling effects. Moreover, their impact on single and multiple cells can also be considered.

3. **A faster RRNS encoding and decoding.**

Chapter 6, 7 and 8 introduced several fault-tolerant schemes that are based mainly on Redundant Residue Number System (RRNS) code. The conversion from the RRNS codewords to binary during the decoding imposes a quite high performance penalty and area overhead. An improved RRNS encoding and decoding scheme can be investigated using faster and more efficient conversion scheme such that proposed by [113].

4. **A fault-tolerant scheme for read circuit of RRAM.**

Chapter 7 mentioned that not only the memory cell array, but also the peripheral circuits must be protected as CMOS devices are becoming more sensitive to faults. For RRAM, read circuit is one of the crucial parts in the peripheral circuits where it must provide a balanced negative and positive pulsewidth in order to alleviate soft error problems after several read operations [55]. Yet, external
disturbances or aging-related faults in read circuit might perturb the pulsewidth. Therefore, in order to ensure that the read circuit is reliable (fault-tolerant) during operation, an appropriate fault-tolerant scheme must be introduced for such circuits.

5. **Fault-tolerant scheme for CMOS to Non-CMOS Vias.**

Chapter 8 indicated that CMOS to Non-CMOS Vias (CNVs) are used to connect the memory cell array fabricated above the peripheral circuits. A fault in CNVs of a three-dimensional (3D) memory such as RRAM impacts the data access between the two emerging memory parts. Since the distance among CNVs is very close, coupling effects will be occurred. In order to ensure that these crucial interconnections can tolerate the coupling effects, body contacts can be inserted between the CNVs. An investigation into the modeling, the appropriate distance, size and orientation, and simulation of the body contacts implementation in 3D RRAM can be carried out.

6. **A complete fault-tolerant architecture that combines the Optimized D3R-based architecture with the Interleaved RRNS architecture.**

Chapter 8 presented the Interleaved RRNS architecture proposed to tolerate fault in the memory cell array and the CNVs. However, the ECC used in this architecture is the conventional RRNS (C-RRNS) code, which incurs larger area overhead and higher performance penalty than the D3R code. Moreover, D3R code has been employed to develop a cost-effective, fault-tolerant architecture that addresses faults in the memory cell array and decoder. Therefore, a complete fault-tolerant architecture can be developed by combining the Optimized D3R-based architecture and the interleaved scheme to address fault in all three memory parts: the memory cell array, decoder and vias.

7. **Transparent memory test development based on symbol-based ECCs.**

Transparent memory testing is an on-line testing scheme that exploits the contents of the memory under test as part of the test stimuli; the memory under test preserves its original content after the testing is completed if no faults occur [114]. This on-line memory testing ensures the reliability of storage data during a life-time operation; it can also detect faults that escape the manufacturing testing. Existing work on transparent memory testing utilizes bit-oriented error correction codes such as Hamming code [114, 115]. No work has investigated symbol-based ECCs such as RRNS code for the on-line testing scheme; hence, this combined testing and fault tolerance research area could be an effective approach to improve the reliability of emerging memories.
A1. Example of RRNS encoding

An 8-bit input data of a memory system will be encoded into RRNS codeword based on the moduli set, e.g., \( \{ m_1, m_2, m_3, m_4, m_5 \} = \{ 5, 7, 8, 9, 11 \} \). The first three moduli are non-redundant moduli \( m_i \), and the last two moduli are redundant moduli \( m_j \). This RRNS code has the correction capability of \( t = \frac{5-3}{2} = 1 \). Note that the operating legitimate range \( M_{op} \) for this RRNS codeword is \( [0, 2^8 - 1 = 255] \). Therefore, RRNS codeword for input data \( X = 234 \) is:

\[
\begin{align*}
x_c &= \{|X|m_1, |X|m_2, |X|m_3, |X|m_4, |X|m_5\} \\
x_c &= \{4, 3, 2, 0, 3\}
\end{align*}
\]

The total bit length of \( x_c \) is 17 bits; this is because the first three moduli require three bits each, while the fourth and the fifth modulus require 4 bits each.

A2. Example of RRNS decoding

Assume that the fourth residue of the codeword in Appendix A1 (i.e., 0) is corrupted during the storage and thereby resulting \( x_c = \{4, 3, 2, 8, 3\} \). First, the decoding process using CRT will be given and thereafter MRC.

