Karlsruhe Institute of Technology

# Design and Test of Spintronic Compute-in-Memory Architectures

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Christopher Münch

aus Heidelberg

---

---

| | |
|---|---|
| Tag der mündlichen Prüfung: | 25. Januar 2023 |
| Erster Referent: | Prof. Dr. Mehdi B. Tahoori<br>Karlsruher Institut für Technologie (KIT) |
| Zweite Referentin: | Prof. Dr. Lorena Anghel<br>Universität Grenoble-Alpes |

## Acknowledgments

This work would have not been possible without the continuous support of many people. For this, I would like to thank my advisor Prof. Dr. Mehdi B. Tahoori, who was always eager to discuss and challenge my ideas on countless occasions. He helped to refine them and guided me through the jungle of academia to make these ideas a reality in form of this thesis. Furthermore, I would like to thank Prof. Dr. Lorena Anghel for her involvement in this work as a second reviewer.

I want to thank all of my coauthors for their collaboration. Specifically, Dr.-Ing. Rajendra Bishnoi who introduced me to the world of Spintronics and who was mentoring me during the start of my PhD. I would like to thank Dr.ir. Moritz Fieback for all the professional and private discussions we had, in particular during the corona lock-downs. An additional thanks goes to Dr. Jongsin Yun for broadening my understanding on the industrial perspective on Spintronics alongside all of our technical discussions.

A wholeheartedly *Thank You* goes to my partner Iris and my parents Hilla and Axel as well as the rest of my family for encouraging and supporting me through the years.

Last but not least, a big thanks to my friends and colleagues at the Chair of Dependable Nano Computing. In particular, I would like to thank our secretary Iris Schröder-Piepka for her constant support on navigating all the administrative hurdles at the university. Thank you all for a welcoming work environment, for the openness and for the fun we had alongside all the work.

Hiermit erkläre ich an Eides statt, dass ich die von mir vorgelegte Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben haben und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Karlsruhe, 15. Januar 2023
Christopher Münch

# Abstract

Modern computer architectures strive to support the ever increasing demand of computational power resulting from a growing pool of complex tasks. Many modern applications base their results on a huge set of data, which needs to be stored and ready to be used for computation. Yet, the memory performance does not scale the same as the performance of the processing unit, resulting in a growing disparity between the two, the so-called *memory wall*. Because of this memory wall, new computational concepts are in the focus for state-of-the-art hardware accelerators and fully integrated embedded systems.

*Emerging resistive non-volatile memories*, like Spin-Transfer-Torque Magnetic Random Access Memory (STT-MRAM), can be used to build memory architectures fulfilling all the needs for conventional storage. In addition, they offer the ability to easily calculate the result of basic and even more complex operations directly in the memory itself. These Compute-in-Memory (CiM) architectures allow to save energy and speed up calculations by significantly reducing the data traversing the memory hierarchy, e.g., from the memory through multiple levels of cache to the processing unit and back. With this, the effects of the memory wall can be mitigated for a broad range of applications.

However, to support CiM operations in STT-MRAM the memory devices need to be properly tested after manufacturing specifically for CiM operations. This is because CiM operations are much more susceptible to process variations and thus require more testing in addition to their already needed tests for their conventional memory operations like read operations and write operations. Therefore, reducing the needed test resources for the regular memory operations, in terms of time and test equipment, also benefits the overall test resource usage for CiM capable memories. It is therefore needed to optimize the conventional memory tests as well as evaluating known STT-MRAM defects, their implication on CiM behavior and create reasonable means of testing for them to avoid faulty CiM capable memories.

This thesis contributes to the reliable use of Spintronic-based CiM architectures on three levels: First, we show the usage of low temperature test data to ensure the reliable usage of the conventional memory operation at high temperatures while avoiding high temperature testing and how to speed up the conventional memory test by only testing parts of the memory whilst maintaining reliability. Second, we show a new concept for in-memory calculation using a current-controlled oscillator which can be used for highly configurable CiM architectures. Third, the defect behavior for CiM operations of different complexity in STT-MRAM is evaluated in detail and different testing schemes according to the observed faults are being presented.

# Zusammenfassung

Moderne Computerarchitekturen sind darin bestrebt, den ständig steigenden Bedarf an Rechenleistung zu bewältigen, der sich aus der zunehmenden Anzahl komplexer Aufgaben ergibt. Viele moderne Anwendungen basieren auf der Nutzung großer Mengen von Daten, die gespeichert und für Berechnungen vorgehalten werden müssen. Die Geschwindigkeit des Speichers wächst jedoch nicht mit der selben Geschwindigkeit der Verarbeitungseinheit, was zu einer wachsenden Diskrepanz zwischen den beiden führt, der so genannten *Memory Wall*. Wegen dieser Memory Wall stehen neue Berechnungskonzepte im Mittelpunkt moderner Hardware-Beschleuniger und vollintegrierter eingebetteter Systeme.

Aufkommende resistive nichtflüchtige Speicher, wie Spin-Transfer-Torque Magnetic Random Access Memory (STT-MRAM), können verwendet werden, um Speicherarchitekturen aufzubauen, die alle Anforderungen an konventionelle Speicher erfüllen. Darüber hinaus bieten sie die Möglichkeit, das Ergebnis einfacher und sogar komplexer Operationen direkt im Speicher selbst zu berechnen. Diese Compute-in-Memory (CiM)-Architekturen ermöglichen es, Energie zu sparen und Berechnungen zu beschleunigen, indem sie die die Speicherhierarchie durchlaufenden Daten vom Speicher durch mehrere Cache-Ebenen zur Verarbeitungseinheit und zurück erheblich reduzieren. Auf diese Weise können die Auswirkungen der Memory Wall für eine breite Palette von Anwendungen abgemildert werden.

Um CiM-Operationen in STT-MRAM zu unterstützen, müssen die Speicherzellen jedoch nach der Herstellung speziell für den CiM-Betrieb getestet werden. Dies liegt daran, dass CiM Operationen viel anfälliger für Prozessherstellungsschwankungen sind und daher zusätzliche Tests zu den bereits erforderlichen Tests für ihre konventionellen Speicheroperationen wie Lese- und Schreibvorgänge benötigen. Daher ist eine Reduzierung der erforderlichen Testressourcen zur Sicherstellung regulären Speicheroperationen in Bezug auf Zeit und Testausrüstung sinnvoll, da dies sich auch auf die Gesamtnutzung der Testressourcen für CiM-fähige Speicher auswirkt. Es ist daher erforderlich, die konventionellen Speichertests zu optimieren sowie bekannte STT-MRAM Defekte und ihre Auswirkungen auf das CiM-Verhalten zu evaluieren und angemessene Testmöglichkeiten zu schaffen, um fehlerhafte CiM-fähige Speicher zu vermeiden.

Diese Arbeit leistet einen Beitrag zum zuverlässigen Einsatz von Spintronic-basierten CiM-Architekturen in drei Gebieten: Erstens zeigen wir die Verwendung von Niedertemperaturtestdaten, um die Zuverlässigkeit der konventionellen Speicheroperationen bei hohen Temperaturen sicherzustellen und wie konventionelle Speichertest beschleunigt werden können, indem nur Teile des Speichers unter Beibehaltung der Zuverlässigkeit getestet werden. Zweitens zeigen wir ein neues Konzept für In-Memory-Berechnung unter Verwendung eines stromgesteuerten Oszillators, der für konfigurierbare CiM-Architekturen verwendet werden kann. Drittens wird das Defektverhalten für CiM-Operationen unterschiedlicher

Komplexität in STT-MRAM detailliert ausgewertet und verschiedene Testverfahren entsprechend den beobachteten Fehlern vorgestellt.

# Contents

## List of own publications included in this thesis

[69] C. Münch, J. Yun, M. Keim, and M. B. Tahoori. "MBIST-based Trim-Search Test Time Reduction for STT-MRAM". in: *IEEE VLSI Test Symposium*. 2022.

[70] C. Münch, J. Yun, M. Keim, and M. B. Tahoori. "MBIST-supported Trim Adjustment to Compensate Thermal Behavior of MRAM". in: *IEEE European Test Symposium*. 2021, pp. 1–6. DOI: 10.1109/ETS50041.2021.9465383. (Best Paper Candidate)

[67] C. Münch, N. Sayed, R. Bishnoi, and M. Tahoori. "A Novel Oscillation-Based Reconfigurable In-Memory Computing Scheme With Error Correction". In: *IEEE Transactions on Magnetics* 57.2 (2020).

[75] S. M. Nair, C. Münch, and M. B. Tahoori. "Defect Characterization and Test Generation for Spintronic-based Compute-In-Memory". In: *2020 IEEE European Test Symposium (ETS)*. 2020. (Best Paper Candidate)

[63] C. Münch and M. B. Tahoori. "Defect Characterization of Spintronic-based Neuromorphic Circuits". In: *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 2020, pp. 1–4.

[68] C. Münch and M. B. Tahoori. "Testing resistive memory based neuromorphic architectures using reference trimming". In: *Design, Automation & Test in Europe Conference & Exhibition*. 2021.

## List of other publications not included in this thesis


[65] C. Münch, R. Bishnoi, and M. B. Tahoori. "Reliable in-memory neuromorphic computing using spintronics". In: *Proceedings of the Asia and South Pacific Design Automation Conference*. ACM. 2019.

[66] C. Münch, R. Bishnoi, and M. B. Tahoori. "Tolerating retention failures in neuromorphic fabric based on emerging resistive memories". In: *Asia and South Pacific Design Automation Conference*. 2020.

[3] S. T. Ahmed, M. Mayahinia, M. Hefenbrock, C. Münch, and M. B. Tahoori. "Process and Runtime Variation Robustness for Spintronic-Based Neuromorphic Fabric". In: *2022 IEEE European Test Symposium (ETS)*. IEEE. 2022, pp. 1–2.

[12] R. Bishnoi, L. Wu, M. Fieback, C. Münch, S. M. Nair, M. Tahoori, Y. Wang, H. Li, and S. Hamdioui. "Special Session – Emerging Memristor Based Memory and CIM Architecture: Test, Repair and Yield Analysis". In: *2020 IEEE 38th VLSI Test Symposium (VTS)*. 2020, pp. 1–10.

[58] M. Mayahinia, C. Münch, and M. B. Tahoori. "Analyzing and Mitigating Sensing Failures in Spintronic-based Computing in Memory". In: *2021 IEEE International Test Conference (ITC)*. IEEE. 2021, pp. 268–277.

[1] S. T. Ahmed, M. Hefenbrock, C. Münch, and M. B. Tahoori. "Neuroscrub: Mitigating retention failures using approximate scrubbing in neuromorphic fabric based on resistive memories". In: *2021 IEEE European Test Symposium (ETS)*. IEEE. 2021, pp. 1–6.

[39] A. Gebregiorgis, L. Wu, C. Münch, S. Rao, M. B. Tahoori, and S. Hamdioui. "Special Session: STT-MRAMs: Technology, Design and Test". In: *2022 IEEE 40th VLSI Test Symposium (VTS)*. IEEE. 2022, pp. 1–10.

[2] S. T. Ahmed, M. Hefenbrock, C. Münch, and M. B. Tahoori. "NeuroScrub+: Mitigating Retention Faults Using Flexible Approximate Scrubbing in Neuromorphic Fabric Based on Resistive Memories". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[64] C. Münch, R. Bishnoi, and M. B. Tahoori. "Multi-bit non-volatile spintronic flip-flop". In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2018, pp. 1229–1234. (Best Paper Candidate)

[34] M. Fieback, C. Münch, A. Grebegiorgis, G. C. Medeiros, M. Taouil, S. Hamdioui, and M. Tahoori. "PVT Analysis for RRAM and STT-MRAM-based Logic Computation-in-Memory". In: *IEEE European Test Symposium*. 2022. (Best Paper Candidate)

# Part I.

# Preliminaries

# 1. Introduction

Modern computer architectures need to support the usage of an ever-increasing amount of data. Big Data applications in general and neuromorphic applications specifically depend on the capability to perform operations on large chunks of memory. Applications like Neural Networks can easily demand storage for millions of parameters, which are needed on a constant basis [6].

However, to perform these operations the data has to traverse the memory hierarchy from the typically large but slow storage, through main memory, multiple layers of caches to the CPU where the data is used and the result is written back, again through the entire memory hierarchy. As the memory is slower than the CPU, this access has a huge impact on the entire operation. The gap between memory and CPU speed additionally increases with newer generations of memory and computing circuits, thus worsening the so-called memory wall.

There are two directions to investigate solutions to the problems arising from the memory wall: The memory technology and the computational concept. The readily available and bleeding-edge memory technologies offer different features. Regular Static and Dynamic RAM (SRAM/DRAM) are well proven, but lack the capability to store data in a non-volatile way. They have to be constantly supplied with power to make sure that their content is preserved. In contrast, emerging resistive non-volatile memories (NVMs) like the Magnetic Tunneling Junction (MTJ) based Spin-Tranfer-Torque Magnetic RAM (STT-MRAM) [51], Resistive RAM (ReRAM) [95, 4] or Phase Change Memory (PCM) [96] offer a non-volatile storage, in addition to other features.

To mitigate the effects of the memory wall, Compute-in-Memory (CiM) architectures offer the ability to perform operations directly in the memory, without traversing the memory hierarchy. This helps to save the associated time and energy for the data transfer from the memory to the CPU and back. Whilst there are CiM solutions using SRAM and DRAM [83, 76, 46], resistive NVMs allow for easy to integrate implementations of new CiM concepts because of their inherently resistive nature. Their resistive behavior can be used to perform in-memory operations with only small adjustments to their regular RAM architecture. In addition, they offer several other device features, like a high integration density and good cell scaling in general, which makes them very attractive in comparison to regular SRAM and DRAM [77, 51, 97].

New technology always introduces new defects leading to faults, which need to be identified, detected and mitigated [34]. Specifically, STT-MRAM has been able to reach a mature state over the last few years with regards to its usage as regular RAM and cache, which can be seen by a variety of different publications from key industrial manufacturers [98, 17, 73, 53]. However, the introduction of the CiM concept to STT-MRAM mandates the thorough evaluation of the currently available test methodology for the conventional STT-MRAM. By ensuring the functionality of CiM-capable STT-MRAM not only for

the conventional memory operations, but also for its CiM operations, many different applications can profit from reliable in-memory operations to optimize their calculations.

## 1.1. Contributions and Research Directions

This thesis makes contributions to the reliable usage of STT-MRAM as a CiM-capable memory on multiple levels by improving the test solutions for the conventional memory operations, introducing a new CiM calculation design and proposing test procedures for individual CiM operations.

### 1.1.1. Conventional Memory Operation Test

In state-of-the-art architectures, resistive memories base their read operation on the comparison of the resistive memory cell to a reference resistance. Reference resistance trim circuits allow to use manufacturing test data from the memory to calibrate the reference resistance and mitigate process and temperature variations. Generally, larger memories need more test time for this calibration. We therefore improved the test time for STT-MRAM by introducing a progressive sampling test, which modifies the on-chip test infrastructure to perform the calibration based on reduced test data in Section 3.2. This work has also been published in [69] together with Jongsin Yun, Martin Keim and Mehdi Tahoori. In addition, the memory has to perform its operation within the expected operational temperature. Testing and calibrating at very high temperatures to ensure functionality over the entire expected temperature range is demanding from a technical point of view and stresses the device. To allow for a calibration which factors in high temperature chip behavior without expensive high temperature test data, a model-based temperature behavior prediction is presented in Section 3.1. It uses technology characterization data to predict high temperature memory behavior from low temperature test data to calibrate the trim circuit for the entire operation temperature range. This work has also been published in [70] together with Jongsin Yun, Martin Keim and Mehdi Tahoori.

### 1.1.2. Compute-in-Memory Design

Different concepts and flavors for in-memory computation are possible. A common approach for resistive memory-based CiM architectures is to make use of Kirchhoff's laws and use the impact of multiple STT-MRAM cells on a current passing through the cells in parallel to calculate the result of simple Boolean operations. For this, the current is either used to conditionally switch a target cell and thus store the result of the operation, or it is directly evaluated with a reference resistance like in a conventional read operation. More complex functions can be realized by either performing multiple different read operations to relate the read current to a result or by using a current-controlled oscillator circuit to reduce the needed hardware for the operation. The concept and evaluation of this oscillator-based approach have been published in [67] together with Nour Sayed, Rajendra Bishnoi and Mehdi Tahoori, and are discussed in Section 4.

### 1.1.3. Compute-in-Memory Test

As only a few modifications to the conventional STT-MRAM are needed to add CiM capabilities, a large part of the standard test procedure to find cell defects can be used. However, to ensure the functionality of the memory specifically for CiM operations, additional tests are necessary. These contributions are discussed in Section 5.1 and have been published in [75] together with Sarath Mohanachandran Nair and Mehdi Tahoori. Generally, allowing more complex CiM operations introduces new faults from defects which need to be tested for. This is shown for an increased number of operands for a binary in-memory AND operations in Section 5.2 which is based on the work and has been published in [63] together with Mehdi Tahoori. To reduce test time even further, the resistive behavior of the STT-MRAM cells can be exploited and thus multiple defects can be checked at the same time with only a reduced amount of test cases. This is discussed in Section 5.3 and has been published in [68] together with Mehdi Tahoori.

## 1.2. Dissertation Outline

This thesis is organized into the chapters as follows:

- In Chapter 2 the theoretical background to this thesis is covered. Device and architecture level descriptions for STT-MRAM are given as well as their manufacturing implications and the reference trimming concept to mitigate them. In addition, the usage of STT-MRAM for CiM operations is shown.
- Chapter 3 discusses possibilities to improve conventional memory test methodologies. This is done by improving the post-manufacturing trim search with different approaches.
- The concept and implementation of a new oscillation based CiM architecture is shown in Chapter 4.
- In Chapter 5 the implications of MTJ defects on CiM operations and possible testing solutions are presented.
- Chapter 6 concludes the thesis and gives an outlook to the possible influences of the presented work.

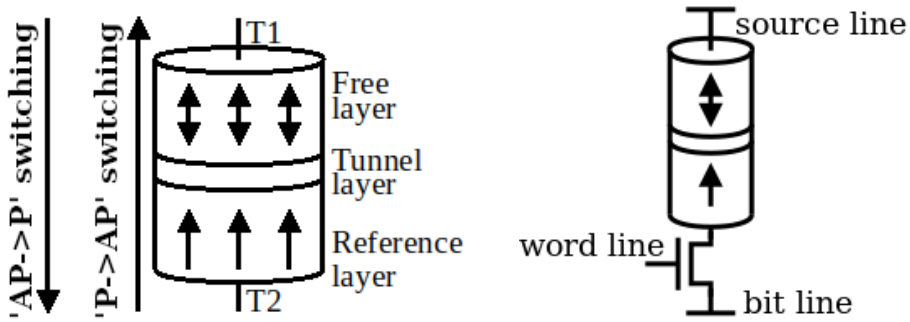
# 2. Background

## 2.1. Resistive Non-volatile Memories

There are multiple different resistive non-volatile memory technologies currently considered by academia and industry. The core work of this thesis is based on Spintronic STT-MRAM, which centers around the MTJ and will be discussed in detail in the following. Afterwards, there will be a brief overview of the main competitors to STT-MRAM, namely ReRAM and PCM.

### 2.1.1. Magnetic Tunnel Junctions and Spin Transfer Torque RAM

An MTJ is a multi-layer structure, composed of two ferromagnetic layers separated by an oxide-based tunnel layer (see Fig. 2.1a). One of the ferromagnetic layers has a fixed magnetic orientation, the reference layer (RL), the other one is free, the free layer (FL). A bidirectional write current above the device dependent critical current $I_c$ is used to switch the magnetic orientation of FL to either the same magnetic orientation as RL, the parallel state (P-state) or to the opposite magnetization state, the anti-parallel state (AP-state). In case the write current flows from T1 to T2 (from FL to RL) the MTJ switches the magnetization direction of the FL to the P-state, whereas a current larger than $I_c$ applied the other way around from T2 to T1 (RL to FL) switches the magnetization direction of the FL to the AP state. In addition, this switching is significantly more likely to happen, if the MTJ is in the AP-state [41] as the energy barrier for AP to P switching is lower than the energy barrier for P to AP switching [91]. The key parameter to influence the MTJ behavior is the thermal stability $\Delta$, which has a direct impact on the write time, critical write current and the retention of the cell.




(a) MTJ (b) Bit-cell

Figure 2.1.: STT-MRAM storing device

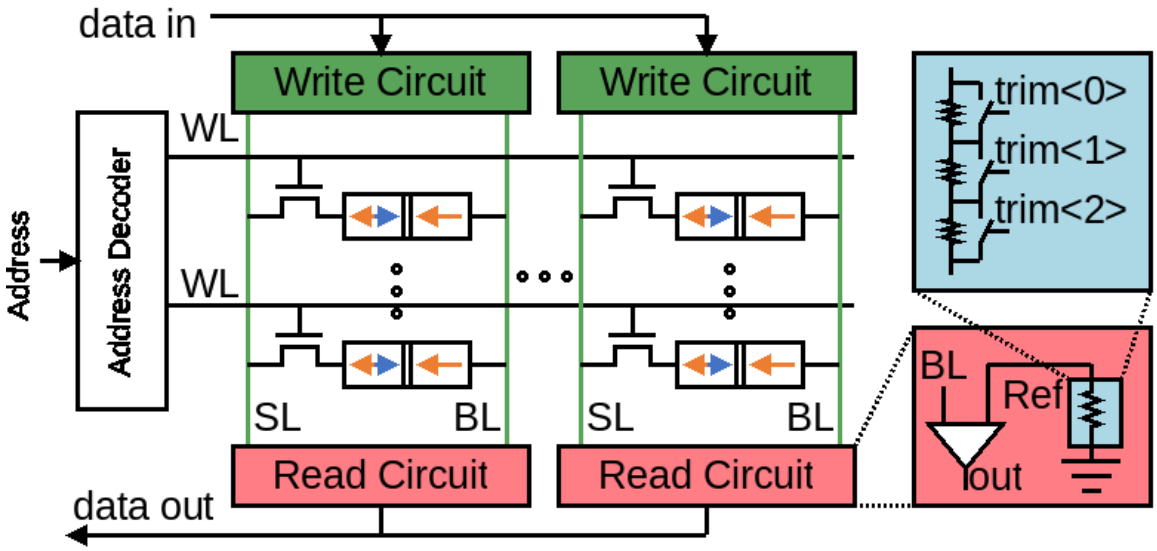


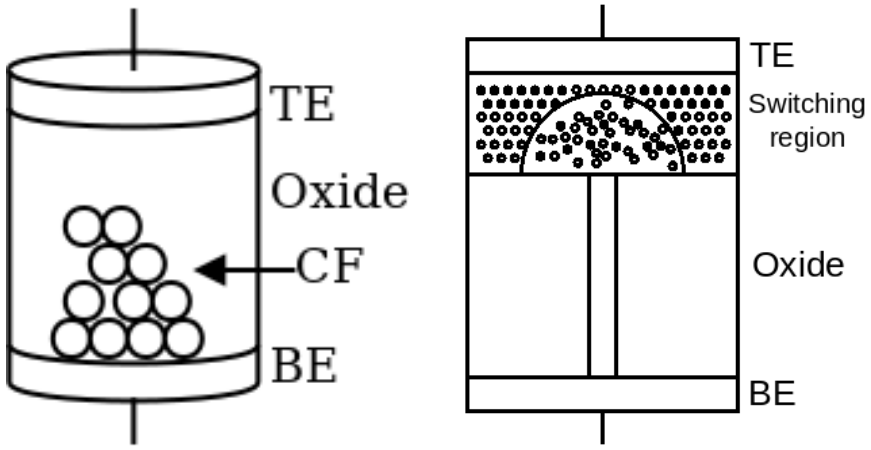

Figure 2.2.: STT-MRAM architecture with reference trimming.