Before that, parameters such as legitimate ranges, multiplicatives, multiplicative inverses and mixed radix digits are required to accomplish the decoding. These parameters can be pre-calculated; for instance, \( M = \prod_{c=1}^{n} m_c = 27720 \) and \( M_{op} = 2^d - 1 = 255 \). The other parameters as given in Table 1.
The following calculation shows the RRNS decoding process based on CRT.

\[
X = \left( M_1 | x_1 \times M_1^{-1} | \mod m_1 \right) + \left( M_2 | x_2 \times M_2^{-1} | \mod m_2 \right) + \ldots + \left( M_5 | x_5 \times M_5^{-1} | \mod m_5 \right)
\]

\[
= \left[ \frac{(5544 \times 4 + 3960 \times 3 + 3465 \times 1 + 3080 \times 5 + 2520 \times 1)}{27720} \right]
\]

\[
X = 12554 > 255
\]

The following calculation shows the RRNS decoding process based on MRC.

\[
X = x_1 + (v_2 \times m_1) + (v_3 \times m_1 \times m_2) + (v_4 \times m_1 \times m_2 \times m_3) + (v_5 \times m_1 \times m_2 \times m_3 \times m_4)
\]

\[
= 4 + (4 \times 5) + (6 \times 5 \times 7) + (8 \times 5 \times 7 \times 8) + (4 \times 5 \times 7 \times 8 \times 9)
\]

\[
X = 12554 > 255
\]

Because both calculation results in a value larger than the operating legitimate range, errors have been detected. The error correction procedure is invoked because the codeword is invalid. A systematic iterative calculation that discards a residue (and its corresponding moduli) in each iteration is performed for a maximum \(C^*_T = 5\) iterations. This exhaustive search produces the integer values as shown in Table 2. It recovers the correct value when \(x_4\) is discarded where it results in \(X = 234\), which is within the operating legitimate range.
Table 2: Calculated data during correcting phase

<table>
<thead>
<tr>
<th>Iteration (g)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discarded Residue ((x_g))</td>
<td>(x_1)</td>
<td>(x_2)</td>
<td>(x_3)</td>
<td>(x_4)</td>
<td>(x_5)</td>
</tr>
<tr>
<td>Recovered Data ((X'_g))</td>
<td>1466</td>
<td>3594</td>
<td>2159</td>
<td>234</td>
<td>2474</td>
</tr>
</tbody>
</table>

**A3. Example of maximum likelihood decoding**

Assume a 16-bit memory word encoded into the 6M-RRNS code based on the non-redundant moduli \(m_i = \{257, 256\}\) and the redundant moduli \(m_j = \{127, 63, 31, 17\}\). The first residue of the codeword \(x = \{221, 0, 72, 18, 9, 2\}\) (represents \(X = 9216\)) is corrupted during storing and is read as \(x' = \{0, 0, 72, 18, 9, 2\}\). During decoding, the erroneous residue is detected and is corrected. After the maximum iteration of the correction has been reached, there are two decoded data with value less than \(M_{op} = 2^{16} - 1\): \(X'_1 = 9216\) when \(m_1\) and \(m_2\) are discarded, and \(X'_2 = 257\) when \(m_3\) and \(m_6\) are discarded. This ambiguity requires MLD to determine the authentic output data as follows.

First, \(X'_1\) and \(X'_2\) are re-encoded producing \(x'_1\) and \(x'_2\), respectively.

\[
x'_1 = \{[9216]_{257}, [9216]_{256}, [9216]_{127}, [9216]_{63}, [9216]_{31}, [9216]_{17}\}
= \{221, 0, 72, 18, 9, 2\}
\]

\[
x'_2 = \{[257]_{257}, [257]_{256}, [257]_{127}, [257]_{63}, [257]_{31}, [257]_{17}\}
= \{0, 1, 3, 5, 9, 2\}
\]

Then, the resulting residues \(x'_1\) and \(x'_2\) are compared with the the read codeword \(x'\). This comparison reveals that \(x'_1\) has Hamming distance \(d_{min} = 1\) (i.e., differs in the first residue) while \(x'_2\) has \(d_{min} = 3\) (i.e., differs in second, third and fourth residues). Therefore, the authentic output data is \(X'_1 = 9216\).