In the AP-state the MTJ exhibits a high resistance ($R_{AP}$), whereas in the P-state its resistance is comparably lower ($R_P$). This resistance can be sensed with the help of a unidirectional small read current $I_r$, with $I_r < I_c$. As the MTJ can be in on of two different resistance states, this small read current can be used to differentiate between the High Resistance State (HRS), which is the AP-state, and the Low Resistance State (LRS), which is the P-state. The HRS and the LRS can in turn be mapped to logical representations (e.g. LRS represents a logical '0' and HRS represents a logical '1').

By using an additional NMOS access transistor in combination with an MTJ, a 1T1MTJ bit-cell can be created as shown in Fig. 2.1b. The access transistor allows the read and write current to flow through the MTJ from sourceline (SL) to bitline (BL) based on the control signal wordline (WL). This bit-cell can then be used to generate a memory array with row-wise access, based on an address decoded by a one-hot decoder and appropriate source-line/bit-line conditioning circuits as well as read sense amplifiers as depicted in Fig. 2.2. The read operation in STT-MRAM is then performed with the help of one current sense amplifier per bitline, which compares the current flow through the MTJ to read and a reference resistance to evaluate the state of cell. With this arrangement, it is possible to store information on a word basis and read it out on the same granularity as conventional SRAM and DRAM [23].

### 2.1.2. Other resistive NVMs

A ReRAM device, as depicted in Figure 2.3a, stores its content as the resistance of a *Conductive Filament* (CF) which is interfaced by two electrodes, namely the Bottom Electrode (BE) and the Top Electrode (TE). There are two categories of ReRAM devices, broadly grouped by their CF composition [12]. The Conductive Bridge RAM (CBRAM) is creating a connection between the bottom and the top electrode by using the dissolution process of an active electrode, whereas the Oxygen-Vacancy (OxRAM) devices are based on oxygen




(a) ReRAM device (b) PCM device (modeled after [15])

Figure 2.3.: Other emerging non-volatile storing devices.

vacancies in the filament. In general, both types can be written by applying a sufficiently high write voltage through the filament. Depending on the polarity of the write voltage, the conduction of the filament is either increased by breaking the bonds between the ions and forming a conductive path of oxygen vacancies in the filament, or it is decreased by moving back ions to the vacancies and thus destructing the conductive path. A high conducting path is resulting in a low resistance of the ReRAM cell and consequently it is in its LRS, whereas the low conducting path is reflected in a high resistive cell in its HRS.

In contrast, PCM (see Fig, 2.3b) is based on a material which can be changed from an amorphous state to a crystalline state [16], sandwiched between two interface electrodes. In its amorphous state the PCM cell exhibits a significantly larger resistance for an electrical pulse than in the crystalline state. To set the cell the high resistance amorphous state (HRS), a high current is passed through the cell and is cut off abruptly. This forces the phase change material to heat up, leading it to melt and stay in that amorphous state. For the PCM to recrystallize and thus changing to the crystalline LRS, a lower current is passed through the cell for a longer period of time, which leads to a crystallization process.

## 2.2. Compute-in-Memory

The CiM paradigm allows executing operations as close to the storage location of the operands as possible. This is done to avoid multiple data transfers between the processor and memory subsystems [57]. CiM architectures merge their logical operations with the memory to implement the logic within the memory itself by modifying the memory architecture. They can be grouped by three main characteristics: How the operands and the result of the operation are represented, where the operation is performed, and which operations can be handled by the architecture [81]. Operands and results either reside within a cell as part of the memory state or are generated dynamically and therefore stateless. The calculation can be done within the core array (CiM-A), as part of the memory periphery (CiM-P), or close to the memory (near memory) on the same die. An

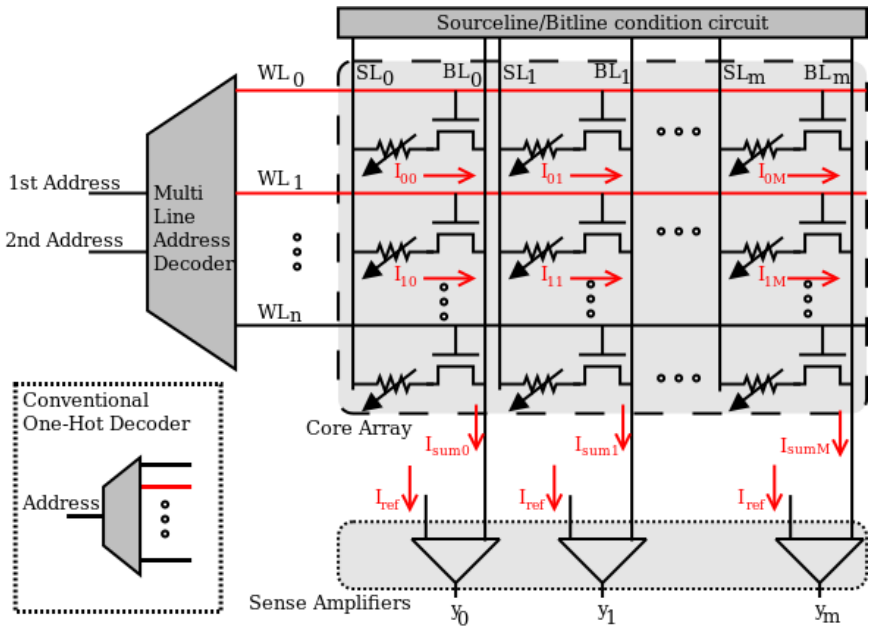



Figure 2.4.: Resistive CiM architecture with multiple rows enabled (In this example: row 0 and row 1). The resulting currents sum up on the bit-line and can be evaluated with the sense amplifiers.

architecture can handle different operations, like bit-wise AND or OR with the usage of the same hardware, only by changing control signals. Examples of CiM-A include Snider logic [88] and Majority-oriented logic [37], whereas examples for CiM-P can be found in Scouting Logic for ReRAM [100], scouting-like CiM for STT-MRAM[45] and Boolean Matrix Multiplication on memristive crossbars [93].

Fig. 2.4 shows a generic architecture using scouting logic [100] to calculate the bit-wise AND/OR operations of two words. Here, the write operation is similar to that of conventional memory write, where each row in the memory is written separately. However, for the read operation, the row with the operands are enabled together via their respective wordlines ($WL_0$ and $WL_1$ in Fig. 2.4), the source-line is conditioned with a small bias voltage and the resulting current sum on the bit-line is compared with a reference to calculate the result of the operation.

By using an appropriate reference current in combination with a current sense amplifier, simple binary operations are realized [100, 45]. We can see in Fig. 2.5 that the different states of activated MTJs influence the overall resistance on the bitline and thus can be compared to a reference between two possible state combinations. In case an OR operation is needed, this reference needs to be between the ’00’ and the ’10’ state, whereas the reference of an AND operation needs to be set between the ’10’ and the ’11’ state. As can be seen in the image, a larger On/Off ratio of the memory cell translates to a lager sense margin between the different states. This results in the comparably larger sense margin for CiM operations in ReRAM architectures compared to STT-MRAM. Different ways of generating these references are possible, e.g. by using a polysilicon resistor or a compound reference [31] using the technology of the NVM in question as shown in Fig. 2.6.

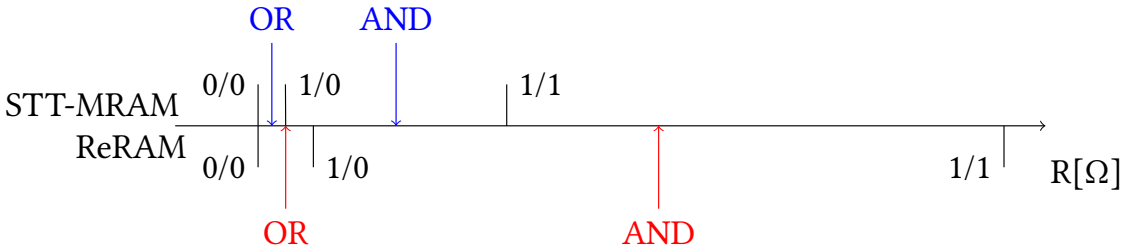



Figure 2.5.: Resistive behavior for multiple active cells in different HRS/LRS state combination and the needed reference resistance for the according binary operations. Relative to 0/0 state for the individual technology. (Adapted from [34])

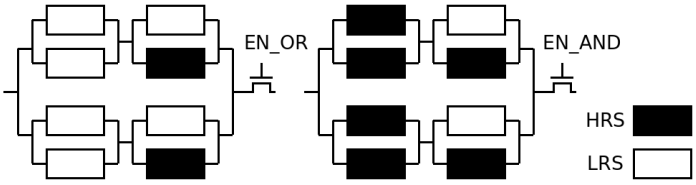
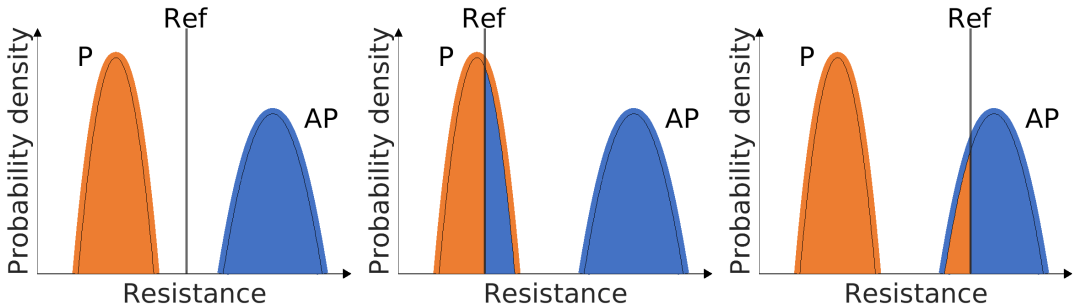



Figure 2.6.: Generic compound references based on resistive memory cells for CiM-OR (left) and CiM-AND (right) operations. (From [34])

## 2.3. Reference Trimming

In resistive architectures, a sense amplifier is typically used to compare a reference resistance to the resistance of the enabled cell on the bit-line [100]. As all MTJs within the same column of the array share the same reference, this reference needs to be able to distinguish the state of all individual cells. On an ideal chip, this would be easy. However, all memory cells are prone to Process Variation (PV) from the manufacturing process, leading to small but impactful differences from one to another. In addition, this is also the case for the reference resistance. The reference resistance may be too close to the P-state resistances (Fig. 2.7b), resulting in an increased number of cells in P-state being erroneously read as an AP-state cell, or the reference resistance may be too close to the AP-state resistances (Fig. 2.7c), resulting in an increased number of AP-state cells being read as P-state cells.




(a) Good setting (b) Many P faults (c) Many AP faults

Figure 2.7.: Reference resistance trimming

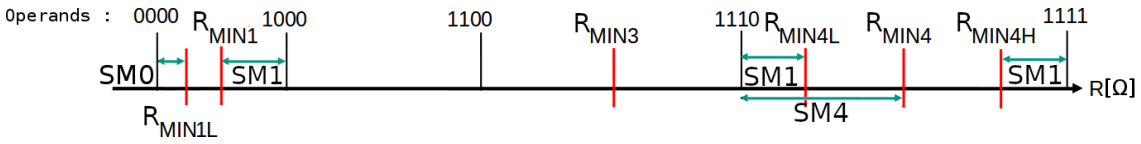



Figure 2.8.: Relative resistive states of four enabled Cells for in-memory computation and the relevant reference resistances with different sense margins (SM) for the evaluated threshold operations. IN addition, non-optimal operation references ($R_{MIN1L}$, $R_{MIN4L}$ and $R_{MIN4H}$) are shown. The operands encode a LRS with 0 and a HRS with 1.

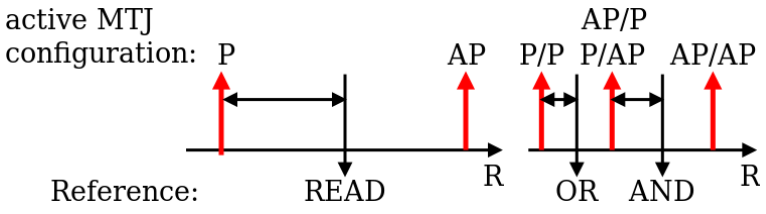
Therefore, multiple selectable resistive elements are placed with the reference resistance to mitigate process variations [5]. An additional circuit allows to add and remove these resistive elements to the sensing path and increase or decrease its resistance. This way, post-manufacturing information can be used to properly set up the reference [103].

This can for example be realized by a serial connection of multiple resistors, which can be individually bypassed (Fig. 2.2). Additionally, tunable current sources can be used to adjust the resulting reference current [5]. These techniques can be directly integrated within the read circuit. The goal is to offer a range of adjustable trim settings to shift the reference resistance (or current) in a way, that the reference resistance (or current) can be placed between the AP-state resistances and the P-state resistances. Ideally, the reference is set between the *lowest AP-state* resistance and the *highest P-state* resistance (Fig. 2.7a) to minimize and eliminate the misclassification of any cell state. However, as these distributions can be influenced by chip-to-chip variations and even memory to memory variation in the same chip (bank to bank in the same memory etc.), the needed trim setting has to be found after the manufacturing of the cells. This can be done through MBIST by using a trim search algorithm. The MBIST can search the lowest trim setting with no failing P-state read operations (P-state trim boundary) and the highest trim setting with no failing AP-state read operation (AP-state trim boundary) and sets the trim setting to the middle in between these two boundaries.

Trimming can also be used to adjust the reference to distinguish between different CiM thresholds. In Figure 2.8 we show the threshold operations of four-bit operations: There are five possible resistance states of the activated cells depending on how many of them are in LRS(0) or HRS(1). Trimming can now be used to shift the reference between these states, e.g. to evaluate if at least one cell is in HRS the reference can be shifted in between "0000" and "1000" to the resistance $R_{MIN1}$. This is also possible for the other threshold operations in our example. In general, this can be extended to any number of operands to implement any threshold operation $MIN^N m$ to check if at least $m$ bit are set out of $N$ activated cells. The only restriction of this generalization is the capability of the sense amplifier to distinguish between the different state combinations.

## 2.4. Manufacturing Defects and Runtime Failures

MTJs are subjected to several different manufacturing defects [21, 11, 74]. A resistive short in the MgO tunnel layer results in a very low resistance of the MTJ, whereas internal MTJ




(a) conventional MTJ read TMR (b) CiM OR/AND TMR

Figure 2.9.: Resulting cell resistances (red) of different single cell activations (P and AP) and different two-bit in-memory cell activations (P/P, AP/P, P/AP and AP/AP). Reference resistances (black) are used to evaluate either the one-bit read or the two-bit operations.

damage can lead to an open connection to the magnetic layers, resulting in a very high MTJ resistance. In addition, the free layer can be permanently fixed to either the AP or the P magnetization state during the magnetization step [99].

The defects are typically modeled by opens and shots between different nodes in the bit-cell. These defects result in faults in both the read and write memory operations. Since the write operation for CiM-based memories is same as that for conventional memory write, the test algorithms for conventional STT-MRAM write would cover the write faults for CiM operations as well. However, for the read-like CiM operation, the faults are different since two (or more) rows are enabled together. Hence, they need to be considered independently from regular read operations.

The opens/shorts in a bit-cell introduced due to defects result in a drift in the sensed current during the read and CiM operations. Fig. 2.9 compares the sense margin of a conventional cell read, which distinguishes between P-state and AP-state, and the sense margins of CiM operations. The sense margin depends on the resistance difference between the $R_{AP}$ and $R_P$ states, which is quantified by the Tunneling Magnetoresistance ratio (TMR), given by

$$TMR = \frac{R_{AP} - R_P}{R_P}. \tag{2.1}$$

However, as shown in Fig. 2.9, the effective TMR available for CiM operations are much lower than that for conventional memory operation. The effective TMR for CiM operation is given by

$$\begin{aligned} TMR_{OR} &= \frac{R_{AP/P} - R_{P/P}}{R_{P/P}}, \\ TMR_{AND} &= \frac{R_{AP/AP} - R_{AP/P}}{R_{AP/P}}, \end{aligned} \tag{2.2}$$

where $TMR_{OR}$ and $TMR_{AND}$ are the TMRs for CiM OR and AND operations, respectively. Substituting

$$R_{AP/AP} = \frac{R_{AP}}{2},\ R_{P/P} = \frac{R_P}{2},\ \text{and}$$
$$R_{AP/P} = \frac{R_{AP} \cdot R_P}{R_{AP} + R_P}$$

in Eq. 2.2 and simplifying, we get

$$TMR_{AND} = \frac{R_{AP} - R_P}{2 \cdot R_P},$$
$$TMR_{OR} = \frac{R_{AP} - R_P}{R_{AP} + R_P} \tag{2.3}$$

Comparing Eq. 2.1 and Eq. 2.3, it is clear that $TMR_{AND}$ and $TMR_{OR}$ are much lower than that of the TMR for a conventional memory read operation. Conventionally, the TMR is given as the relative difference of the two states, so a $TMR = 1.5$ is given as 150%. Hence, the impact of current drift due to defects is more pronounced for CiM operations as compared to conventional memory read operations. This necessitates the testing for CiM-specific faults in addition to the conventional memory read faults to ensure correct functionality of CiM enabled STT-MRAM architectures. In particular, spintronic-based architectures need to be tested more thoroughly, as the typical TMR of their cells are far lower than the On/Off ratio of other resistive memories like ReRAM ($R_{ON}/R_{OFF}$ between 10× to 100×) or PCM ($R_{ON}/R_{OFF}$ between 100× to 1,000×) [102]. Therefore, process variation and device mismatch have a higher impact on the functionality of CiM based on STT compared to other resistive memories, as even the smallest differences can change the circuit behavior.

Also, there are runtime failures during the lifetime of the MTJ. If the supplied write current is too low or not applied for the appropriate duration, it is possible that no magnetic switching can happen in the free layer. This is a *write error* [71].

While reading the cell, a small sensing current is applied. Due to the stochastic switching of the cell, it is possible to accidentally switch the cell during the read operation, destroying the content for the next read operation. This is a *read disturb* [71].

The magnetization of the free layer might also change its state randomly due to thermal noise, resulting in corrupted data. These *retention faults* [20] are highly depended on the device manufacturing parameters of the MTJ, specifically $\Delta$. Similar defects and faults can be observed during the manufacturing and lifetime of ReRAM devices [12].

Usually, Error Correction Codes (ECCs) are used to mitigate the runtime failures as well as manufacturing defects [61, 26]. This can be done on a per-line basis or smaller groups in one row, which allows to perform checks on the data integrity during each read operation. It is also possible to scrub the memory on a regular basis and refresh corrupted data with the correct data. This refresh rate is dependent on the reliability requirements of the memory.

# Part II.

# Contributions

# 3. Conventional STT-MRAM Test Improvements

The focus of this work is on the testability of the CiM-capabilities of the spintronic memory. However, it is important to test the conventional memory behavior of the CiM capable memory. As the later described CiM-related faults need to be detected on top of the known defect behavior, they are going to increase the cost of the already needed memory test procedure. Therefore, this chapter elaborates on the possibility to reduce the test overhead in terms of test time and needed test environment for the conventional memory test, in order to free up test resources for the later needed CiM tests and reduce the total test overhead.

## 3.1. MBIST-supported Trim Adjustment to Compensate Thermal Behavior of MRAM

MTJs, MOS transistors and all circuit components are affected by PV. PV edge fails for memories include marginal cell fails which can escape chip test. Very high test coverage is required to ensure the detection of those weak cells and keep defective parts per million (dppm) rates low. In addition to the existing CMOS failures, MRAM has additional types of defects related to additional fabrication steps for the magnetic layers of the MTJ. To confirm the full functionality of the memory in production chips, functional testing is generally performed during the manufacturing process. However, the functional test only provides limited information of the Chip Under Test (CUT), as it only provides a binary *pass/fail* result, rather than the specific properties such as the exact resistance of the tested memory cell. To improve the operating margin and pass rate, detailed PV analysis and corresponding corrective adjustments are required. Trim circuits are one of the ways to improve the read margin after process completion. Tunable resistances (or current sources) are attached to the trim circuit to enable fine compensation of chip to chip variation [94] or intra-chip variations [85]. Trim circuits enable the automatic post process control [103] of the reference resistance within a certain range to compensate for variations that inherently happen in the memory array and the read circuitry.

In addition to PV, thermal fluctuations have a significant impact on the electrical properties of MTJs [105] and MOS transistors [86] and thus the operation of MRAM. MOS transistors and MTJs have a different thermal behavior of their conductivity. MTJs are also affected differently depending on their magnetic state [8]. This can lead to an increased read failure rate especially at high temperatures. Ideally, any additional failures by thermal behavior should be captured during the manufacturing test, but it is difficult to fully test

all thermal behaviors of a chip because a full temperature test increases the test cost significantly and reduces the life span of the device [10]. Therefore, testing should be done at a low temperature, while still ensuring a low dppm rate across the entire operating temperature range. The presented approach is able to minimize the failure rate over the entire operating temperature range by considering an additional adjustment of the trim setting. Our contributions discussed in this chapter are as follows:

- We created a failure prediction flow for STT-MRAM memory cells based on device models and the functional test results gathered from Memory Built-in Self Test (MBIST) at a low test temperature to predict the expected failure behavior at the highest operation temperature.
- We evaluated the required number of MBIST runs with different trim settings for the prediction to evaluate the quality of our proposed prediction flow.
- We used our prediction flow to choose an optimized trim setting for the tested array which minimizes the failures over the entire operating temperature range of a chip.

The rest of the chapter is organized as follows. Section 3.1.1 discusses related work. Section 3.1.2 compares different STT-MRAM components by their temperature dependency. Afterwards we present our temperature dependent cell behavior prediction in Section 3.1.3 and use this in Section 3.1.4 to adjust the trim setting for failure reduction. Finally, Section 3.1.5 concludes the chapter.

### 3.1.1. Related Work

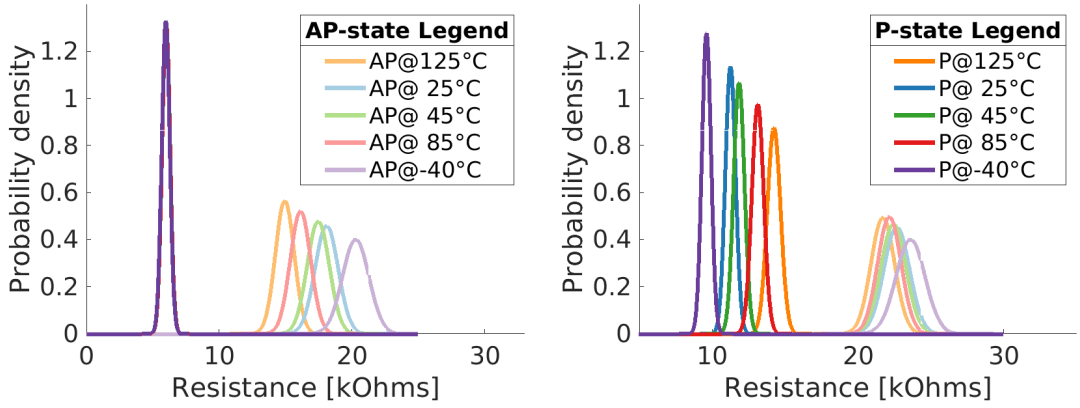
To address process variation and thermal influence on STT-MRAM, different solutions have been proposed. A common approach is to use replicated columns of MTJs as a reference [30]. This allows the reference to follow the thermal behavior of an active MTJ. A negative-temperature-coefficient resistor can be used in series with a P-state MTJ from a single dummy line [13]. Careful tuning of the sensing path allows to generate the proper reference signal even at higher temperatures. A polysilicon resistor-only approach for reference generation is discussed in [5], which uses an additional temperature dependent voltage bias to introduce an offset current in the reference generator. To mitigate the read margin reduction at high temperatures, reducing the word-line signal strength at higher temperature is another possible way [94]. These solutions dynamically adjust the memory behavior to the operating temperature. Shih et al. suggest memory testing up to 125°C in their article [85]. They note that guardbanding can be used in addition to properly trim the circuit at room temperature. However, memory tests at high temperatures are not a preferred option, because it takes additional time and equipment, which significantly increase the test cost.

### 3.1.2. Temperature-dependent Device Behavior

This section elaborates on the thermal behavior of different device types and components commonly found in STT-MRAM, such as bit-cell and reference circuit.

| Parameter | Value |
|---|---|
| VDD | 1.0V |
| Nominal Temperature | 27°C |
| CMOS library | TSMC 40nm low-power |
| MTJ radius | 20 nm |
| Free/Oxide layer thickness | 1.3/1.48 nm |
| RA | 7.5 $\Omega\mu m^2$ |
| TMR @ 0V | > 150% for all temperatures |
| 'AP'/'P' resistance | 19 kΩ/6 kΩ |

Table 3.1.: MTJ parameters and simulation setup




(a) MTJ (b) Full cell (1T1MTJ)

Figure 3.1.: Temperature influence on MTJ and full cell (1T1MTJ) variations

#### 3.1.2.1. Simulation Setup

We performed all our simulations with the Cadence Virtuoso Tool Suite specifically Cadence Spectre for SPICE simulations. Our MTJ model is described in [59] and characterize with silicon data. Yet, our approach is not limited to this specific model and can be used generically. The general simulation setup as well as the specific MTJ device parameters are given in Table 3.1. Our failure prediction flow as well as our reference adjustment flow and their evaluations are performed using MathWorks Matlab.

#### 3.1.2.2. 1T1MTJ cell

The 1-transistor/1-MTJ (1T1MTJ) cell is composed of two devices with opposite thermal behaviors. The NMOS transistor changes its effective threshold voltage under thermal stress, leading to an increased cell resistance. The MTJ has a nearly invariant P-state resistance and a decreasing AP-state resistance for increasing temperatures (Fig. 3.1a). This has an inverse influence of the full cell resistance.

As a result of the different wave function symmetry across the tunneling layer, P-state MTJs have a lower thermal conductance dependency than AP-state MTJs [56], hence the thermal conductance change in P-state 1T1MTJ cells is dominated by the transistor's thermal behavior, whereas that of AP-state cells is dominated by the MTJ (Fig. 3.1b) [8].

#### 3.1.2.3. Polysilicon Reference Resistor

To generate the optimum reference current for the sense amplifier a stably accurate reference resistance setting is necessary. One candidate device in a CMOS process is the polysilicon resistor. Polysilicon resistors are nearly temperature invariant. The small size and parasitic BL resistance tracking benefit make a P-state MTJ a good reference resistance. Additionally, AP-state MTJ offer temperature dependence. However, the poly resistance is not affected by retention faults (accidentally flipping the state). In some applications this makes the polysilicon resistor a better choice for a reference resistor.

### 3.1.3. Trimming-based Failure Prediction for High Temperature

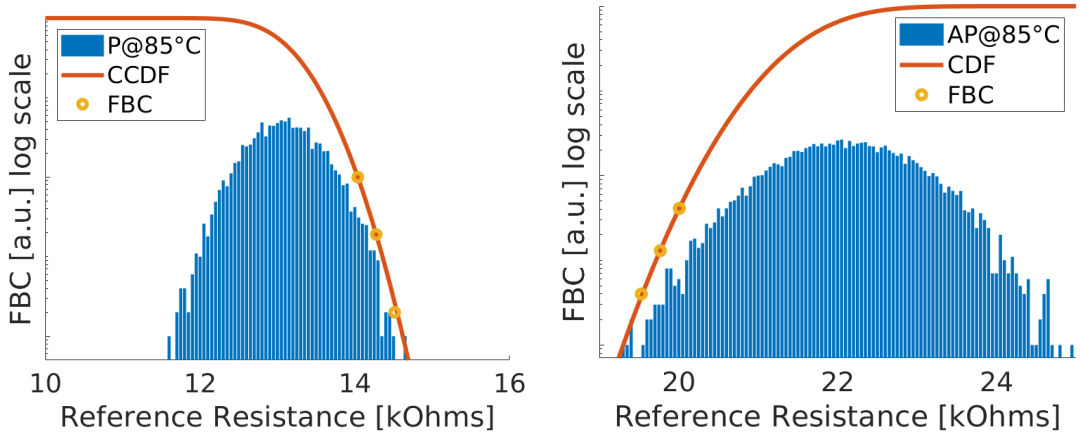
As discussed before, the temperature dependent behavior of the cell differs from that of the polysilicon reference resistance. As a result of this, additional failures may occur at different temperatures. With a properly trimmed reference resistance read failures can be minimized over the full operating temperature range for reasonably sized memory macros. For large sized arrays such as 1Gbit, it is difficult to set a single reference that works properly for the entire array, because of possible cell variations. In this case, we need to set the trim based on each properly sized sub-array or use other supporting techniques such as ECCs or *Repair* by redundant columns or rows.

We want to achieve a low dppm rate over the entire operating temperature range. Yet, it is an expensive option to test the CUT at high temperatures. Therefore, we propose a method to predict the Fail Bit Count (FBC) curve behavior at high temperatures using the FBC information obtained at a lower temperature. The FBC is generated by counting the number of wrongly read bits (read value does not match the expected value) of the whole memory array [103].

As we cannot extrapolate any useful information from zero failures, we use the MBIST-based trim circuit control to force failures at a tested temperature and use the results from this to predict the behavior at high temperatures. The hardware counter used by the MBIST to count failed bits has only a limited width, e.g. 16 bits [104], to keep the test hardware overhead within an margin acceptable to the memory designer. Each trimming step needs to read the entire memory twice, so data gathering is costly unless a skip address mode is used to limit the tested addresses, which reduces the number of tests with the cost of limited data. This can also be used to mitigate the FBC counter size. Also, it motivates us to evaluate the usage of fewer data points to predict the high temperature failure rate, in order to reduce test time and costs. Most of the trim settings do not result in failing bits (hard failure cells are pre-screened), as the trim range is supposed to be set to cover the tails of the P and AP distributions (compare Fig. 3.2). In the following, we will refer to the lower chip test temperature as $T_T$ and the higher target temperature to be predicted as $T_P$.

#### 3.1.3.1. Extracting CUT Variation Map with Trimming

By changing the reference resistance of the STT-MRAM read operation we are influencing the sense margin of the operation. We use this to provoke failures during the chip test. In the following, we assume a lower trim setting results in a lower reference resistance and




(a) P-state cell at 85°C

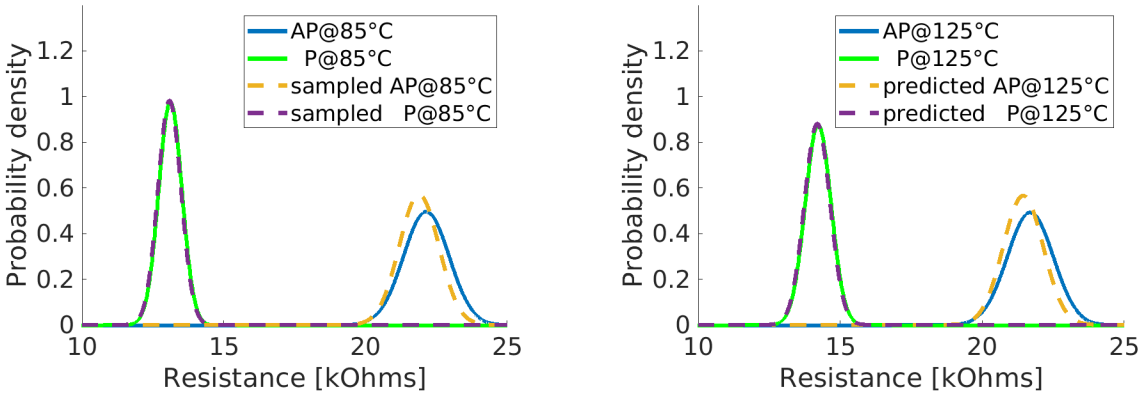
(b) AP-state cell at 85°C

Figure 3.2.: Actual Cell Distribution, FBC at different Trim settings and reconstructed CDF/CCDF.

vice versa, without a loss of generality. The general test data generation is done at $T_T$ with two (N) *pre-selected* consecutive decreasing trim settings $tr_0^P$ to $tr_{N-1}^P$ for reconstructing the P distribution, and increasing trim settings $tr_0^{AP}$ to $tr_{N-1}^{AP}$ for the AP distribution, respectively. The number $N$ of used trim steps is chosen by the desired accuracy vs. the acceptable test time overhead. We evaluate our approach for $N \in [2, 5]$. To find the lowest trim setting $tr_0^P$ without $R_P$ failures and the highest trim setting $tr_0^{AP}$ without $R_{AP}$ failures, "binary-judge-based-search" [103] can be used. The full flow is as follows.

- Use the MBIST to find the trim setting $tr_0^P$ with "binary-judge-based-search" at temperature $T_T$ and with all memory cells in the P-state.
- The number of failures at the trim setting $tr_0^P$ is the FBC a for P-state cells at trim setting $tr_0^P$ [$\text{FBC}_P^{T_T}(tr_0^P)$].
- Repeat the MBIST operation with the next trim setting $tr_1^P$ which yields the resulting FBC of $\text{FBC}_P^{T_T}(tr_1^P)$. This is done for the *selected* trimming steps $tr_0^P$ to $tr_{N-1}^P$ and gives the discrete function $\text{FBC}_P^{T_T}(tr^P)$.
- The same procedure as before is performed for the trim settings $tr_0^{AP}$ to $tr_{N-1}^{AP}$ with the difference, that in this second run the expected cell state is AP during the MBIST operation. This results in $\text{FBC}_{AP}^{T_T}(tr^{AP})$.

The data points of $\text{FBC}_{AP}^{T_T}$ (compare Fig. 3.2b) and $\text{FBC}_P^{T_T}$ (Fig. 3.2a) are now interpreted as the Cumulative Distribution Function (CDF) and a Complementary CDF (CCDF) of a statistical distribution, respectively. If an FBC is capped at the limit of the MBIST counter, the data point is not used, as it is not a valid statistical information. By increasing the fail bit counter and with that the hardware cost, more data points are available. As we see later, this may allow for a more accurate prediction. As transistors and MTJs are typically considered normal distributed, their serially connected resistances also follow




(a) Reconstructed Distribution based on FBC/Simulated Distribution

(b) Predicted Distribution/Simulated Distribution

Figure 3.3.: Reconstructed/Predicted Distributions vs Golden Distributions, based on 3 samples per distribution.

such a distribution. Hence, we fit $FBC_{AP}^{T_T}$ with the simplex algorithm[52] to the CDF $F$ which is given in Eq. 3.1. Similarly, $FBC_{P}^{T_T}$ is fit to a CCDF, which is given by $1 - F$.

$$F(x) = \frac{1}{2}[1 + \text{erf}(\frac{x-\mu}{\sigma\sqrt{2}})], \quad \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{3.1}$$

This results in the ***sampled*** **Normal Distributions** $\bar{\mathcal{N}}_P^{T_T} = \mathcal{N}(\bar{\mu}_P^{T_T}, \bar{\sigma}_P^{T_T})$ (with substituted $\bar{\mu}_P^{T_T}$ for $\mu$ and $\bar{\sigma}_P^{T_T}$ for $\sigma$) and $\bar{\mathcal{N}}_{AP}^{T_T} = \mathcal{N}(\bar{\mu}_{AP}^{T_T}, \bar{\sigma}_{AP}^{T_T})$ (with substituted $\bar{\mu}_{AP}^{T_T}$ for $\mu$ and $\bar{\sigma}_{AP}^{T_T}$ for $\sigma$) (Fig. 3.3a). We named the sampled $(\bar{\mathcal{N}}_P^{T_T}, \bar{\mathcal{N}}_{AP}^{T_T})$ distributions and the later described prediction-based normal distribution with a bar, in contrast to the later described device model-based distributions, which do not have a bar. This distribution information will be used in the next chapter to reconstruct the FBC curve at a higher temperature.

#### 3.1.3.2. Model-based High-Temperature Failure Prediction

Our proposed prediction model is based on the device models of the NMOS access transistor and the MTJ. Therefore, we need the full cell resistance distributions for different temperatures at each cell state. We extract the temperature dependent variation by performing individual Monte-Carlo SPICE simulations for the 1T1MTJ cell in both states (AP and P) at test temperature $T_T$ and our prediction target temperature $T_P$. These simulations provide the four ***modeled*** **Normal Distributions** $\mathcal{N}_P^{T_T}$, $\mathcal{N}_{AP}^{T_T}$, $\mathcal{N}_P^{T_P}$, $\mathcal{N}_{AP}^{T_P}$. We then calculate the linear correlations $k_\mu^P$ and $k_\sigma^P$ between the mean and variation of the distributions for P-state cells at test temperature $T_T$ and the prediction temperature $T_P$. Similarly, we compute the linear correlations $k_\mu^{AP}$ and $k_\sigma^{AP}$ for AP-state cells as given in Eq. 3.2-3.5. Our prediction model is defined by these parameters $k_\mu^P$, $k_\sigma^P$, $k_\mu^{AP}$, $k_\sigma^P$, which represent the linear shift of mean and variation values of the different distributions for the evaluated temperature shift.

$$k_\mu^P = \mu_P^{T_P} / \mu_P^{T_T} \tag{3.2}$$

| corner | added tests per MTJ state | failure rate (sim./pred.) | FBC (1M) | prediction accuracy |
|---|---|---|---|---|
| 85°C sim. | - | 4.780e-12 | 0 | used trim setting |
| 125°C sim.* | - | 1.073e-04 | 112 | given model |
| 125°C pred. | 2 | 6.7296e-05 | 70 | 62.5% |
| 125°C pred. | 3 | 1.0543e-04 | 110 | 98.21% |
| 125°C pred. | 4 | 1.0543e-04 | 110 | 98.21% |
| 125°C pred. | 5 | 1.0602e-04 | 111 | 99.11% |

Table 3.2.: Simulated and predicted corners with failure rates, corresponding fail bit counts based ob the failure rate for a 1 MBit array and prediction accuracy of FBC compared to the golden result from the model, *golden

$$k_{\mu}^{AP} = \mu_{AP}^{T_P} / \mu_{AP}^{T_T} \tag{3.3}$$

$$k_{\sigma}^{P} = \sigma_{T_P}^{P} / \sigma_{T_T}^{P} \tag{3.4}$$

$$k_{\sigma}^{AP} = \sigma_{T_P}^{AP} / \sigma_{T_T}^{AP} \tag{3.5}$$

To predict the resistance distribution of a CUT at high temperature we scale the sampled mean and variance (as described in Section 3.1.3.1) by these factors and get the ***predicted* Normal Distributions** $\bar{\mathcal{N}}_{P}^{T_P}$ and $\bar{\mathcal{N}}_{AP}^{T_P}$ as given in Eq. 3.6 and 3.7.

$$\bar{\mathcal{N}}_{P}^{T_P} = \mathcal{N}(k_{\mu}^{P} \times \bar{\mu}_{P}^{T_T}, k_{\sigma}^{P} \times \bar{\sigma}_{P}^{T_T}) \tag{3.6}$$

$$\bar{\mathcal{N}}_{AP}^{T_P} = \mathcal{N}(k_{\mu}^{AP} \times \bar{\mu}_{AP}^{T_T}, k_{\sigma}^{AP} \times \bar{\sigma}_{AP}^{T_T}) \tag{3.7}$$

The predicted failure rate $P_{fail}$ for a reference resistance $R_{ref}$ is then calculated as given in Eq. 3.8, with the help of the CDFs $\bar{F}_{P}^{T_P}$ and $\bar{F}_{AP}^{T_P}$ of $\bar{\mathcal{N}}_{P}^{T_P}$ and $\bar{\mathcal{N}}_{AP}^{T_P}$, respectively.

$$P_{fail} = \bar{F}_{P}^{T_P}(R_{ref}) + (1 - \bar{F}_{AP}^{T_P}(R_{ref})) \tag{3.8}$$

Our full failure prediction flow is therefore as follows.

- Generate $\mathrm{FBC}_{AP}^{T_T}$ and $\mathrm{FBC}_{P}^{T_T}$ as described in Section 3.1.3.1.
- Fit the two FBCs to two individual Normal Distributions (Eq. 3.1) and shift them according to our model parameters to approximate the distribution at the prediction temperature $T_P$ (Eq. 3.6,3.7).
- Calculate the expected failure rate by using the CDFs of the predicted distributions (Eq. 3.8).

#### 3.1.3.3. Failure Extrapolation Results

To evaluate the accuracy of our prediction, we performed Monte-Carlo simulations at 85°C and 125°C. We used the simulation results to build our prediction model. Additionally, we discretely sampled the distributions at 85°C with 32 steps between the tails of the

P-distribution and the tail of the AP-distribution to model the possible trim settings. Only two to five (depending on the acceptable number of different trim settings) of these data points are used to generate the extrapolated failure map, to limit the sampling test time. This failure map is used to predict the failure rate at 125°C. From our initial simulation for generating the prediction model we already have the golden result of this prediction. We can now compare our failure prediction result to this golden simulation result. The outcome of this comparison is shown in Table 3.2. We chose a reference resistance setting based on the previously described PDF intersection which does not produce any failure in a 1MB array at 85°C and used it for failure evaluation at different temperatures. A reference setting optimized at 85°C without an additional adjustment for temperature effects showed noticeable failure counts in our golden simulation at 125°C. Using 5 trim settings, it is possible to predict this golden failure behavior with an accuracy of 99.11%. Reducing the used trim setting reduces the prediction accuracy. There are not as many useful data points for the P-distribution as for the AP-distribution because the resistance variation of P-state cells is less than the variation of AP-state cells. Therefore, the P-distribution modeling does not improve with more than 3 trim settings, whereas the AP distribution benefits from up to 5 trim settings.

### 3.1.4. High Temperature Trim-Adjustment

The common practice is to find the trim range with no FBC based on the MBIST data at the test temperature and use the trim setting in the middle of that range. However, due to asymmetric thermal behavior of cells, this does not lead to the minimum FBC at higher operating temperatures, as it is evident in our results (Table. 3.3), leading to higher dppm. In contrast, we use our failure rate prediction to compute a reference resistance which minimizes the failure rate over the entire operating temperature range of the CUT. Based on our observation, the STT-MRAM read margin is smaller at higher temperatures, therefore we try to minimize the read failure rate at the highest operating temperature.

To calculate the adjusted reference resistance, we use the two Probability Density Functions (PDFs) $\bar{f}_P^{T_P}(x)=f(x, \bar{\mu}_P^{T_P}, \bar{\sigma}_P^{T_P})$ and $\bar{f}_{AP}^{T_P}(x)=\bar{f}_{AP}^{T_P}(x, \bar{\mu}_{AP}^{T_P}, \bar{\sigma}_{AP}^{T_P})$ of the predicted distributions $\bar{\mathcal{N}}_P^{T_P}$ and $\bar{\mathcal{N}}_{AP}^{T_P}$ with $f$ defined according to Eq. 3.9.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.9}$$

Our adjusted trim setting is given by the intersection $\bar{x}$ of $\bar{f}_P^{T_P}(\bar{x})$ and $\bar{f}_{AP}^{T_P}(\bar{x})$, which a good approximation to minimize the failure rate for P and AP-distributions. The resolution of the final reference resistance depends on the trim resolution. This allows an optimal reference shift for high temperature operation.

#### 3.1.4.1. Adjustment Flow

The full flow of our trim adjustment is depicted in Fig. 3.4 with test temperature $T_T = 85°C$ and the predicted (higher operating) temperature $T_P = 125°C$.

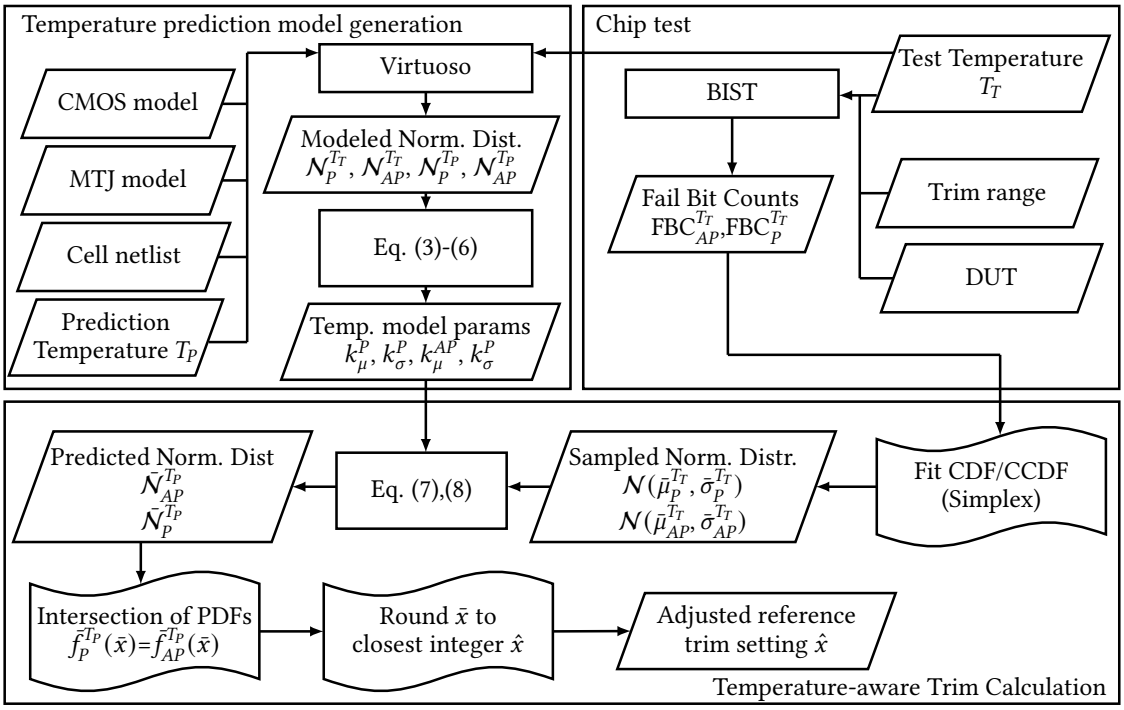



Figure 3.4.: Trim setting calculation flow overview.