**A4. Example of D3R encoding**

A 16-bit input data \(X = 65535\) is encoded into D3R codeword based on moduli set \(\{m_1, m_2, m_3\} = \{255, 511, 512\}\). The encoded data for the original D3R codeword is as follows.
$C = \{65535_{255}, 65535_{511}, 65535_{512}\}$

$C = \{0, 127, 511\}$

Then, $C$ is duplicated producing $D3R = \{0, 127, 511\}$.

### A5. Example of D3R decoding

Assume that faults induce clustered errors that corrupt all three residues of $C'$ of the D3R codeword in Appendix A1 resulting into $D3R = \{0, 127, 511, 3, 255, 31\}$. The decoding process based on Mixed-Radix Conversion starts by calculating the mixed-radix digits; thereafter, the conversion of RNS to binary (decimal) for comparison to operation legitimate range.

For the original part $C$, the output binary data is:

$v_{1(new)} = x_3 = 511$

$v_{2(new)} = \left| (x_2 - v_{1(new)}) \times b_{12(new)} \right|_{m_{0(new)}} = \left| (127 - 511) \times 1 \right|_{511} = 127$

$v_{3(new)} = \left| (x_1 - v_{1(new)}) \times b_{13(new)} - v_{2(new)} \times b_{23(new)} \right|_{m_{2(new)}}$

$= \left| (0 - 511) \times 128 - 127 \right|_{255} = 0$

$X_C = v_{1(new)} + (v_{2(new)} \times m_{1(new)}) + (v_{3(new)} \times m_{1(new)} \times m_{2(new)})$

$= 511 + (127 \times 512) + (0 \times 512 \times 511) = 65535$

For the duplicated part $C'$, the output binary data is:

$v'_{1(new)} = x'_3 = 31$

$v'_{2(new)} = \left| (x'_2 - v'_{1(new)}) \times b_{12(new)} \right|_{m_{2(new)}} = \left| (255 - 31) \times 1 \right|_{511} = 224$

$v'_{3(new)} = \left| ((x'_2 - v'_{1(new)}) \times b_{13(new)} - v'_2 \times b_{23(new)} \right|_{m_{3(new)}}$

$= \left| ((3 - 31) \times 128 - 224) \right|_{255} = 17$

$X_{C'} = v'_{1(new)} + (v'_{2(new)} \times m_{1(new)}) + (v'_{3(new)} \times m_{1(new)} \times m_{2(new)})$

$= 31 + (224 \times 512) + (17 \times 512 \times 511) = 4562463$

Because $X_C \leq LR$, it is valid; on the other hand, $X_{C'} > LR$ it is invalid. Therefore, $X_C = 65535$ is taken as the output data.
BIBLIOGRAPHY


LIST OF PUBLICATIONS

International Journals


International Conferences/Workshops


**Digest**


**Other publications not directly related to this thesis**


Nor Zaidi bin Haron was born on 7th of April, 1976 in Johor, Malaysia. After finishing primary education at his hometown, he attended a secondary education at Royal Military College in Kuala Lumpur, Malaysia from 1989 to 1993. In 1997 and 2001, respectively, he received his Diploma in Electronic Engineering and Bachelor of Engineering in Electrical Engineering degree both from Universiti Teknologi MARA, Malaysia. In 2004, he obtained his Master of Science in Microelectronics degree from University of Newcastle upon Tyne, United Kingdom. Nor Zaidi worked as a test technician and design engineer in several manufacturing companies in Malaysia. Currently, he is a lecturer at Universiti Teknikal Malaysia Melaka, Malaysia.

In November 2007, he has been awarded a scholarship from Ministry of Higher Education of Malaysia to pursue Philosophy Doctorate degree at Computer Engineering Laboratory (CE), Delft University of Technology (TU Delft). At TU Delft, he worked under the supervision of Dr. ir. Said Handioui on the topic of testing and fault tolerance for emerging nanoelectronic memories. His research interests include memory fault modeling and testing, design-for-testability, fault tolerance, design-for-reliability and nanoarchitectures. Nor Zaidi has published several papers in the field of electrical defect simulation, fault modeling and test, as well as fault-tolerant techniques for emerging memories. During his PhD study, he co-supervised two master degree students at CE, TU Delft. He also served as the reviewer of a number of IEEE/ACM journals and conferences. Nor Zaidi is a member of the Institute of Electrical and Electronics Engineers (IEEE) and Board of Engineer Malaysia (BEM).