- Use the MTJ and MOS transistor models to generate the golden model parameters $k_\mu^P$, $k_\mu^{AP}$, $k_\sigma^P$ and $k_\sigma^{AP}$ from the *modeled* Cell Resistance *Normal distributions* $\mathcal{N}_P^{85}$, $\mathcal{N}_{AP}^{85}$, $\mathcal{N}_P^{125}$, $\mathcal{N}_{AP}^{125}$ (Section 3.1.3.2).

- Generate the variation map at test temperature (85°C): Find the trim settings with the lowest failures using binary-search and use two (or more) consecutive trim settings for each, AP and P, to get the fail bit count vectors $FBC_{AP}^{85}$ and $FBC_P^{85}$.

- Fit $FBC_{AP}^{85}$ and $FBC_P^{85}$ to the respective *sampled Normal Distributions* $\bar{\mathcal{N}}_{AP}^{85}$ and $\bar{\mathcal{N}}_P^{85}$ using the Simplex algorithm (Section 3.1.3.1).

- Predict the temperature dependent resistance shift of the distributions by calculating the *predicted Normal distributions* $\bar{\mathcal{N}}_{AP}^{125}$ and $\bar{\mathcal{N}}_P^{125}$ using Eq. 3.6 and Eq. 3.7 (Section 3.1.3.2).

- Calculate the adjusted reference resistance $\bar{x}$ by calculating the intersection $\bar{f}_P^{125}(\bar{x})=\bar{f}_{AP}^{125}(\bar{x})$ of the PDFs of $\bar{\mathcal{N}}_{AP}^{125}$ and $\bar{\mathcal{N}}_P^{125}$ as described above.

- Round the result to the closest integer $\hat{x}$ to get the adjusted trim setting.

- Set the calculated adjusted trim setting in the CUT

| Reference | Temperature | failure rate | FBC (1M) | Total failures (FoM) |
|---|---|---|---|---|
| fixed nominal reference shift | -40°C | 2.63e-08 | 0 | |
| | 25°C | 1.88e-07 | 0 | |
| | 85°C | 3.86e-07 | 0 | |
| | 125°C | 6.79e-06 | 7 | 7 |
| Ref85 (no shift) | -40°C | 3.91e-13 | 0 | |
| | 25°C | 1.04e-12 | 0 | |
| | 85°C | 4.78e-12 | 0 | |
| | 125°C | 1.07e-04 | 112 | 112 |
| Ref85 (125°C adjusted) [ours] | -40°C | 1.06e-10 | 0 | |
| | 25°C | 5.02e-10 | 0 | |
| | 85°C | 6.43e-10 | 0 | |
| | 125°C | 5.25e-08 | 0 | 0 |

Table 3.3.: Results based on reference calculation at 85°C. First, using the middle of the nominal resistances of AP/P and shifting the reference by a fixed offset, second using PDF intersection without any temperature adjustment and third using our temperature adjusted reference (2 trim settings per MTJ state)

#### 3.1.4.2. Results

To evaluate our adjustment flow, we performed Monte-Carlo simulations for wide temperatures ranging from -40°C to 125°C. Each temperature/cell-state combination was evaluated with 10k simulation runs. We calculated the failure rate with respect to an optimal reference resistance, which is not bound to any trimming resolution to evaluate the quality of our prediction. To model the process variation of the periphery we included a variation of 1.7% to the reference resistance [85]. Our baseline is a pre-calculated fixed shift of the reference trim setting, which is generated by the golden simulation results. The trim is set in the middle of the failure boundaries, once for 85°C and once for 125°C. The evaluation is done with our simulated test array by finding the trim setting in the middle of the failure boundaries at 85°C and applying the fixed nominal shift. Using our prediction approach without shifting the trim setting has a large impact on the failure rate over the whole temperature range. The adjusted reference resistance trim setting for 125°C was predicted based on 85°C data and calculated according to our adjustment flow and results are shown in Table 3.3. With this, we were able to reduce the number of total defects from 112 to zero, while reducing the highest failure rate from 1.07e-04 to 5.25e-08. We can see that a reference setting based on 85°C data may still cause a significant number of failures at 125°C. At the same time, an optimal trim setting for 125°C may cause some failures at low temperatures, as indicated by the increased failure rate for e.g. the *Ref85* reference with no shift at -40°C compared to the *Ref85* reference with the 125°C adjustment.

If there is a fail budget that we can use to correct failures in a later step, like ECC, we can use this budget with our flow to expand the operating temperature range. In this case we can use the same test data from 85°C to adjust the final reference resistance that

can minimize the failures over the entire temperature range. As long as the number of failures is lower than the fail budget of the ECC we can make the chip operable for the evaluated temperature range. To put these values in perspective, we calculate the FBC of a 1MB array for each simulation corner and give the summed FBC over all four corners as a figure of merit (FoM).

### 3.1.5. Conclusion

Thermal variation has a more severe effect on the read failure rate of STT-MRAM compared to conventional CMOS-based memories. It is beneficial to know the memory resistance distribution of each state for a chip with respect to the entire operating temperature range. We proposed a statistical flow to predict the high temperature failure behavior of a chip with low temperature MBIST data by the means of a trimming circuitry. The presented simulation data has proven that our method can achieve a very high prediction accuracy. For the highest number of different BIST runs, the accuracy is up to a 99.11% accuracy. This prediction can then be used to adjust the trim setting for high temperature operation and minimize the dppm rate, without executing a cost intensive, high temperature test of the STT-MRAM array.

## 3.2. MBIST-based Trim-Search Test Time Reduction for STT-MRAM

Functional testing is performed as a screening process to detect faulty cells within the memory array and sort out unrepairable chips which have a number of defects above the repairable budget of a chip. To minimize the influence of PV on the yield, additional circuitry in form of trim circuits can be used to allow the fine-tuning of the memory read circuitry post-manufacturing [42, 25, 5]. Configurable resistances or current sources allow to adjust the reference resistances needed for memory operations. This can be used to mitigate chip-to-chip[94] or intra-chip[85] variations, which may otherwise lead to failures and render an entire chip defective.

To search for the optimal trim setting, several steps are needed to choose the trim setting with the least number of failures. If the failure rate is too high, the chip is considered failing or unrepairable, else additional repair mechanisms, like redundant memory rows/columns or ECCs [61] may repair or correct the failing bits. This test can be as simple as testing the entire memory with all possible trim settings, resulting in an extremely expensive procedure. However, all known trim search approaches base their method on testing the entire memory array exhaustively and multiple times. Faster trim searches like the binary-search based "Binary-Judge" trim search [103] can be used with MBIST to reduce the number of times the entire array has to be tested during the trim search.

In this chapter we propose to skip certain word addresses during the individual trim setting test, resulting in reduced information about the memory array for the final trim-search calibration and improving the individual trim test time and thus the overall trim search time significantly. Word address skipping can be achieved by modifying the address generation of an MBIST and is thus a low-cost modification. The address skipping trim

test is then used to create a progressively decreasing address skipping distance algorithm, which tries to find the trim boundaries with a large number of skipped addresses. In case no valid trim setting can be found, the algorithm increases the test coverage by reducing the number of skipped addresses and repeating the procedure. This flow guarantees finding a valid trim setting, as previous exhaustive test methods do, with significantly less run time and only minimal hardware overhead.

Our contributions discussed in this chapter are as follows:

- We developed a framework to evaluate address skipping for MBIST-based trim searching.
- We analyzed the impact of skipping addresses during the trim search test.
- We propose a progressive trim search algorithm using multiple address skipping trim search runs to optimize the overall run time.

The reminder of the chapter is structured as follows: Section 3.2 discusses related work. In Section 3.2.2 we describe the main idea of address skipping and the proposed progressive algorithm, followed by the results of our simulation based experimental results in Section 3.2.3. Section 3.2.4 concludes the chapter.

## 3.2.1. Related Work

Obtaining a suitable trim setting without spending too much time and hardware resources was explored in recent publications: Instead of testing the trim settings high to low (or the other way around) in a linear fashion to find the P-state and AP-state trim boundaries, a binary search approach was proposed in [103]. This allows to reduce the trim search time spent on the memory test for each trim setting from $O(k * N)$ of the linear search to $O(log(k) * N)$, with the number of trim steps $k$ and an array size of $N$. The result of this trim setting calculation can further be adjusted to compensate for temperature behavior by directly changing the trim setting by a certain offset.

## 3.2.2. Trim-Search using Address Skipping

In this section, we first describe the overall memory architecture, elaborate on the MBIST capabilities, and give a brief description of possible hardware implementation details before going into the details of the proposed progressive test flow.

### 3.2.2.1. MBIST Capabilities

Typically, the memory operations performed by an MBIST circuit are applied on the entire memory array. This includes writing all cells consecutively to 0 or 1 (W0/W1) and reading all data from the memory and compare it with the previously written value (R0/R1). This is done using an address register which is e.g. initialized to zero at the beginning and increased by 1 for each individual word-based operation of the MBIST, until the maximum address value has been reached. If a bit in a read word is different than the expected value, it will increment the *Fail Bit Counter* (FBC) to accumulate the *all fail count* in the array. To allow individual SA trimming, fail bits can be identified based on the column address.

#### 3.2.2.2. Skip Test Procedure

During the conventional trim setting evaluation procedure, all the memory cells are tested for a given trim setting, in order to find whether their resistive states cross the trim boundary. Therefore, by sweeping the trim setting linearly the distributions of P and AP resistances of memory cells in the array are obtained. The tails of these distributions are used for setting the trim boundaries. Instead of doing this process exhaustively for all memory cells in the array, such distributions could be *sampled* from a subset of the population (memory cells) to obtain the majority behavior of the distributions. To ensure the validity of the resulting trim boundary from the sampled information, it is again tested on the full array at the specific trim setting, as will be explained later in detail. This is the main idea behind the skip test method and can help to significantly reduce the test time (the number of cells tested) for trim search.

The address skipping trim search can be performed with different algorithms, e.g. linear search or binary search algorithms like "Binary-Judge" [103]. The key difference is the replacement of the address sequence from the full address to interleaved (skipped) addressing with a user-defined specific interval. This means that all write, read and compare operations are only performed on a fixed sub-array instead of the entire array.

#### 3.2.2.3. Progressive Test Flow

The trim-search test is a prerequisite test required to be completed before the actual full functional test. Due to the relatively slow operation speed of an MRAM, a long trim search can be an obstacle in meeting the test budget. However, if the skip trim search results in an incorrect trim setting to be used for the entire array, it will cause extra failures. To improve the manufacturability, ECC and repair solutions are often used in MRAM to mitigate chip failures which would result from only a few failing bits. It is possible to use some of the repair budget to increase the read margin by ignoring some of the read failures of the tail bits. The proposed progressively decreasing address skip test flow using the address skip $s$ of the MBIST, as shown in Fig. 3.5, can improve the read margin further with the usage of this extra repair budget.

The test flow is as follows: First, all chips are screened for hard P-failures (cells that read as P-state irrespective of the trim setting) by writing the memory array in P-state and testing the entire memory in P state at the highest trim setting. Defective chips that produce an intolerable high number of failures are discarded without further test. Once a chip passes the P-state hard fail screening, addresses with a large skip size, for example $s$=256, will be selected. This means every $256^{th}$ address is tested, and the rest of the address bits won't be accessed. This is used to perform a trim search for the P-state boundary with the address skipping enabled. Once the trim boundary is found, we need to confirm if the current boundary can cause fails by untested tail bits. The full memory ($s$ = 1) is tested at the searched trim setting to confirm if any bit fails at the current trim boundary. In this work, we experimentally obtained the initial skip $s$=256 from our simulation as the highest acceptable initial skip to perform the skip test while keeping the number of needed confirmation tests low. If the tested trim setting causes failures above the tolerable failure threshold th_fail for the entire array, the address skip is reduced based on the stepsize by

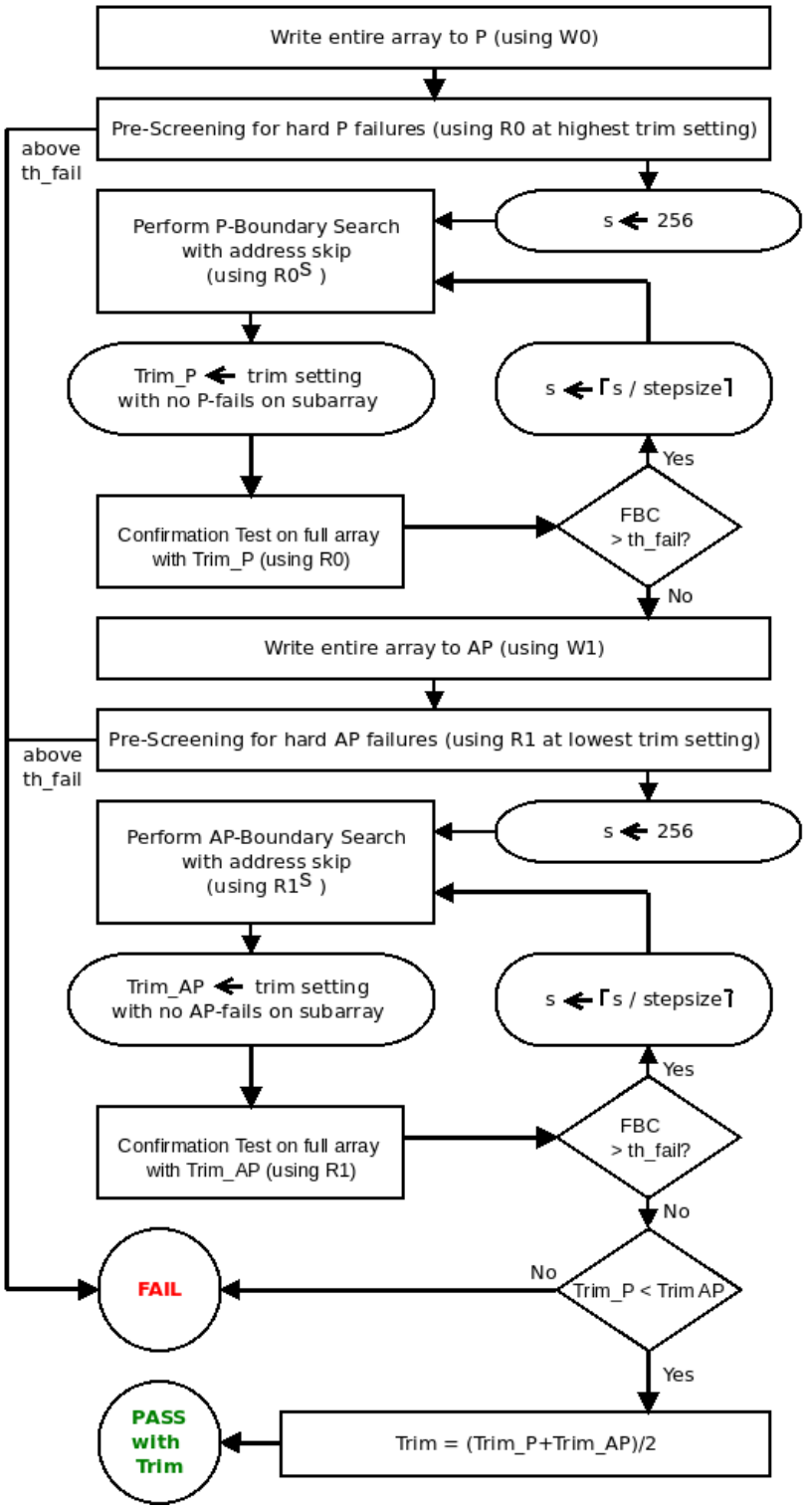



Figure 3.5.: Progressive Test Flow using pre-screening to sort out unrecoverable chips. Start with maximum skip to find fail boundaries Trim_P/Trim_AP and confirm them on the entire array. Use fail threshold *th_fail* for acceptable failure rate and repeat until the failure threshold is not surpassed.

dividing the current skip through the stepsize and the boundary search test is repeated with the new skip. Testing more cells allows to sample more from the distribution of resistive states and find the trim boundaries (tails of respective P and AP distributions) more accurately. This procedure will be repeated until we found the trim boundary with a failure count of the entire array with less than the repairable bit budget. Afterwards, the entire memory is written to AP, checked for hard AP-failures at the lowest trim setting and the chip is discarded if too many failures are present. Then the AP-boundary is searched in the same way as the P-boundary above. After both P and AP-boundary have been found, they are compared to check if there is an overlap of P and AP distributions (Trim_P < Trim_AP). If there is no overlap, a value between the P and AP-boundaries is used as the final trim setting. Usually, the value in the middle between the P and the AP-boundaries is used as the final trim setting, but it can also be biased towards the P-boundary to accommodate for temperature effects [70], which reduces the resistance of AP-state MTJs for higher temperatures and thus reduces the TMR and shifts the ideal trim setting [56]. If the trim setting of the P-boundary is larger than the trim setting for the AP-boundary, there are no valid trim settings to properly distinguish between the two state distributions, and thus the chip is sorted out.

The flow allows for multiple optimizations, depending on the available hardware resources of the MBIST and array (see Section 3.2.3.3). If multiple individual address skipping tests with different skip settings result in the same trim setting, the full array test result can be cached, and it is not necessary to perform the boundary confirmation test again. By using th_fail = 0, this flow also ensures that no chip that would fail a conventional test can escape due to the address skipping. There are no test escapes caused by our sampling-based approach.

#### 3.2.2.4. MBIST Adjustments for Address Skipping

To allow the MBIST to perform address skipping, the offset between consecutive addresses needs to be configurable. This can be done by using an MBIST with the capability to use an offset for the next address calculation, instead of a static address increase. To test individual SAs, it is also necessary to specify a starting address, used as the initial address register value, a final address which will stop the test once the address register reaches, and a mask register which is used to mask failures from SAs currently not counted in the FBC. This way, it is possible to only write, read, and evaluate cells which are sensed by a specific SA and choose the number of cells which are considered during the test.

### 3.2.3. Evaluation

In this section, we first have a look at the implications of the address skipping during the trim search. Then, we use this data to provide more information about the test time reduction.

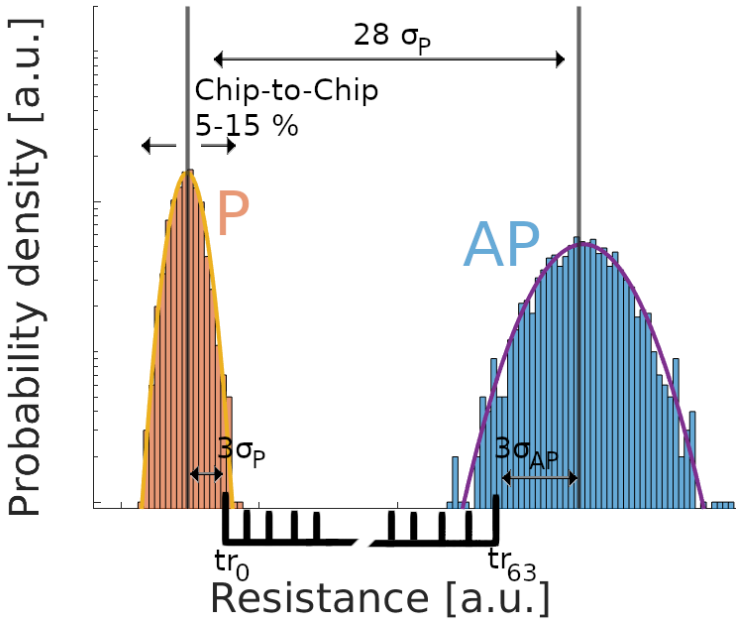



Figure 3.6.: Cell level distributions, trim setting assumptions and chip-to-chip variation using mean distribution shift for an example distribution of a single chip.

#### 3.2.3.1. Simulation Setup

We use a reference sense scheme as described in [13] with a trimmable resistor and a column of reference cells which are used based on the enabled data cells. The temporarily generated reference resistance (or current) is then compared to the accessed cell resistance (or current) using a pre-charge sense amplifier. The result is then used as the content of the cell. A local address decoder enables the wordlines of the read cells and multiplexes the correct bitline to the SA, where the accessed cell resistance (or current) is compared to the reference resistance (or current) value based on the enabled reference cell and the trimmed resistor.

The specific memory architecture used in our evaluation is similar to the one shown in Fig. 3.7. The core memory array is organized in 256 rows of 512 data columns and 16 reference columns. Each SA is responsible for 32 data columns which are connected via a 32:1 multiplexer. This is replicated 8 times resulting in a 1Mbit memory array with 8kbit 1T1MTJ cells and 256 reference bits per SA. The general array organization is based on the memory architecture presented in [27]. The assumed trim resolution is 6 bit, resulting in 64 available trim settings.

We performed a Monte-Carlo Analysis with 10K simulated 1Mbit chips. This simulation was repeated with differently assumed chip-to-chip variation of the memory array. The chip-to-chip variation is simulated by varying the median cell resistance for each individual chip based on a corresponding normal distribution with a variation of 5%, 10% and 15% as depicted in Fig. 3.6. The figure also shows an exemplary local cell variation. The individual intra-chip cell variation is 6.95% of the nominal mean cell resistance to achieve the 28$\sigma$ P to AP offset reported in [72] with a TMR at 25°C of 195% for the nominal case. This model has been chosen to reflect a currently achievable MTJ manufacturing process flow and is

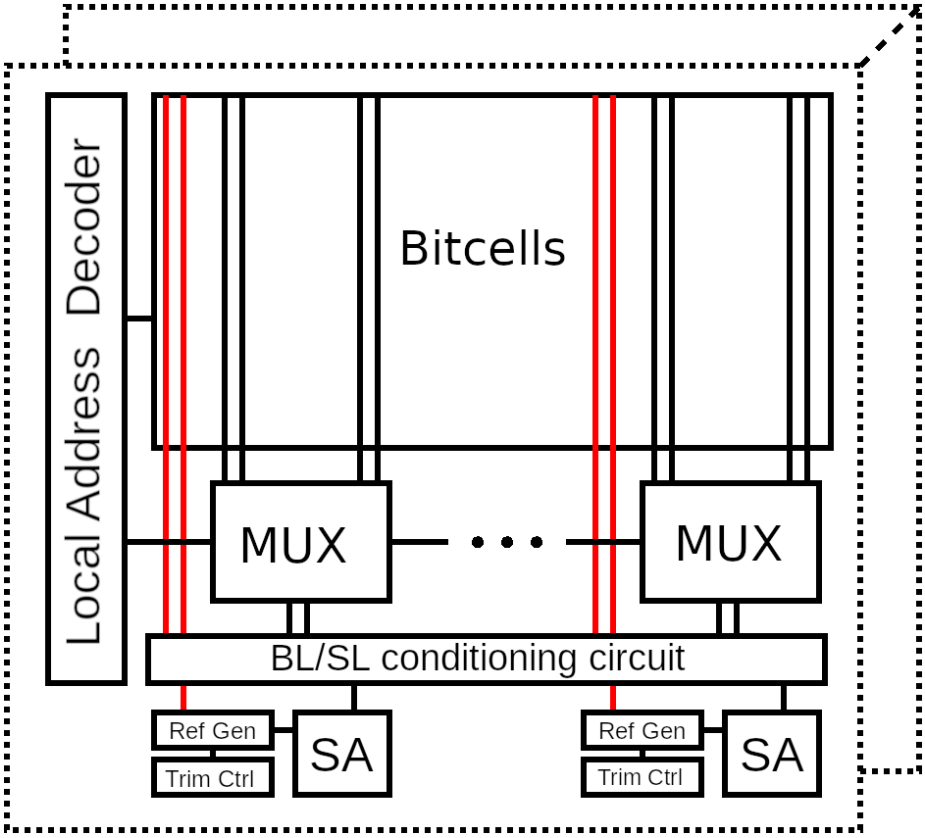



Figure 3.7.: Memory organization of the evaluated STT-MRAM.

used to generate the cell array for each simulated chip. The used failure threshold *th_fail* is 0, resulting in chips being considered failing if at least one cell failure is present. A data cell is considered failing, if the resistive sum of the trim resistance and the corresponding reference cell is lower (higher) than the cell in P-state (AP-state). Increasing *th_fail* can lead to a reduced test time, on the cost of a possibly higher error rate of the resulting trim setting, which in turn can be mitigated by the remaining repair budget.

#### 3.2.3.2. Progressive Test Flow Results

The next step is to gradually decrease the address skipping and test more cells in the chips, for better sampling of the cell distributions and finding the trim boundaries more accurately, as described in Section 3.2.2.3. This is performed individually for the P-boundary and the AP-boundary.

We evaluate the progressive skip address flow on a population of 10K chips, with the goal of setting all of them to a valid reference trim setting for each individual SA. Fig. 3.8 shows the accumulated time spent using the R0/R1 operation during the trim search for every SA in each chip. If a chip has set its trim configuration to a valid setting during the trim search, it is considered good. In case the result of the skip mode trim search does not yield a trim setting with the resulting failures below the chip correction budget the chip is sorted out. Therefore, all 10K chips are first pre-screened for hard P-fails (and sorted out, if too many of them are present) by testing all cells in P state with the highest trim setting. Then the search starts with the largest address skip setting. The remaining SAs of a chip which had no valid trim setting are tested again using a smaller skip, resulting in additional, longer tests. After writing the entire chip to AP and performing the pre-sceen for hard AP-failures, the boundary search is performed for AP. The progressive skip test flow guarantees that only for such chips with skewed distribution a more thorough trim search is performed while for the majority of the memory, a quick trim search with larger skipping is sufficient.

As the memory operations are the main contributors to the trim search time (compared to e.g. internal MBIST calculations), we evaluate the runtime improvement of the progressive flow based on the reduced memory cell tests. The pre-screening for hard failures can be achieved by writing and reading the memory twice which yields the pre-screening time

$$T_{pre} = t_{W0} + t_{R0} + t_{W1} + t_{R1}.$$

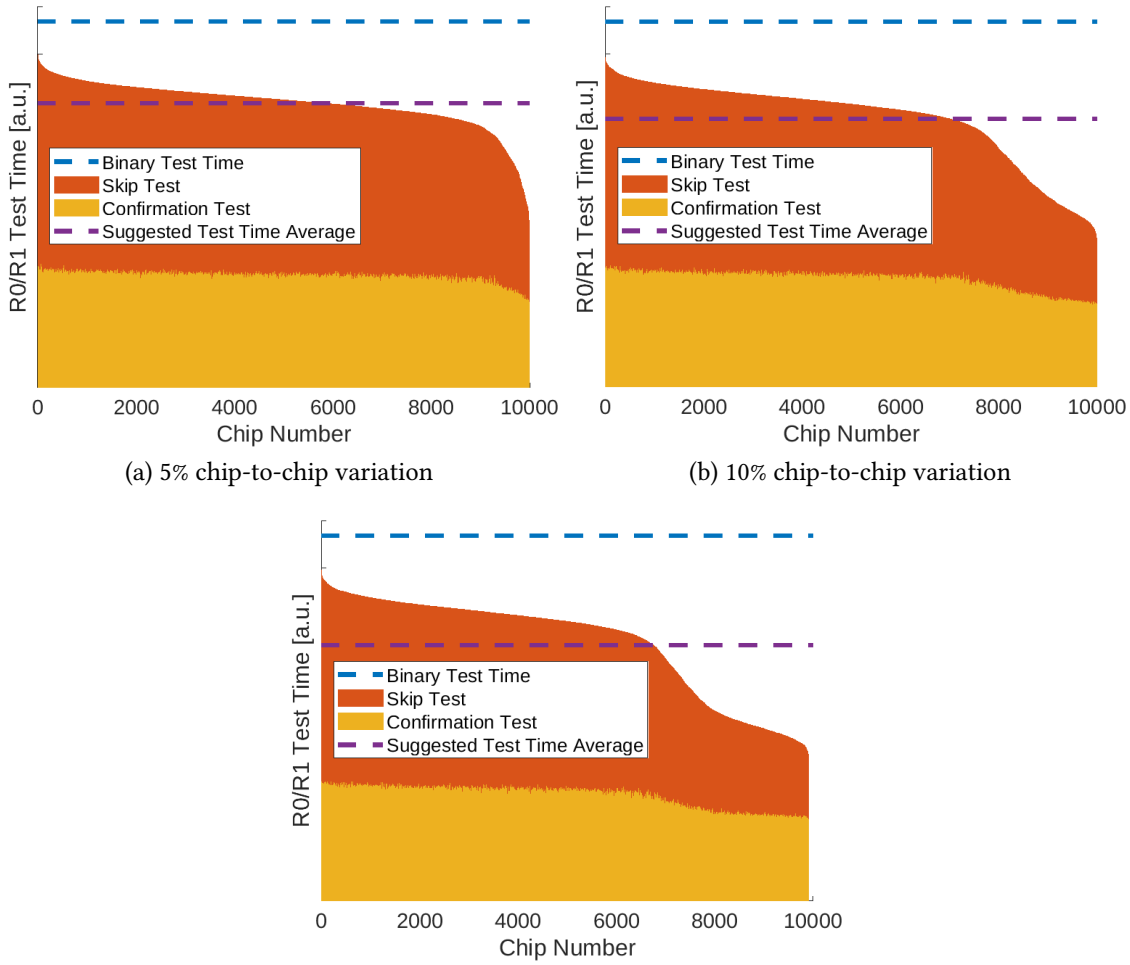
The pre-sceening for hard AP-failures (W1, R1) however is only needed if the chip passed the pre-screening for the hard P-failures. The worst-case runtime of a single linear trim search in this metric is

$$T_{lin} = T_{pre} + (k * t_{R0}) + (k * t_{R1})$$

for trim step size $k$ and the time to perform the memory test operations $t_{W0}, t_{R0}, t_{W1}, t_{R1}$ on the full memory array. This changes to

$$T_{lin,skip} = T_{pre} + \sum_{s \in S}(o_s^P * k * t_{R0}/s) + \sum_{s \in S}(o_s^{AP} k * t_{R1}/s)$$

by using the address skippings $S$, which are restricted to positive powers of two. From our simulation we get $o_s^P$ and $o_s^{AP} \in \{0, 1\}$ per tested array, indicating if a searched trim




(a) 5% chip-to-chip variation

(b) 10% chip-to-chip variation

(c) 15% chip-to-chip variation

Figure 3.8.: Total read time spent during test for each chip. The chip number is the sorted result by total test time per chip and separated colors indicate the time spend in skip tests and the confirmation tests needed to approve a skip test result as a trim boundary. Performed using the presented skip flow with binary search and a stepsize of 16. Binary test time (without using address skipping) is shown above with the blue dotted line for reference. Different test times are due to PV.

| Chip-to-Chip variation | Linear [a.u.] | stepsize=8 Linear +Skip [a.u.] | stepsize=8 Reduction | stepsize=16 Linear +Skip [a.u.] | stepsize=16 Reduction | Binary [a.u.] | stepsize=8 Binary +Skip [a.u.] | stepsize=8 Reduction | stepsize=16 **Binary +Skip [a.u.]** | stepsize=16 **Reduction** |
|---|---|---|---|---|---|---|---|---|---|---|
| 5% | 1.64e8 | 1.08e8 | 34.2% | 8.20e7 | 50.0% | 1.56e7 | 1.55e7 | 0.3% | **1.22e7** | **21.9%** |
| 10% | 1.64e8 | 1.01e8 | 38.6% | 7.65e7 | 53.4% | 1.56e7 | 1.47e7 | 5.9% | **1.15e7** | **26.2%** |
| 15 % | 1.63e8 | 9.49e7 | 41.9% | 7.22e7 | 55.8% | 1.55e7 | 1.39e7 | 10.2% | **1.10e7** | **29.3%** |

Table 3.4.: Total test time reduction (including write operations and hard fail pre-screening) using the progressive address skip trim search with different progressive offsets (*stepsize*) based on the linear and binary search algorithms.

boundary at a given skip setting passed the test for the entire array. $\sum_{s\in S}$ sums up the time of each skip setting $s$ from all used skip settings $S$. If the test passed, all $o$ with a smaller $s$-index will be set to zero to exclude the tests from the test time. E.g., using the flow with an initial skip setting $s$=256 and a stepsize of 16, all *possible* skip tests considered during the search are S={256, 16, 1}. If a SA has not found its P-boundary using $s$=256, but with $s$=16, the corresponding $o_s^P$=[1 1 0], indicating that the first two skip tests have been performed, but the last ($s$=1) was not necessary. In case the sampled array of a SA has only a few outliers, they are very hard to find with the skip test and will introduce a test overhead. However, this overhead is small compared to the test time reduction of the arrays with no outliers, which can perform their boundary searches with a high address skip. The same calculation can be done for a binary-search based approach with

$$T_{binary} = T_{pre} + (log_2(k) * t_{R0}) + (log_2(k) * t_{R1})$$

$$T_{binary,skip} = T_{pre} + \sum_{s\in S}(o_s^P * log_2(k) * t_{R0}/s) + \sum_{s\in S}(o_s^{AP} * log_2(k) * t_{R1}/s)$$

To evaluate the benefits of the progressive test flow, we calculate the cumulative test time of all simulated 10K tested chips with their individual trimmable SAs using the test time of each chip shown in Fig. 3.8, which gives $\sum_{s\in S}(o_s^P * log_2(k) * t_{R0}/s) + \sum_{s\in S}(o_s^{AP} * log_2(k) * t_{R1}/s)$ and the cumulative confirmation test times for each chip for the equations above. The results, including the pre-screen times to reflect the entire test time, are shown in Table 3.4. We assumed $k = 64$ for a 6-bit trim circuit, $t_{W0}, t_{W1} = 10$, $t_{R0}, t_{R1} = 1$, meaning that write operations are ten times slower than the read operations[72]. As more of the chips can use a large address skip for their SAs in case of a greater chip-to-chip variation, (compare Fig. 3.8) the overall improvement in test time is significant. When using address skipping in a linear trim search, the overall reduction in cumulative test time is above 50%. Even with a binary trim search there is a reduction of up to 29%. However, in case the chip-to-chip variation is less than 5% and using a stepsize below eight results in an unwanted overhead due to the confirmation tests after each skip search. Therefore, larger stepsizes are generally preferred. The data in Table 3.4 shows the benefit of a large stepsize during the skip progression, with the best results between 21% and 29% for a stepsize of 16 using binary search. This improvement is done without adding any additional failures and thus without impacting the yield.

#### 3.2.3.3. Future Work

There is an opportunity for improvement of the skip test concept, which we want to briefly mention. After a confirmation test of the boundary candidate, the failure locations for the specific trim settings are known. By using additional temporary memory in the MBIST to store these failure addresses, consecutive tests for refining the trim boundary with a lower address skip are only needed on these specific addresses. This has the potential to save even more test time on the cost of an MBIST with increased complexity and area. Provoking more failures by using a higher test temperature can potentially reduce the test time by allowing larger skips. This however needs special equipment to perform high temperature test. Other patterns to select the tested subarray can be used to avoid a sampling bias.

### 3.2.4. Conclusion

The trim setting search is an important step during the manufacturing process of an STT-MRAM chip. However, thorough testing is typically used to perform the necessary algorithms. In this chapter, we evaluated the possibility to skip memory addresses during these tests to reduce test time. As using a single fixed skip setting for all chip tests does not ensure a reliable trim configuration for all chips, we proposed a progressive trim search, which gradually increases the test coverage during the trim boundary search in case a valid trim setting could not be found with less thorough testing. This allows to reliably set the trim configuration of a chip while reducing the trim test time by up to 55% and 29% for linear trim search and binary trim search, respectively, without sacrificing any yield.

# 4. A Novel Oscillation-based Reconfigurable In-Memory Computing Scheme with Error-Correction

In-memory computing architectures can reduce the shortcomings of conventional memories and provide massive parallelism and energy efficiency. On top of that, the emerging resistive memories make the storage highly scalable, more dense and non-volatile [97, 65, 92]. Resistive-based memories are typically organized in a crossbar form [29], where the storing devices act as digital switches based on their resistance states. Several general-purpose in-memory architectures have earlier been proposed for bit-wise Boolean logic operations [45, 89, 84, 55, 44, 80, 100]. However, all these architectures are either bound to a specific memory technology or have a high energy or area overhead for in-memory computation. Additionally, they are rather inflexible in their supported operations, meaning the binary operations and their number of inputs are fixed at the design time. Therefore, there is a decisive need for a generic architecture for resistive memories that can easily be tuned for a wide range of logic operations with a minimal cost. This versatility can be utilized at the higher programming levels to better take advantage of in-memory computing.

As it is the case for conventional memory architectures, in-memory computing architectures are also suffering from run-time faults [45] depending on their used cell technology. Therefore, modern memory designs implement ECCs [61] to detect and eventually correct them [26]. These ECCs usually work on a wordline basis as this is the default access mechanism for a conventional memory architecture. However, this access pattern changes drastically for in-memory computing architectures, where it is common to use multiple wordlines at once.

In this chapter, we propose a generic in-memory computation technique that provides on-demand realization of any symmetric Boolean logic operations for any number of inputs. Unlike in traditional approaches, we have employed a unique method in which the resistance states are first converted to an electrical oscillation. Later, a sensing mechanism is activated to obtain the output when the oscillation pattern has reached to a level where it represents the required logic function. To perform this operation for in-memory computation, we have designed a current controlled mechanism to generate oscillations, developed a methodology to track the oscillation patterns and designed a unique sensing scheme to obtain the final output. Since our proposed technique is oscillation based, a single sensing operation is sufficient to obtain any in-memory logic operations independent of their number of inputs. Due to these reasons, our proposed technique consumes significantly less energy as well as it can be easily generalized for any resistive memory technologies such as ReRAM, PCM and STT-MRAM. In this work, we have developed a

simulation framework for in-memory computing using both ReRAM as well as STT-MRAM based devices. This framework is further used to implement a novel column-wise ECC architecture which is able to correct errors on a per-column basis and can be used to significantly reduce the area and static power consumption of the ECC overhead.

Results show that the XOR2 implementation of our proposed technique can improve the dynamic energy by 41% compared to the state-of-the-art comparator based scheme with a timing penalty and area overhead of 86% and 31%, respectively. The area overhead of our proposed design is reduced and turns into an improvement for higher input operations. Our evaluations show that the calculations on the STT-MRAM array are typically faster and consequently less energy demanding than the operation on the ReRAM, yet the ReRAM operations are more resistant against process variation as they offer a much higher sense margin for the sense amplifier. Additionally, we have evaluated our approach under the influence of process variation using extensive Monte-Carlo simulations to verify the correct functionality of the proposed circuit.

The rest of the chapter is organized as follows. Section 4.1 discusses related work. Section 4.2 describes our proposed in-memory computation concept and the actual circuit implementation. In Section 4.3, we elaborate on our idea to use the previously presented in-memory computing concept for column-wise ECC. Afterwards, in Section 4.4, we evaluate our proposed idea at circuit level as well as on system level based on a case-study on ECC. Finally, we conclude the chapter in Section 4.5.

## 4.1. Related Work

There are many in-memory computing architectures proposed earlier targeting application specific designs such as dot-product [49], scratchpad [45], parallel adder [29] and *Ternary Content Addressable Memory* [101]. Additionally, many general purpose bit-wise logic operations using in-memory computing architecture are proposed previously [100, 89, 84, 55, 45, 44, 80]. Statefull logic as described in [89] destroys the content of the cell during an operation. Ambit [84] uses DRAM as a storage cell, which are volatile and additionally destroy their content during the in-memory operation as well. Destroying the cell information is a drawback we want to avoid, therefore we use a near-memory comparator-based scheme as the following designs do as well. Pinatubo [55] uses a modified sense amplifier to implement the XOR2 operation alongside the OR2 and AND2. However, they only describe two-bit operations that are not capable of scaling with an increase in the number of inputs. As for [45, 100, 80], their comparator based designs offer all conventional two-bit operations in a way, that would theoretically allow scaling. They explore the Compute-in-Memory paradigm on MTJs, ReRAM and *Ferrorelelectric Field Effect Transistors* (FeFETs) respectively, based on the same concept of generating a current sum and using comparators along with additional reference currents to evaluate the results of their target function. This implies a reference for each supported function for basic logic functions like AND or OR and additional CMOS devices for more complex functions like XOR. Additionally, they cannot efficiently be scaled to an increased number functional parameters as it would result in a large area overhead by replicating the needed

circuitry to cover the multiple inputs. The method in [44] provides the possibility to scale AND and OR operations for multi-bit input, yet a multi-bit XOR operation is not included.

The logic operations using in-memory computing can be leveraged for error detection, correction and tolerance in order to enhance the reliability of the system. For instance, *Algorithm-Based Fault Tolerance* (ABFT) schemes that are widely adopted for *Graphics Processing Unit* (GPU) accelerator architectures, mainly used to protect important scientific operations like matrix multiplications [43], matrix LU-Decomposition [47], matrix inversion [47], and QR factorization [79]. In ABFT, the input data is first encoded with a checksum before it is accessed and latter it is compared with the original ones to detect and locate errors, if any mismatch found. However, for floating-point operations, which are inevitably prone to rounding errors, the ABFT has to be adjusted with the rounding errors in the checksum, which make it inefficient to tolerate required errors. Therefore, an autonomous ABFT scheme is proposed in [14], that determines rounding error bounds at runtime. Using in-memory computing can allow these ABFT schemes to optimize their operational overhead. ECCs are commonly employed for on-chip and off-chip data storage to detect, correct and monitor errors, which are mainly based on the *Bose-Chaudhuri-Hocquenhgem* (BCH) codes. ECC schemes utilize certain parity bits to detect and locate errors when memory bits are accessed (row-wise). ECC for STT based in-memory computing is explored in [45]. They use conventional row-wise error correction for data retention and the ECC preserving property of the XOR operation to validate their 2-bit in-memory XOR operation. Our proposed ECC scheme on the other hand is focused on data integrity of the whole memory array.

In summary, in this chapter, we propose a novel method to perform in-memory computing for various symmetric Boolean logic operations, which scales well with more input and can select the type of function at runtime. In contrast to previous works, we are able to compute multi-bit complex function like higher input XOR. Moreover, we have exploited the XOR logic in-memory operation to demonstrate an efficient column-wise error detection/correction scheme, which is relevant in the context of in-memory computing.

## 4.2. Proposed Oscillation-based in-memory computing scheme

### 4.2.1. Overview

The general flow of our scheme is shown in Figure 4.1. Our main idea is to calculate the result of an in-memory operation using only *one* reference and *one* comparison, independent of the number of inputs or the implemented Boolean function. This is achieved by converting the logic (resistance) states of the selected cells into different oscillations, depending on the actual stored content. Comparing the oscillation created by an operation with a pre-determined threshold at a specific sensing time, it is possible to calculate the result of an operation. This concept can be used to calculate the result of any *symmetric Boolean function*, of which the output solely depends on the number of LRS input cells, independent of the input order. In particular, all possible oscillations of an N-input XOR have a common synchronization time, depending on their frequency.

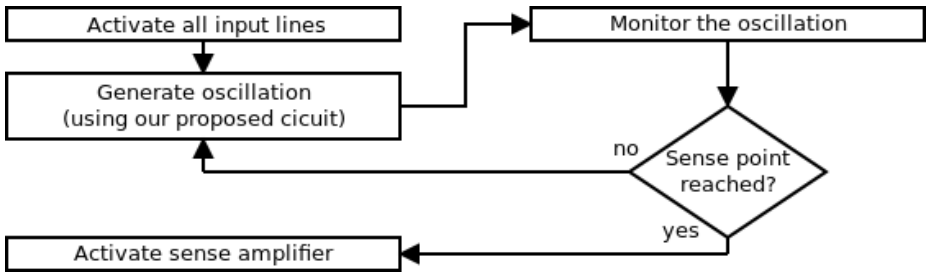



Figure 4.1.: Overview of the proposed in-memory computing scheme flow.

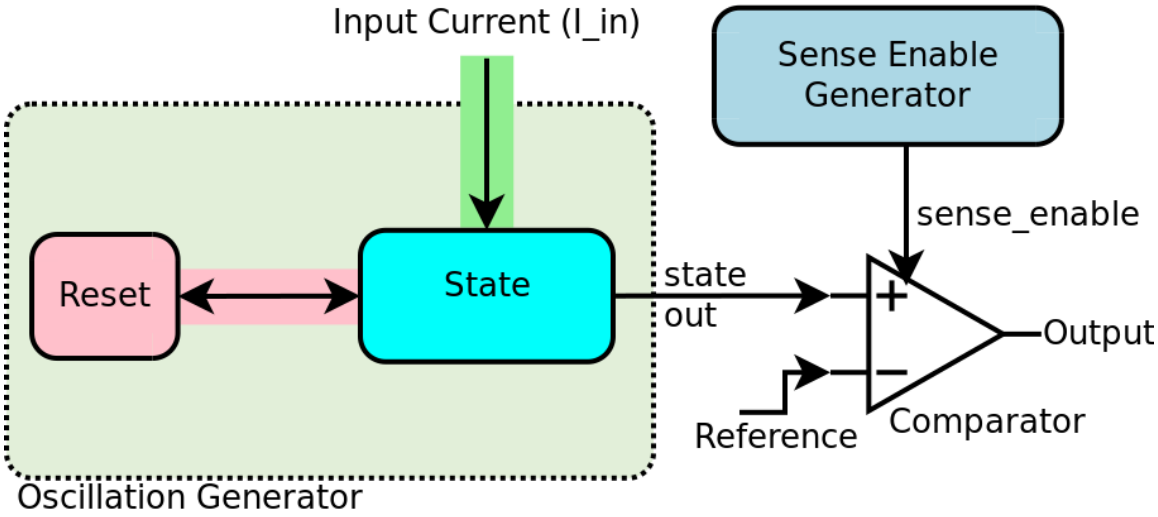



Figure 4.2.: Block diagram of the proposed scheme.

### 4.2.2. Implementation details

An important step in the proposed evaluation scheme is the creation of the oscillations based on the total resistance of selected cells. The block level description of the proposed idea is shown in Figure 4.2. The frequency of the oscillation should be dependent on the current flowing through a fixed number of cells and their respective state. As we want to calculate symmetric Boolean functions, it is sufficient that only the ratio of cells in LRS and the cells in HRS, i.e. the binary inputs, defines the oscillation frequency. Therefore, we can use the sum of all currents passing through the cells to be evaluated as the input of our oscillation generator. The current state of the oscillation is stored in a continuous state holding component, which is varied over time depending on the aforementioned input current. A reset component is used to set the value of the state holding component to a fixed content, whenever a threshold is reached. The constant input current from the memory array continuously *increases* the state value and the regular reset event sets the state back to its initial, which generates the oscillation. A sense enable generator keeps track of the passed time since the oscillation started. After a certain amount of time, which is determined a priori, a comparator is used to evaluate the current state by comparing it to a reference. The calculation of this comparison time is described later in Section 4.2.2.2. This is then used to determine the output of the implemented function.

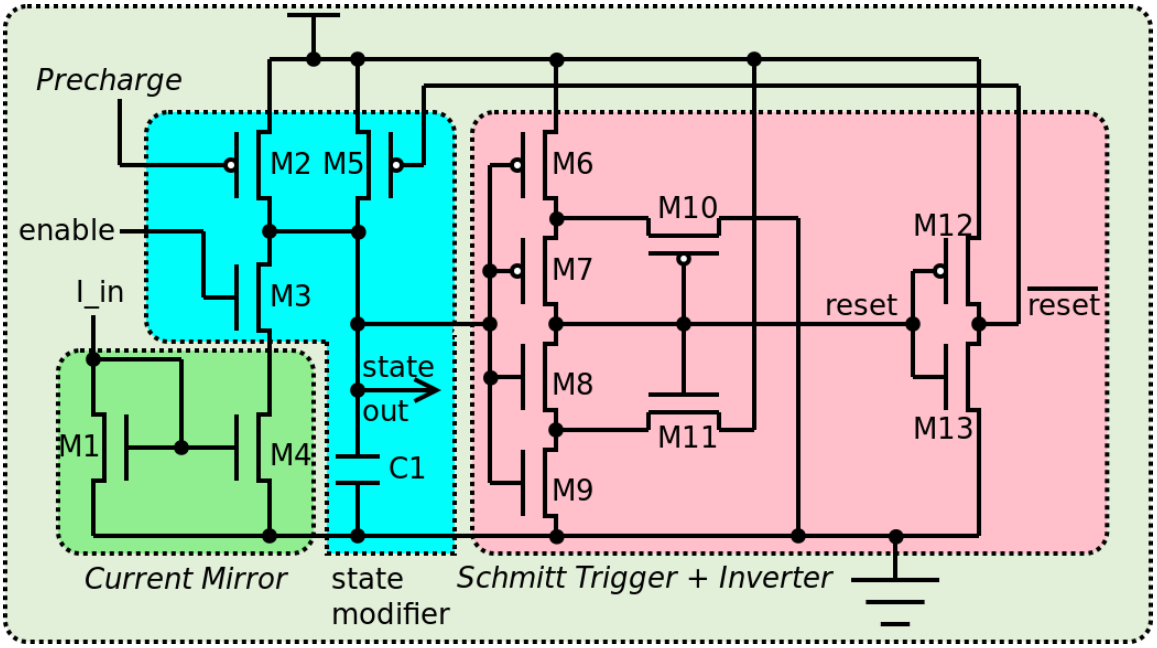



Figure 4.3.: Proposed circuit diagram of the current controlled oscillator.

#### 4.2.2.1. Circuit design

The circuit level description of our proposed oscillator is shown in Figure 4.3. It is built around the central capacitor C1. Additionally, a precharge transistor (M2), a Hold transistor (M3), a current mirror (M1 and M4), a reset transistor (M5), a Schmitt Trigger (M6-M11) and an inverter (M12 and M13) are used. The charge on C1 is used as the current state as described in Section 4.2.2. An *enable* signal is used to activate the discharge of C1 through M3 and consequently the current mirror. It can be used to hold the oscillator for the sensing operation if needed. A precharge signal enables the initial fully charge of C1 with transistor M2. If the enable signal is high and the capacitor discharges, the discharging rate depends on the current flow of *I_in* through M1, as that flow is mirrored using a current mirror circuitry. When using the current sum of the enabled bit cells as the input current to the oscillator, the overall discharging rate of C1 is proportional to the number of the enabled cells in one column and their resistance (logic) states. When C1 has been discharged long enough, the voltage across C1 falls below the lower threshold of the Schmitt Trigger, a reset signal is triggered. This reset signal recharges the capacitor again until the upper threshold of the Schmitt Trigger is reached. By design, the higher the current flow through M1, the faster C1 discharges and consequently the faster it is reset. The circuit therefore behaves like a current-controlled oscillator with a fast rising and a very slow falling slope.

#### 4.2.2.2. Working details

The idea is to create an oscillation dependent on the number of enabled cells in HRS and in LRS. Every LRS cell increases the current flow through M1 with a higher magnitude than a cell in HRS and consequently the *frequency of the oscillation increases*. If we consider a

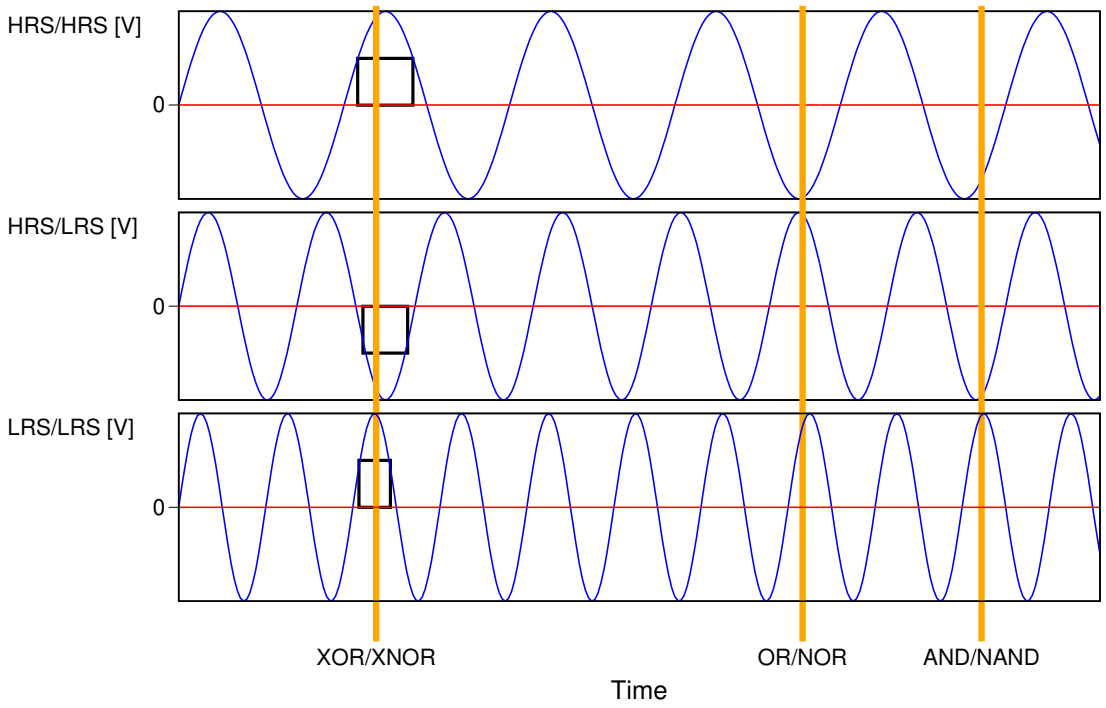



Figure 4.4.: Illustration of conceptual oscillation waveforms for a two-input operation **(top)** with both selected resistive cells in HRS, **(middle)** one cell in LRS and the other one in HRS, and **(bottom)** both in LRS.

symmetric N-input operation (function), N+1 different input variants are possible. Each of those inputs can either be one or zero (HRS / LRS), so there are N+1 possible current flows and also N+1 possible frequencies $\varphi_{I\_in}$.

To implement a symmetric Boolean function, we are interested in the point on time $t$ where

$$state(I\_in, t) < V_{th}, \ \forall I\_in \in I\_in_{true} \tag{4.1}$$

$$state(I\_in, t) \geq V_{th}, \ \forall I\_in \in I\_in_{false} \tag{4.2}$$

are satisfied. In Equation 4.1 and 4.2, $I\_in_{true}$ and $I\_in_{false}$ are the currents when the cells are set in a state that would implicate the output of the evaluation to be true or false, respectively.

As there are a finite number of frequencies, the time $t$ can always be found by searching for the synchronization points of all possible frequencies. Therefore, the difficulty to find t increases with the number of inputs for the symmetric Boolean function to realize. Figure 4.4 gives a conceptual example of three waveforms from the possible oscillations of a two-input operation. By sensing the state of the capacitor at the marked times and comparing it to the reference (shown in red), the result of the binary operation can be calculated. The actual symmetric function is selected by choosing the appropriate sense time. As an example, consider two cells, the first representing a binary 1 and the second representing a binary 0. Both are enabled and the reading current is summed up on the bitline and used as the input to the oscillation generator. The oscillator will output the

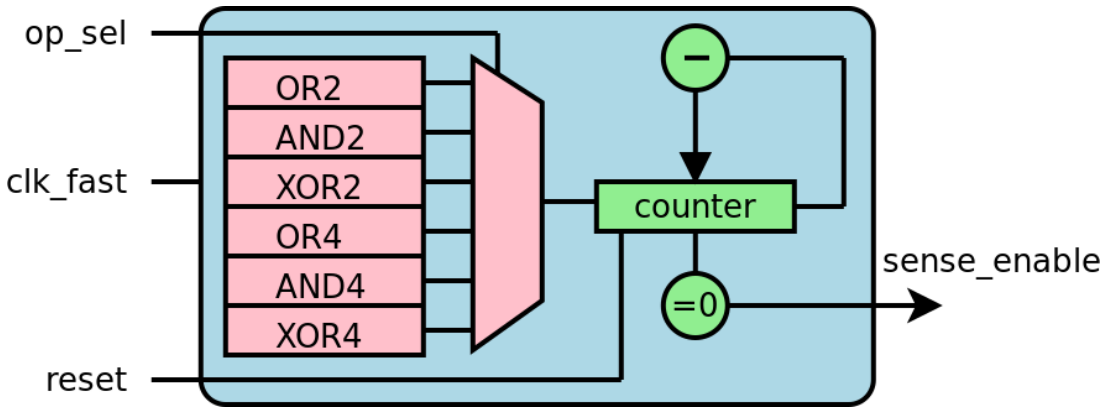



Figure 4.5.: Block level description of the sense amplifier enable signal generator composed of a lookup table for the counter initialization values (red) and the counter with underflow detection (green). The content of look-up table can be trimmed post fabrication to account for process variations. The op_sel signal can be used at runtime to select the function.

signal as shown in the middle waveform of Figure 4.4 (HRS/LRS). To calculate the result of an XOR operation, the output of the oscillator is compared to the reference voltage at the XOR sense time. The comparison shows that the output of the oscillator is below the threshold and therefore the result of the XOR operation is 1. To calculate the sense margins of the operation, the areas of the black boxes indicate the possible sensing time and voltage margins. Their intersection, depending on the performed function, is used to select the appropriate sensing time.

The main benefit of this approach is that independent of the number of function inputs and the type of symmetric function (XOR, XNOR, Majority, etc.), only one comparison has to be done, which makes the circuit implementation of the sensing and aggregation very simple and compact. However, there exists a trade-off between the delay of the evaluation and the number of inputs. As the number of inputs increases, each new input adds one additional frequency to be considered. This will increase the time to find an appropriate synchronization point and the resulting sensing time. The main limitation of our approach is the indistinguishably of states with equal current sums. This is common among all approaches using the current sum as a basis of their in-memory computation. However, to sense small differences, a comparator-based design needs precise references, which are difficult to manufacture because of process variations. With our design this is not necessary, as we can slow down the oscillation process and amplify the differences by adjusting only the capacitance of the oscillator.

#### 4.2.2.3. Programmability and post-fabrication calibration

The generation of the sense timing signal can easily be implemented by using a counter which is fed by a fast clock (clk_fast). A block level implementation of this concept is shown in Figure 4.5. At the beginning an initialization vector is loaded from the lookup table into the counter and continuously counted down. When the counter underflows

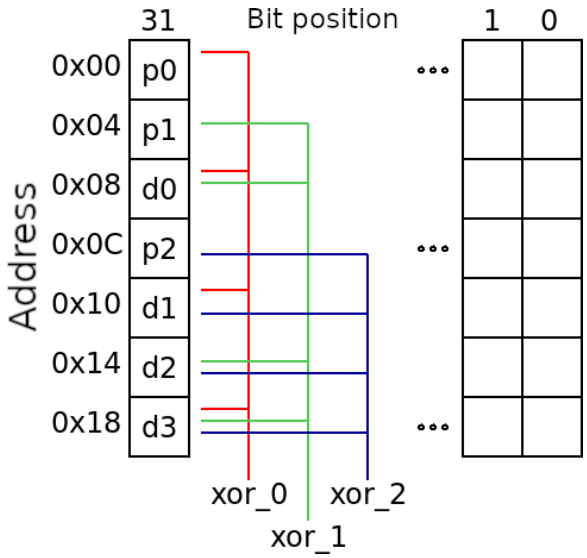



Figure 4.6.: Description of the proposed column-wise error correction scheme using a Hamming code on a seven-bit code word and three XOR4 operations per column.

the sense enable signal is generated. This approach allows to use different timings and therefore reuse the whole circuitry and reconfigure it for different symmetric functions and different number of inputs. In other words, the operation select (op_sel) signal to select the output of the MUX can be used as a part of in-memory instruction opcode to enable runtime selection of implemented functions on the same fabric. This means that the same memory block (array and periphery) can be used to realize different functions. It is also possible to reconfigure the lookup table on a per-application basis, so that the supported operations are loaded at the beginning and can quickly be used when needed. Additionally, by including post-manufacturing characterization data, the content of the look-up table (the values of counters) can be adjusted in a per-chip manner to compensate process variation. This can be achieved by calculating the output frequency from measurements for known inputs and adjust the sense time accordingly. The post-manufacture tuning of multiple sense amplifiers and their reference generation is more difficult than the adjustment of a value in our lookup table.

## 4.3. Proposed column-wise error detection/correction scheme

Using the reconfigurable in-memory computation scheme, which was presented in the previous section, we show how to take advantage of multi-input XOR operations for an efficient implementation of an error correction scheme which fits the requirement of in-memory computing. Typically, for normal memory accesses, error correction is done on line level (i.e. row), when a certain data word is accessed. It means that redundant information is stored alongside each line of the memory. This is mainly because in traditional memory architectures, the unit of an access is one memory row. This redundant information is dependent on the used error correction code, the needed reliability

guarantees and of course the actual data stored in the data bits of the row. In traditional memory designs this redundant information is then used to correct errors either during each read operation or by a specific refresh operation, such as scrubbing [24], which is performed regularly based on the reliability guarantees. However, the access model used for in-memory computing is completely different. As the calculation is done on a column-wise basis and data access is performed on multiple rows or even the entire array at once, it is no longer appropriate to consider only row-level correction. In addition, exploiting the fact that there are far more rows than columns in a typical aspect ratio of a memory, putting check bits column-wise versus row-wise can significantly reduce the number of redundant check bits, for the same error correction capability.

Modifying the error correction from this traditional row-wise approach to a column-wise one as shown in Figure 4.6 affects the overall interface of the error correction significantly. To correct an error while reading a row from the memory in column-wise ECC, the whole memory would need to be read in order to calculate the checksum and perform error decoding and correction. For conventional memory architectures, this leads to a sequential read of one row after another for every data acquisition from the memory. However, in-memory computing architectures provide an opportunity to perform this in an efficient way.

To perform the necessary calculations for error correction, a bitwise XOR is needed. Using the conventional approach of calculating an N-bit XOR with comparator based in-memory computing like in [45], this calculation would need N+1 different reference resistances and at least log(N)+1 comparisons using binary search with a sense amplifier. The implementation of this concept would be complex, costly and error prone mainly due to process variations in bit cells as well as the sensing circuitry. Our proposed scheme is able to evaluate the N-bit XOR in-memory by using the previously described circuit to avoid these drawbacks.

## 4.4. Evaluation

### 4.4.1. Experimental Setup

For circuit level evaluation of our proposed design, we use the *Cadence Spectre* Simulator tool. To back up our claim on the generic usability of resistive devices, we evaluated our approach on Spintronic based MTJs, as well as on ReRAM cells. MTJs have a low HRS/LRS ratio compared to ReRAM. Nevertheless, our approach is directly compatible with other resistive memories like OxRAM, PCM and others. All simulations were performed with VDD=1.1V, T=27°C and the TSMC 40 nm low-power SPICE models for *Complementary Metal-Oxide-Semiconductor* (CMOS) components.

The MTJ model for the evaluation is done as described in [59]. The device parameters of the MTJ was as follows: the radius was 20nm, the Free/Oxide layer thickness was 1.84/1.48 nm the RA was 6.12 $\Omega\mu m^2$ and the HRS/LRS (AP/P) resistance was 11 k$\Omega$/5 k$\Omega$. For the MTJ circuit evaluation a drain resistance with 42.5 k$\Omega$ was put on the bitline during sensing, the comparator threshold voltage was 0.5V and functional validation was done with $\pm$ 3$\sigma$ corners.

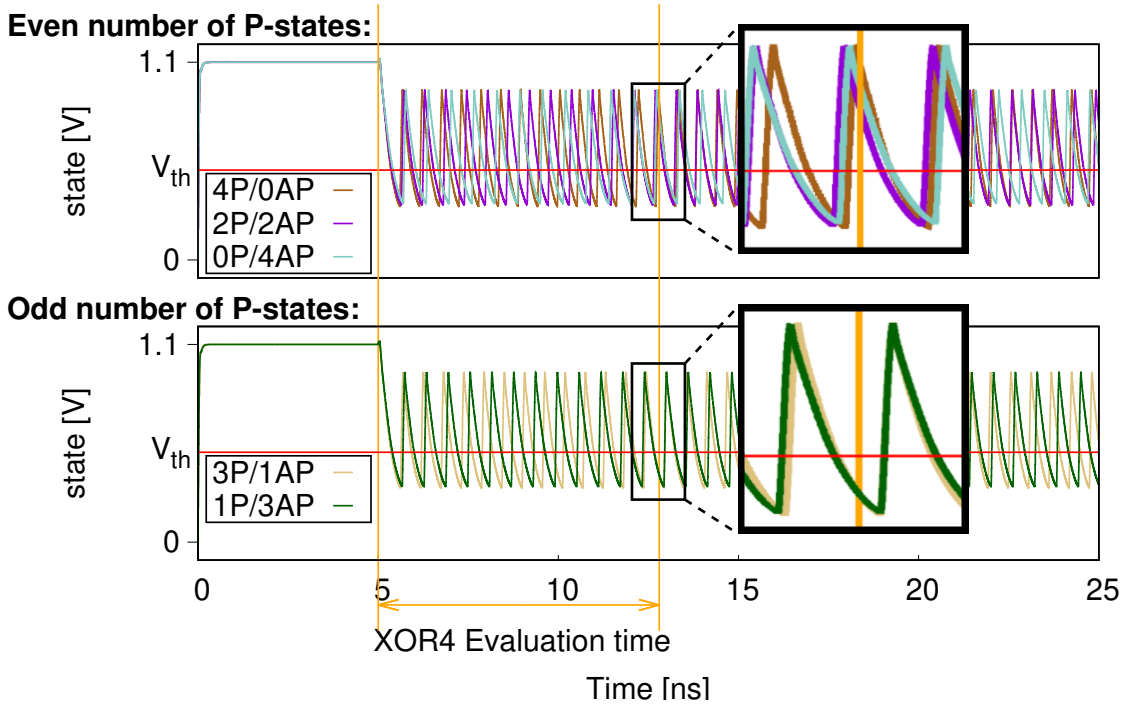



Figure 4.7.: Waveforms of the possible input configurations for XOR4, grouped by the number of P-states from the input MTJs with a reference voltage $V_{th}$ marked in red.

We modeled the ReRAM with HRS/LRS resistance of 100 kΩ/10 kΩ as experimentally obtained in [22], the used drain resistance was 10 kΩ and the comparator threshold voltage was 0.7V. As there are a range of process parameters for the ReRAM cells which influence their behavior, we chose this particular model to balance the need for a larger HRS/LRS ratio compared to the respective MTJ ratio and an acceptable expected resistance variation in the different ReRAM states [54].

### 4.4.2. Functional Validation

Figure 4.7 shows the different possible oscillations for the realization of XOR4 based on STT-MRAM. Each waveform is created by exactly one input value combination. The three signal sequences in the top graph are created with the current sum of 0P/4AP (no MTJ in P-state, four MTJs in AP-state), 2P/2AP and 4P/0AP input MTJ configuration (all even number of P-states). So for all of these inputs the expected output of the XOR4 is 0. The two signal sequences in the bottom graph are created likewise with the current sum of 1P/3AP and 3P/1AP input MTJ configurations (all odd number of P-states). Both of these configurations have the expected output of 1.

We can find a time $t$, in the presented simulation $t = 12.8ns$, where all the odd number of P-state waveforms are above the set threshold of 0.5V and all the even number of P-state waveforms are below this threshold, with a proper margin. As a result, we can now define

an evaluation time for the XOR4 operation of 7.8 ns from the beginning of the oscillation to the sense time.

For such an XOR4 operation the difference between the lowest state above the threshold and the highest state below the threshold during the sense time is around 350mV. This margin is sufficient for a sense amplifier to evaluate it correctly. Nevertheless, there is a trade-off between the evaluation time and the possible sense margins. By re-sizing the oscillation circuit, it is possible to increase the frequencies and to reach the needed synchronization point earlier. However, it is typically not possible to evaluate the state with the sense amplifier at exactly the calculated time, instead it will vary due to process variation. Therefore, a slower synchronization will allow for a more reliable sensing point. Also, the oscillator can be put in Hold mode to allow for a longer and therefore more accurate sensing operation.

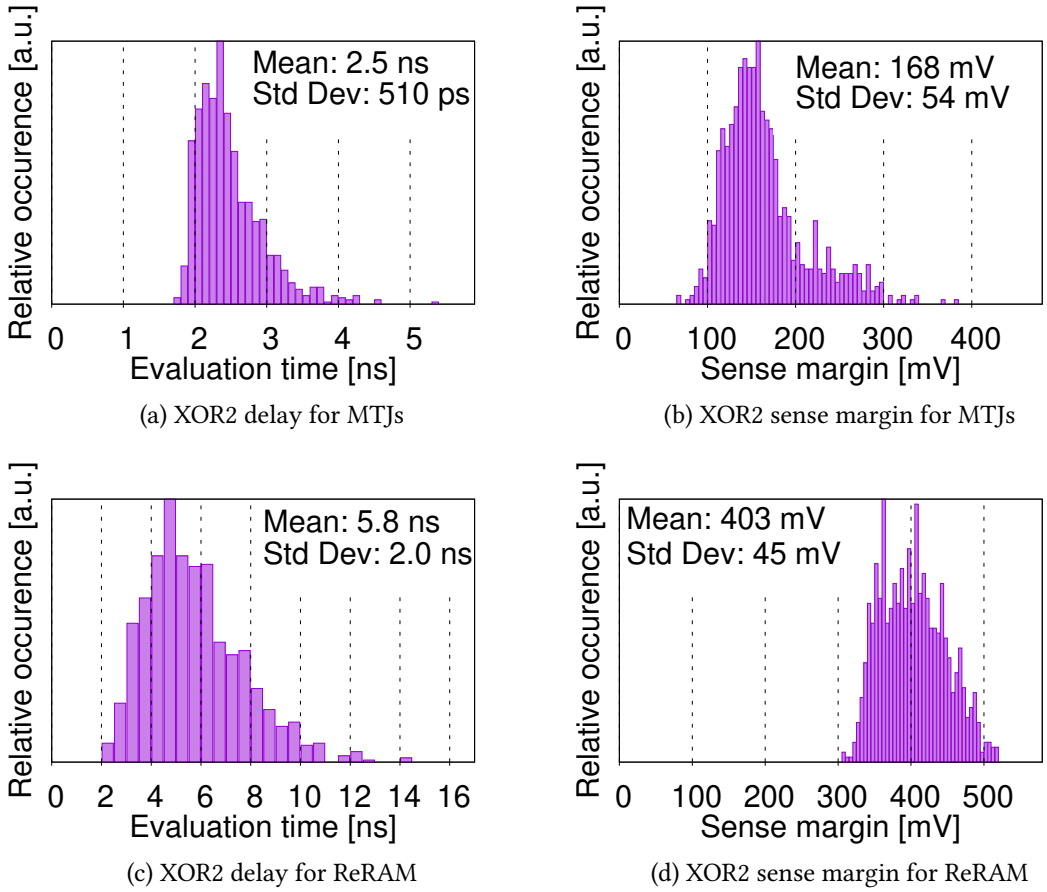
### 4.4.3. Process Variation Analysis

In order to investigate the correct functionality of the proposed approach in the presence of process variations, we have performed extensive Monte-Carlo simulations. Figure 4.8a and Figure 4.8c show results for process variation and mismatch evaluation of the XOR2 evaluation time of our oscillation circuit using MTJ and ReRAM arrays, respectively. For the CMOS and MTJ process variation and mismatch we used the model-provided variations. The ReRAM process variation and mismatch was modeled by varying the resistance with a Gaussian distribution of 5% variance. We considered process variation and mismatch of the MTJ/ReRAM cells and the oscillator. We accept a possible reading time if the resulting value is stable for at least 100 ps. This has implications on the sense margins, which are typically between 100 mV and 200 mV (as shown in Figure 4.8b and Figure 4.8d) and therefore sufficient for the sense amplifier. We confirmed this behavior with the AND2 and the OR2 operations, too. We have observed that due to the large HRS/LRS ratio of ReRAM cells compared to the ratio of MTJs the overall evaluation time for ReRAM is longer and the spread is wider, but once the sensing time is reached, the sense margin is significantly better. This can be explained with a slower oscillation because of the higher resistances and consequently a longer overlap of the oscillation periods.

Figure 4.9a shows the influence of MTJ process variation on the oscillation period of four-bit operations for various MTJ input configurations. Figure 4.9b shows the influence of ReRAM process variation. As the HRS and the LRS are significantly different, the oscillation periods are diverging more, the more cells are set in HRS. As can be seen in the figure, different input configurations do not overlap meaning that there is sufficient margin to be distinguishable, i.e. despite the influence of process variation, the design can deliver correct output.

### 4.4.4. Energy and Delay Analysis

We explored the design space for the oscillating AND/OR/XOR implementation up to 8 inputs, allowing up to 8 rows to be processed in a single evaluation run. We have also compared it with the traditional comparator-based approaches. The idea of evaluating the current sum was also used in [45, 80, 100]. We used the approach described in [45] to




(a) XOR2 delay for MTJs

(b) XOR2 sense margin for MTJs

(c) XOR2 delay for ReRAM

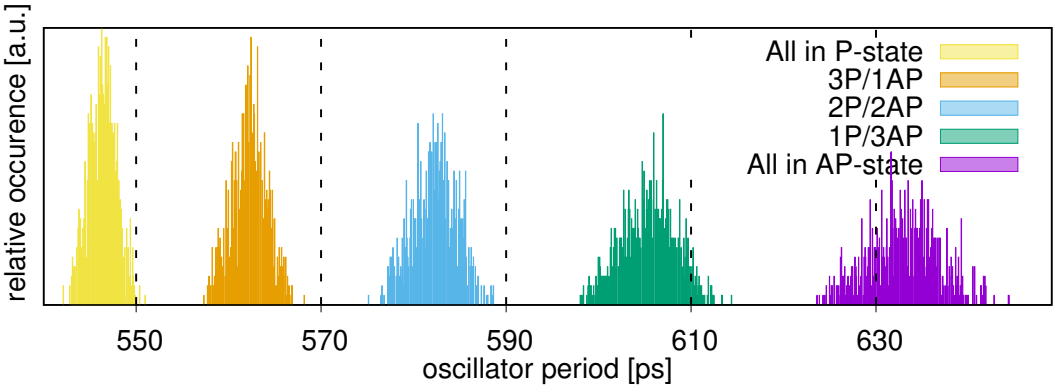
(d) XOR2 sense margin for ReRAM

Figure 4.8.: Monte-Carlo simulation for XOR2 oscillation. The data is generated from 1000 Monte-Carlo simulations.

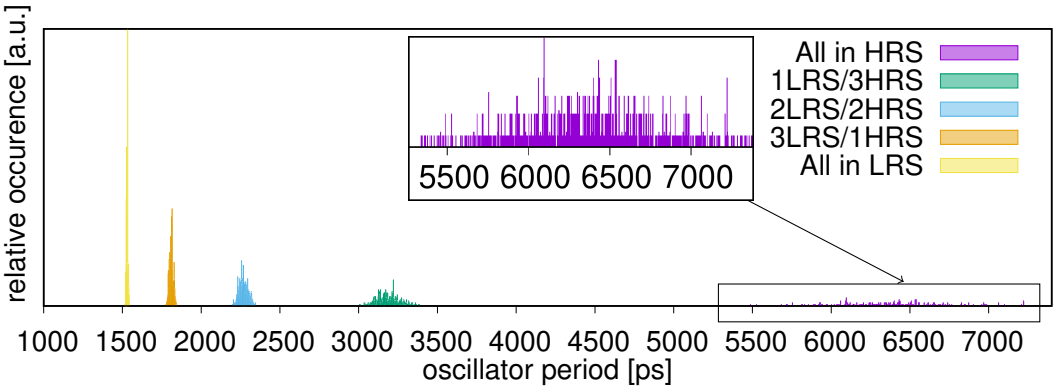
evaluate the result of a XOR2 in which two sense amplifiers in parallel and two different reference currents are used to calculate the result with CMOS-logic. Our approach is able to do the same XOR operation with only one sense operation. To compare our higher input functions with the comparator-based approach, we designed four-bit and eight-bit versions of the comparator-based approach, which are essentially a version of an analog-digital-converter, that compare the current-sum with multiple thresholds at once and use CMOS-based logic to calculate the result with this measurement.

Table 4.1 shows the extracted evaluation times and the resulting energy consumption of the oscillation-based design for each evaluated operation using MTJs and ReRAM cells as technology as well as the comparable comparator-based design [45]. We also give an estimation of the die area needed to implement each approach. The results have been validated with corner case simulation of $3\sigma$.

Our oscillator-approach on MTJ arrays performs slightly more energy efficient for simple threshold operations like AND and OR for the 2/4/8-bit operations. The more complex evaluation of XOR2/4/8 is performed with a significant improvement of dynamic energy consumption. As our approach only uses one sense amplifier, we are able to mitigate the longer runtime for the evaluation of the operation. The oscillator-approach




(a) MTJ array




(b) ReRAM array

Figure 4.9.: Monte-Carlo results of oscillation period distributions for all possible four-bit oscillations. Each dataset shows the result of 1000 Monte-Carlo simulations.

| | | Delay $[ns]$ | | | Dynamic Energy $[fJ]$ | | | Area $[um^2]$ | Delay-Area Product $[ns * um^2]$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AND | OR | XOR | AND | OR | XOR | | AND | OR | XOR |
| STT-MRAM | 2bit | 8.5 | 2.5 | 3.9 | 213 | 153 | 166 | 39.36 | 334.6 | 98.4 | 153.5 |
| | 4bit | 17.1 | 13.9 | 7.8 | 426 | 370 | 265 | | 673.1 | 547.1 | 307.0 |
| | 8bit | 90.3 | 85.2 | 42.3 | 1044 | 991 | 556 | | 3554.2 | 3353.5 | 1664.9 |
| ReRAM | 2bit | 3.9 | 1.9 | 5.4 | 142 | 134 | 155 | 39.36 | 153.5 | 74.8 | 212.5 |
| | 4bit | 1.2 | 5.6 | 13.1 | 136 | 196 | 224 | | 47.2 | 220.4 | 515.6 |
| | 8bit | 13.7 | 13.6 | 227.1 | 309 | 309 | 3101 | | 539.2 | 535.3 | 8938.7 |
| Comparator [45] | 2bit | 1.3 | | 1.4 | 264 | | 264 | 29.98 | 39.0 | | 42.0 |
| | 4bit | 4.5 | | 4.7 | 528 | | 529 | 72.99 | 328.5 | | 343.1 |
| | 8bit | 4.8 | | 5.0 | 1056 | | 1057 | 174.10 | 835.7 | | 870.5 |

Table 4.1.: Circuit level evaluation of the proposed circuit using STT-MRAM and ReRAM as well as a reference design using the comparator-based approach with various input sizes.

on ReRAM arrays is able to save a significant amount of energy by using a lower frequency swing which allows a faster sense time for AND and OR operations. These operations consume less energy than their MTJ array counterpart and are a large improvement over the comparator-based approach. Except for the XOR8, the XOR operations use less dynamic energy than the MTJ array and the comparator-based method. Our design is around 46% smaller than the four-bit implementation of a comparator- based approach and does even better compared to the eight-bit implementation. This is possible as our oscillator can be used without modification of the circuit for different technologies.

To compare the different operations using the different evaluated methods, we calculated the delay-area product for each individual operation. Both, the delay and the area are designed to be minimal so a smaller delay-area product is more preferable. The evaluated circuit was calibrated to properly function with the 4-bit input operations and characterized with this setup for the two-bit and eight-bit functions. This was done to evaluate the ability of our circuit design to adapt to runtime changes. However, it would be possible to optimize towards another target, e.g. the eight-bit functions, to improve their respective behavior. Therefore, we can see from the delay-area that our proposed design used in STT-MRAM is able to perform more efficiently for the complex XOR4 operation and shows an improvement over AND4/OR4 for ReRAM architectures. We also see that using the four-bit calibrated circuit for the two-bit operations is generally not as efficient as the comparator-based approach. However, as this operation is essentially calculated in an area neutral way (we only need an additional sense time stored within the sense enable generator), this suboptimal behavior is acceptable at the benefit of runtime reconfigurability.

### 4.4.5. Column-wise error detection/correction scheme

As a case study, the implementation of the scrubbing technique supported with the proposed in-memory ECC scheme to mitigating retention failures is presented.

In STT-MRAM architectures, the retention time, which is the maximum time for retaining data reliably, depends exponentially on $\Delta$. However, process variation during the manufacturing of the MTJ leads to significant variations in the thermal stability factor among the entire cells of the STT-MRAM memory array. This in turn induces large variations in the retention time from a few microseconds to several years [87]. This issue is aggravated when STT-MRAM is used with relaxed retention time (i.e., lowered $\Delta$) in order to achieve a higher performance and lower energy [87], which makes the retention failure a major reliability concern.

Scrubbing is a conventional solution to address retention failures [82]. It is performed in a regular interval, which is estimated based on the worst expected retention time of a cell in the memory array (i.e., the cell with a minimum $\Delta$). For each scrubbing process, the data in all lines needs to be read sequentially and checked for errors with conventional row-wise error correcting code. Figure 4.10a illustrates the structure of a memory array of 8 Kbyte, which is supported with a conventional Hamming code for a single error correcting scheme. Hence, ECC$(n, m, e)$ is needed for a regular scrubbing process, where $m = 57$ is the data bits, $n = m + r = 64$ is the code bits, and $r = 7$ is the redundant bits needed to correct $e = 1$ error per row. Figure 4.10b shows the circuit-level implementation

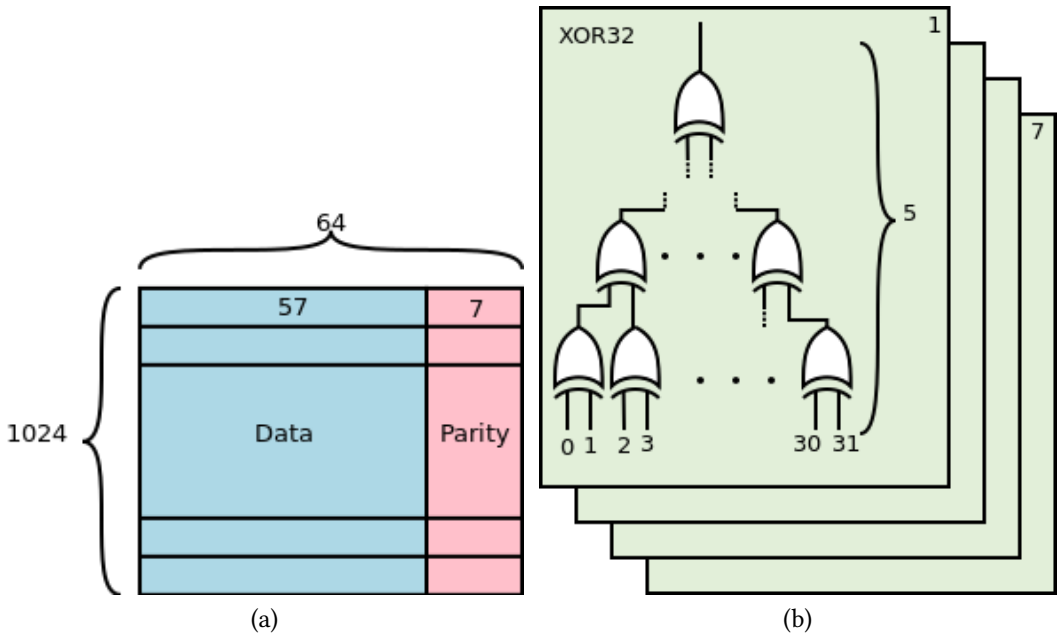



Figure 4.10.: Conventional row-wise ECC using XOR-tree to calculate the error position.

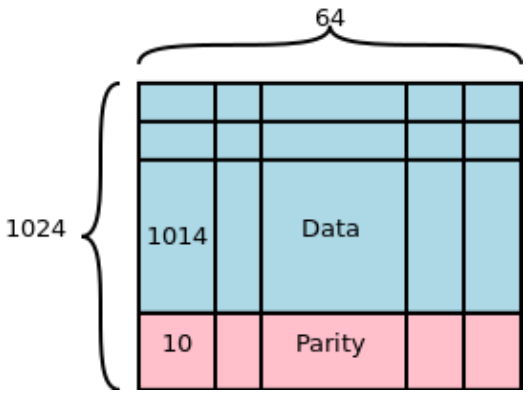



Figure 4.11.: Proposed column-wise ECC correction scheme.

| Δ value | Scrubbing rate per second | |
|---|---|---|
| | Column-wise approach ECC(1024,1014,1) | Row-wise approach ECC(64,57,1) |
| 30 | $1.68 \times 10^{3}$ | $1.05 \times 10^{2}$ |
| 40 | $8.67 \times 10^{-2}$ | $5.4 \times 10^{-3}$ |
| 50 | $4.25X10^{-12}$ | $2.65X10^{-11}$ |

Table 4.2.: Scrubbing rate per second for proposed in-memory computing ECC and conventional ECC.

of ECC(64, 57, 1). This circuitry, which is used to check the parity of each redundant bit, consists of 7 XOR-trees each with a depth of 5.

Our proposed column-wise in-memory error correction provides an attractive method to correct the memory array, where the data of all columns will be scrubbed simultaneously. This reduces the performance overhead of the scrubbing process at system level, as the memory will be blocked from processor accesses, while it is being scrubbed. Figure 4.11 illustrates the structure of an 8 Kbyte memory array, which is supported with our proposed column-wise in-memory single error correcting scheme (ECC(1024, 1014, 1)).

For accurate estimation, the retention time, based on process variation effects, has to be defined according to the failure budget for the whole system, which is measured as *Failure in Time* (FIT). One FIT equals one failure in one billion hours of operation. The FIT rate per single cache row and column has to be bounded by FIT/R and FIT/C, where R and C refers to the number of memory rows and columns, respectively. Consequently, the single bit failure probability has to be driven from either FIT/R or FIT/C. We evaluate the retention failure probability per bit based on a typical FIT rate for memories (i.e., 10 failures in 1 billion hours) and the single error correcting capability ECC(64, 57, 1) for a conventional row-wise correction and an ECC(1024, 1014, 1) for our proposed column-wise approach. Table 4.2 presents the scrubbing rates for both approaches along with Δ scaling considering 5% of process variation. We specifically chose the value of the thermal stability factor to range from 30 to 50 as this interval covers the most relevant MTJ retention times from several minutes (semi-volatile memory applications) to multiple years (non-volatile memory applications) [87].

The parity overhead of ECC(64, 57, 1) using the conventional row-wise approach is 11% of the whole memory array, whereas the parity overhead of ECC(1024, 1014, 1) with our proposed scheme is only around 1%. We are therefore able to reduce the overhead of the redundant parity bits by 10%. As for the additional circuitry to detect and correct errors, using one of our proposed circuit per column instead of multiple XOR-trees decreases the extra area overhead by 21%.

We want to point out the possibility to use our presented approach orthogonal to the row-wise technique. This allows to further increase the error resilience by using both row-wise and column-wise ECC at the same time. Therefore, we generated calculation delays and area overhead for the conventional row-wise ECC correction with the capability to correct one error (ECC1) in form of multiple layers of XOR2s implemented in CMOS. The same is done with our proposed circuit for column-wise calculation.

## 4.5. Conclusion

The in-memory computing paradigm promises to overcome the memory and the power wall. However, energy efficient calculation of various binary operations with different number of inputs is challenging. In this chapter, a highly generic approach for arbitrary input size in-memory symmetric Boolean operations has been presented, which requires only a single comparison and a single reference value, independent of the target operation. This approach has been evaluated based on circuit level implementation of the most common binary operations, namely AND, OR and XOR using ReRAM and STT-MRAM based memory technology. This has further been evaluated for the implementation of column-wise in-memory ECC, which fits the requirements of in-memory computing, in contrast to traditional memory access.

Results show that our in-memory oscillation approach can save 41 % in dynamic sense energy compared to state-of-the-art comparator-based approaches and additionally scales well for more inputs. Additionally, using our approach for a column-wise ECC implementation can reduce the ECC parity overhead by 10% as well as decrease the detection circuit area by 21% compared to conventional ECC approaches.

# 5. Compute-in-Memory Defect Analysis and Test Methods for STT-MRAM

In this chapter, we show the analysis of resistive defects in CiM specific behavior of STT-MRAM. The concept is first shown with basic two-operand binary operations and later in the chapter extended to complex multi-operand operations. The results of the analysis are used to derive efficient testing methods in form of modified marching tests and a trim circuit supported test methodology.

## 5.1. Defect Characterization and Test Generation for Spintronic-based Compute-In-Memory

In the following we model and evaluate the impact of various defects (opens and shorts) in the bit-cell of STT-MRAM on CiM architecture and compare these faults with those of conventional memory faults. With this information, we develop new test algorithms to detect the CiM specific faults, which can be used in conjunction with the conventional memory tests for complete fault coverage of CiM enabled STT-MRAM architectures.

The rest of the chapter is organized as follows. Section 5.1.1 discusses related work. In Section 5.1.2, we describe our fault injection framework and present and evaluate the obtained results. These results are used in Section 5.1.3 to generate a test algorithm, which can be used to detect the CiM related faults in an STT-MRAM architecture. Lastly, Section 5.1.4 concludes the chapter.

### 5.1.1. Related Work

The are several works focusing on testing of emerging non-volatile memories such as [36, 18, 50, 21, 74]. The works in [36] and [18] focus on testing of ReRAM, whereas as [50] focuses on testing of both ReRAM and PCM. The concept of CiM is also discussed in the context of pure CMOS manufacturing process. Test generation for CMOS-based 8T-SRAM cells used in the core array of a CiM architecture is addressed in [90]. Testing of the computational part in the CiM memory array based on ReRAM is discussed in [35]. Also, they extensively describe the fault modeling and test development for scouting logic based CiM periphery. In [32] Memristor Ratioed Logic is discussed and testing mechanisms for an XOR implementation with this concept are evaluated. These works use architectures and implementations which are different from the scouting logic used in this chapter based on STT-MRAM technology [100].

The defects and faults in STT-MRAM are different from those in ReRAM/PCM due to completely different ranges of TMR [21]. These STT-MRAM specific defects and the

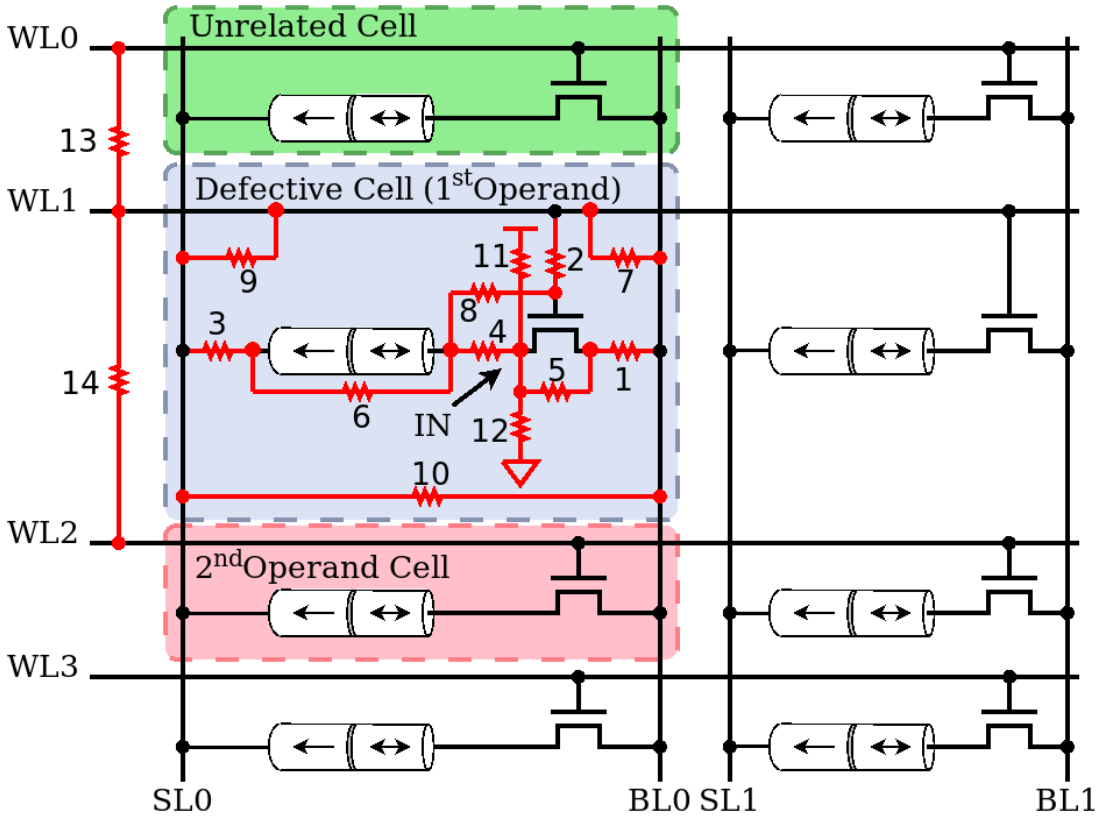



Figure 5.1.: Schematic of defect injection methodology.

resulting faults are analyzed in [21] and [74] for conventional memory operation. However, none of the existing works have discussed STT-based CiM fault modeling and testing. Using STT-MRAM for CiM operations lead to different fault mechanisms, and hence the test patterns for conventional memory testing need to be augmented with additional CiM specific test patterns, which are addressed in this chapter.

## 5.1.2. Fault Analysis Framework

### 5.1.2.1. Defect Injection Setup

For our simulations, we have employed TSMC 40 nm SPICE models for the CMOS components and the PMA-MTJ model from [7]. The parameters of the MTJ model are given in Table 5.4. For defect characterization and fault modeling, we consider two sets of 4×4 memory arrays, one for conventional memory operation and the other for CiM operation. For the CiM operations, the peripheral circuitry is modified to select 2 rows during the read operation and the reference circuitry is adjusted according to the type of CiM operation (AND/OR).

| Defect Type | Location | CiM Operation | | | | | | | | | | | | Conventional Read | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Operation Result AND/OR | | | | Resistance [Ω] AND | | | | Resistance [Ω] OR | | | | Fault | Resistance [Ω] |
| | | 0,0 | 0,1 | 1,0 | 1,1 | 0,0 | 0,1 | 1,0 | 1,1 | 0,0 | 0,1 | 1,0 | 1,1 | | |
| Opens | BL (1) | 1 | 1 | 1 | 1 | 3.3k | 700 | 1.7k | NF | 900 | NF | NF | NF | IRF0 | 900 |
| | WL (2) | 1 | 1 | 1 | 1 | 1M | 100k | 100k | NF | 100k | NF | NF | NF | IRF0 | 700k |
| | SL (3) | 1 | 1 | 1 | 1 | 16k | 1k | 11k | NF | 2k | NF | NF | NF | IRF0 | 2k |
| | IN (4) | 1 | 1 | 1 | 1 | 16k | 1k | 12k | NF | 2k | NF | NF | NF | IRF0 | 2k |
| Shorts | BL - IN (5) | 0 | 1 | 0 | 1 | NF | 200k | NF | NF | NF | NF | 30k | NF | NF | NF |
| | SL - IN (6) | 0 | 0 | 0 | 0 | NF | NF | NF | 20k | NF | 30k | 30k | 8k | IRF1 | 40k |
| | WL - BL (7) | 1 | 1 | 1 | 1 | 2M | 4M | 4M | NF | 3M | NF | NF | NF | IRF0 | 3M |
| | WL - IN (8) | 1 | 1 | 1 | 1 | 2M | 6M | 5M | NF | 5M | NF | NF | NF | IRF0 | 3M |
| | WL - SL (9) | 1 | 1 | 1 | 1 | 900k | 2M | 2M | NF | 3M | NF | NF | NF | IRF0 | 800k |
| | BL -SL (10) | 0 | 0 | 0 | 0 | NF | NF | NF | 30k | NF | 40k | 40k | 20k | IRF1 | 40k |
| | IN - VDD (11) | 1 | 1 | 1 | 1 | 2M | 6M | 6M | NF | 5M | NF | NF | NF | IRF0 | 3M |
| | IN - GND (12) | 0 | 0 | 0 | 0 | NF | NF | NF | 50k | NF | 70k | 90k | 20k | IRF1 | 40k |
| Coupling | WL1 - WL0 (13) | 1 | 1 | 1 | 1 | 100 | 100 | 100 | NF | 100 | NF | NF | NF | IRF0 | 100 |
| | WL1 - WL2 (14) | NF | | | | | | | | | | | | IRF0 | 100 |

Table 5.1.: Fault injection results for two-bit in-memory OR and AND operations as well as the associated read faults. Defect locations can be on the bit-line (BL), the word-lines (WL), the source-line (SL), the internal node (IN) or between each of them. If an operation result does not match the expected outcome, the resistance is given, otherwise it is marked as *No Fault* (NF). For open (short) defects, the resistance shown is the minimum (maximum) resistance above (below) which the output is faulty. The *Operation Results* with the injected errors are given, each with different operand combinations. If the result from the operation (AND/OR) is different from the expected result, the corresponding resistance of the defect is given in the *Resistance* columns.

| Defect Type | Location | AND | | | OR | | |
|---|---|---|---|---|---|---|---|
| | | Resistance Range [Ω] | CiM Fault | Memory Fault | Resistance Range [Ω] | CiM Fault | Memory Fault |
| Opens | BL (1) | R ≤ 700 | NF | NF | R ≤ 900 | NF | NF |
| | | 700 < R ≤ 900 | **IANDF0** | **NF** | R > 900 | IORF0 | IRF0 |
| | | R > 900 | IANDF0 | IRF0 | | | |
| | WL (2) | R ≤ 100k | NF | NF | R ≤ 100k | NF | NF |
| | | 100k < R ≤ 700k | **IANDF0** | **NF** | 100k < R ≤ 700k | **IORF0** | **NF** |
| | | R > 700k | IANDF0 | IRF0 | R > 700k | IORF0 | IRF0 |
| | SL (3) | R ≤ 1k | NF | NF | R ≤ 2k | NF | NF |
| | | 1k < R ≤ 2k | **IANDF0** | **NF** | R > 2k | IORF0 | IRF0 |
| | | R > 2k | IANDF0 | IRF0 | | | |
| | IN (4) | R ≤ 1k | NF | NF | R ≤ 2k | NF | NF |
| | | 1k < R ≤ 2k | **IANDF0** | **NF** | R > 2k | IORF0 | IRF0 |
| | | R > 2k | IANDF0 | IRF0 | | | |
| Shorts | BL - IN (5) | R < 200k | **IANDF0** | **NF** | R < 30k | **IORF1** | **NF** |
| | | R ≥ 200k | NF | NF | R ≥ 30k | NF | NF |
| | SL - IN (6) | R ≥ 40k | NF | NF | R ≥ 40k | NF | NF |
| | | 20k ≤ R < 40k | *NF* | *IRF1* | 30k ≤ R < 40k | *NF* | *IRF1* |
| | | R < 20k | IANDF1 | IRF1 | R < 30k | IANDF1 | IRF1 |
| | WL - BL (7) | R ≥ 4M | NF | NF | R ≥ 3M | NF | NF |
| | | 3M ≤ R < 4M | **IANDF0** | **NF** | R < 3M | IORF0 | IRF0 |
| | | R < 3M | IANDF0 | IRF0 | | | |
| | WL - IN (8) | R ≥ 6M | NF | NF | R ≥ 5M | NF | NF |
| | | 3M ≤ R < 6M | **IANDF0** | **NF** | 3M ≤ R < 5M | **IORF0** | **NF** |
| | | R < 3M | IANDF0 | IRF0 | R < 3M | IORF0 | IRF0 |
| | WL - SL (9) | R ≥ 2M | NF | NF | R ≥ 3M | NF | NF |
| | | 800k ≤ R < 2M | **IANDF0** | **NF** | 800k ≤ R < 3M | **IORF0** | **NF** |
| | | R < 800k | IANDF0 | IRF0 | R < 800k | IORF0 | IRF0 |
| | BL -SL (10) | R ≥ 40k | NF | NF | R ≥ 40k | NF | NF |
| | | 30k ≤ R < 40k | *NF* | *IRF1* | R < 40k | IORF1 | IRF1 |
| | | R < 30k | IANDF1 | IRF1 | | | |
| | IN - VDD (11) | R ≥ 6M | NF | NF | R ≥ 5M | NF | NF |
| | | 3M ≤ R < 6M | **IANDF0** | **NF** | 3M ≤ R < 5M | **IORF0** | **NF** |
| | | R < 3M | IANDF0 | IRF0 | R < 3M | IORF0 | IRF0 |
| | IN - GND (12) | R ≥ 50k | NF | NF | R ≥ 90k | NF | NF |
| | | 40k ≤ R < 50k | **IANDF1** | **NF** | 40k ≤ R < 90k | **IORF1** | **NF** |
| | | R < 40k | IANDF1 | IRF1 | R < 40k | IANDF1 | IRF1 |
| Coupling | WL1 - WL0 (13) | R < 100 | IANDF0 | IRF0 | R < 100 | IORF0 | IRF0 |
| | | R ≥ 100 | NF | NF | R ≥ 100 | NF | NF |
| | WL1 - WL2 (14) | R < 100 | *NF* | *IRF0* | R < 100 | *NF* | *IRF0* |
| | | R ≥ 100 | NF | NF | R ≥ 100 | NF | NF |

Table 5.2.: Resistance range for memory and CiM read faults for two-bit in-memory OR and AND operations. Highlighted operations are either memory only faults (*blue*) or CiM only faults (**red**). We classify CiM faults as *Incorrect AND Faults* (IANDF0 and IANDF1) and *Incorrect OR Faults* (IORF0 and IORF1) where e.g., IANDF0 happens when the result of an AND operation is 1, but 0 was expected.

| Defect Type | Location | AND | | OR | | Minimal Test |
|---|---|---|---|---|---|---|
| | | Resistance | Test Pattern | Resistance | Test Pattern | |
| Opens | BL (1) | 700 < R ≤ 900 | AND(0,1) | - | - | AND(0,1) |
| | WL (2) | 100k < R ≤ 700k | AND(0,1), AND(1,0) | 100k < R ≤ 700k | OR(0,0) | AND(0,1) |
| | SL (3) | 1k < R ≤ 2k | AND(0,1) | - | | AND(0,1) |
| | IN (4) | 1k < R ≤ 2k | AND(0,1) | - | - | AND(0,1) |
| Shorts | BL - IN (5) | R < 200k | AND(0,1) | R < 30k | OR(1,0) | AND(0,1) |
| | SL - IN (6) | - | - | - | - | - |
| | WL - BL (7) | 3M ≤ R < 4M | AND(0,1), AND(1,0) | - | - | AND(0,1) |
| | WL - IN (8) | 3M ≤ R < 6M | AND(0,1) | 3M ≤ R < 5M | OR(0,0) | AND(0,1) |
| | WL - SL (9) | 800k ≤ R < 2M | AND(0,1), AND(1,0) | 800k ≤ R < 3M | OR(0,0) | OR(0,0) |
| | BL -SL (10) | - | - | - | - | - |
| | IN - VDD (11) | 3M ≤ R < 6M | AND(0,1), AND(1,0) | 3M ≤ R < 5M | OR(0,0) | AND(0,1) |
| | IN - GND (12) | 40k ≤ R < 50k | AND(1,1) | 40k ≤ R < 90k | OR(1,0) | OR(1,0) |
| Coupling | WL1 - WL0 (13) | - | - | - | - | - |
| | WL1 - WL2 (14) | - | - | - | - | - |

Table 5.3.: Defects resulting in CiM faults which are not covered by conventional memory testing and the corresponding test patterns.

| Parameter | Value |
|---|---|
| VDD and Temperature | 1.2V and 27°C |
| MTJ radius | 20 nm |
| Free/Oxide layer thickness | 1.84/1.48 nm |
| RA | 6.12 $\Omega\mu m^2$ |
| $TMR_{read}$ @ 0V | 123% |
| $TMR_{AND}$ @ 0V | 60% |
| $TMR_{OR}$ @ 0V | 37% |
| 'AP'/'P' resistance | 11 kΩ/5 kΩ |

Table 5.4.: MTJ parameters and simulation setup

#### 5.1.2.2. Defect Injection Flow

A typical CiM enabled STT-MRAM core array and the associated defect injection is shown in Fig. 5.1. Here, row 1 and row 2 are enabled together for CiM operation by enabling the word-lines WL1 and WL2. To illustrate the defect modeling methodology, we have chosen row 1 as a row with a defective cell. The defects in STT-MRAM are modeled as opens and shorts between different nodes in a bit-cell. We also consider coupling between word-lines of the defective cell and an unrelated cell as well as coupling between the world-lines of the defective cell and the second operand cell. We restrict our modeling to static read and CiM faults. In particular, the write functionality can be tested with conventional memory tests.

All the defects are modeled by resistances inserted between different nodes in the memory as presented in Fig. 5.1. The nodes considered are BL, SL, WL and the internal node in the bit-cell (IN), where the access transistor is connected to the MTJ [21, 74]. First, the defect free circuit is simulated to obtain the correct outputs of the read operation. Next, resistances are inserted for each of the defects as depicted in Fig. 5.1. The resistance values are then swept to find the range of resistances for which faults are observed. This is done for both conventional memory and CiM array to compare the resulting faults. The resistance ranges thus obtained for each of the defects are given in Table 5.1 with location numbers according to Fig. 5.1.

For the conventional read operation, the read faults are classified as *Incorrect Read Faults* (IRF), IRF0 and IRF1. IRF1(0) happens when a bit-cell storing a '1' ('0') is incorrectly read as '0' ('1') by the sense-amplifier. For instance, when BL is open, an IRF0 fault occurs, which means that the read output will be '1' when the bit-cell is storing a '0'.

For the CiM operation, any deviation from the expected logical result of the AND/OR operation is considered as a fault. For instance, for BL open, the output of the CiM operation (both OR and AND) is always '1'. This is because an open in the BL disconnects one of operand cells, which significantly increases the effective (parallel) resistance of the two operand cells, biasing the output to always '1'. Similarly, an SL-IN short significantly decreases the effective resistance due to the short across the MTJ, biasing the output to always '0'. Similar explanations can be given for other defect cases as well.

In addition to the open/shorts explained above, STT-MRAM is also affected by MTJ related defects. The MTJ defects are primarily due to oxide thickness and resistance-area (RA) product variations. These defects mainly affect the 'P' and 'AP' resistances of the MTJ,

resulting in IRFs. A low (high) value of RA or oxide thickness results in lower (higher) values of the 'P' and 'AP' resistances. In turn, a low value of 'P' and 'AP' resistances causes IRF1, since the 'AP' resistance decreases below the reference resistance. For the CiM operation, this results in the read output to be always '0'. Similarly, a high value of 'P' and 'AP' resistances causes IRF0 for the conventional memory operation and the output to be always '1' for CiM operations. Since these faults will be detected by conventional memory testing, we focus on the open/short defects, which need additional test patterns for detection in CiM operations, as explained below. In accordance to the conventional read operation faults, we classify CiM faults as *Incorrect AND Faults* (IANDF0 and IANDF1) and *Incorrect OR Faults* (IORF0 and IORF1). For instance, IANDF0 happens when the result of an AND operation is 1, but 0 was expected.

#### 5.1.2.3. Fault Analysis Results

Table 5.1 shows the resistance range for read faults for both CiM as well as conventional memory operations for shorts, opens and coupling defects. For open defects, the resistance shown in the table is the minimum resistance above which the output is faulty. Similarly, for the short and coupling defects, the resistance corresponds to the maximum resistance below which the output is faulty. For instance, as per Table 5.1, an open in the BL with a resistance of greater than 900 Ω results in IRF0 for the conventional read and a faulty result for the CiM OR operation. There are 4 possible cases to consider.

- *No Memory or CiM Faults*: The memory functions fault-free as expected in this case.
- *Both Memory and CiM Faults*: The conventional memory test patters can detect these cases.
- *Memory Fault but no CiM Fault*: These cases are marked in *blue* in Table 5.1. They can also be detected by conventional memory testing.
- *CiM Fault but no Memory Fault*: These cases are marked in **red** in Table 5.2. Conventional memory test algorithms are not sufficient to detect these faults. They require new test patterns to detect the CiM specific faults, as explained in Section 5.1.3.

### 5.1.3. Test Pattern Generation

A majority of the possible faults can be covered by conventional memory tests. Table 5.3 gives the in-memory specific subset of faults, which are not covered by conventional memory tests and can impact either AND, OR or both CiM operations. We denote an AND and an OR operation with two inputs x and y as AND(x,y) and OR(x,y), respectively. As shown in this table, all faults resulting from shorts between SL and IN, BL and SL and coupling faults between the WLs are covered by conventional memory tests and consequently do not need to be separately tested during CiM specific testing. Each fault can be sensitized by one or multiple test patterns. To reduce the number of test operations, we can see that some test patterns supersede others. However, testing for the most restricting operation, in this particular case AND(0,1) will also detect the faults related to

the OR operation. One possible minimal test set to cover all defects consists of calculating AND(0,1), OR(0,0) and OR(1,0) and validating their expected result covers all possible defect ranges. From these operations we can generate the following March test sequence for N memory addresses.

**Test Sequence** (5.5N):
$\Updownarrow$(W0); $\Updownarrow$(OR0); $\Updownarrow_2$(W1); $\Updownarrow_2$(OR1); $\Updownarrow_2^{+1}$(AND0); $\Updownarrow_2^{+1}$(W1); $\Updownarrow_2$(W0); $\Updownarrow_2^{+1}$(OR1); $\Updownarrow_2$(AND0);

with $\Updownarrow^{+1}$ denoting a test operation starting at offset +1 and $\Updownarrow_2$ performing the operation at every second address. The operands for AND and OR are always consecutively stored in the memory. The sequence first writes all cells to 0 and tests if the result of the OR operation is 0 for all neighboring cells. This is the test case OR(0,0). Then, every second cell is written to 1, resulting in a 0-1 pattern and every even pair of cells is tested with OR(1,0) and every odd pair of cells with AND(0,1). Afterwards, the cell content is inverted, resulting in a 1-0 pattern and the remaining tests for OR(1,0) and AND(0,1) are performed. Merging the CiM specific March test sequence with the conventional memory March test sequence can improve the run-time of the combined tests.

### 5.1.4. Conclusion

STT-MRAM, as a front runner of emerging non-volatile memory technologies, is also a promising solution for CiM architectures. However, the specific defects introduced in its manufacturing process have a unique impact on its capability to do in-memory computing. In this chapter, we performed extensive defect characterization and identified the CiM specific faults relating to these defects in a scouting logic in-memory architecture for two-bit OR and AND operations. We discovered that the normal memory test is not able to detect CiM specific faults. With this characterization, we developed a March test sequence to detect all CiM specific faults (100% coverage), which are not covered by conventional memory testing procedures due to the smaller TMR of two enabled bit-cells and their possible state combinations.

## 5.2. Defect Characterization of Spintronic-based Neuromorphic Circuits

*Machine learning* has gained a lot of attention as an attractive way to solve complex computational problems, particularly with the increase in computational power in the last decade [9, 62]. Neuromorphic computing, aka brain-inspired computing, which partly mimics the behavior of the human brain for learning and inference, has gained a lot of attention [60]. Brain inspired Artificial Neural Networks (ANNs) offer the possibility to solve computational problems, that have been difficult to solve before using traditional computing paradigms, like image classification and pattern recognition in general. In such domains, the analytical relation between inputs and outputs are either unknown or very difficult to model, and hence data-driven models are more successful. However, when implemented on general purpose computational hardware, ANNs need a high amount of computational power. Therefore, specialized hardware with a focus on neuromorphic

architectures can be used to exploit the inherent regular structures of neural networks and reduce the overhead of general purpose architectures for neural computing. In particular, the direct mapping of ANNs to CiM schemes can reduce the unnecessary data transfer between the memory and processing units.

To build these neuromorphic accelerators, STT-MRAM can be used instead of conventional SRAM or DRAM as storage for the neural network weights to additionally allow for in-memory inference because of its resistive storage nature. In addition, it also offers several other advantageous features such as non-volatility and high integration density, which are very attractive for edge inference engines. Many accelerators for different kinds of neural networks have been proposed to exploit this behavior [19, 28, 33, 48, 49, 78].

In this chapter, we characterize, model and evaluate the impact of various defects (opens and shorts) in STT-based neuromorphic architectures. In particular, we evaluate the impact on a threshold synaptic function which is common in neuromorphic operations.

The rest of the chapter is organized as follows. In Section 5.2.1, we describe our fault injection framework and present and evaluate the obtained results. Section 5.2.2 concludes the chapter.

## 5.2.1. Fault Analysis Framework

### 5.2.1.1. Defect Injection Setup

For our simulations, we have employed TSMC 40 nm SPICE models for the CMOS components and the PMA-MTJ model from [7]. The MTJ is modeled with a radius of 20nm, a resistance-area product of $6.12\Omega\, um^2$, a TMR of 125%, a free/oxide layer thickness of 1.84nm/1.48nm and AP/P resistances of 11Ω/5Ω. For defect characterization and fault modeling, we consider an 8×4 memory arrays. For the neuromorphic operations, the peripheral circuitry is modified to select 4 rows during the read operation and the reference circuitry is adjusted for the non-linear activation function. For instance, the output neuron is fired when all activated input neurons are weighted 1, which corresponds to AND4 operation. If AP state is encoded as '1' and P state as '0', the sense amplifier compares the column-wise current to the highest resistant state and in case it is bigger (all selected MTJs in the column in AP state, meaning all synaptic weights are '1'), the output of the sense amplifier is '1'.

### 5.2.1.2. Defect Injection Flow

A typical STT-based neuromorphic array and the associated defect injection is shown in Fig. 5.2. Here, four rows are enabled together for neuromorphic operation by enabling the respective word-lines. To illustrate the defect modeling methodology, we have chosen row 1 as a row with a defective cell. The defects in STT are modeled as opens and shorts between different nodes in a bit-cell. We also consider coupling between word-lines of the defective cell and other cells. We restrict our modeling to inference faults, since the write functionality can be tested with conventional memory tests.

All the defects are modeled by resistances inserted between different nodes in the memory as presented in Fig. 5.2. The nodes considered are BL, SL, WL and the internal

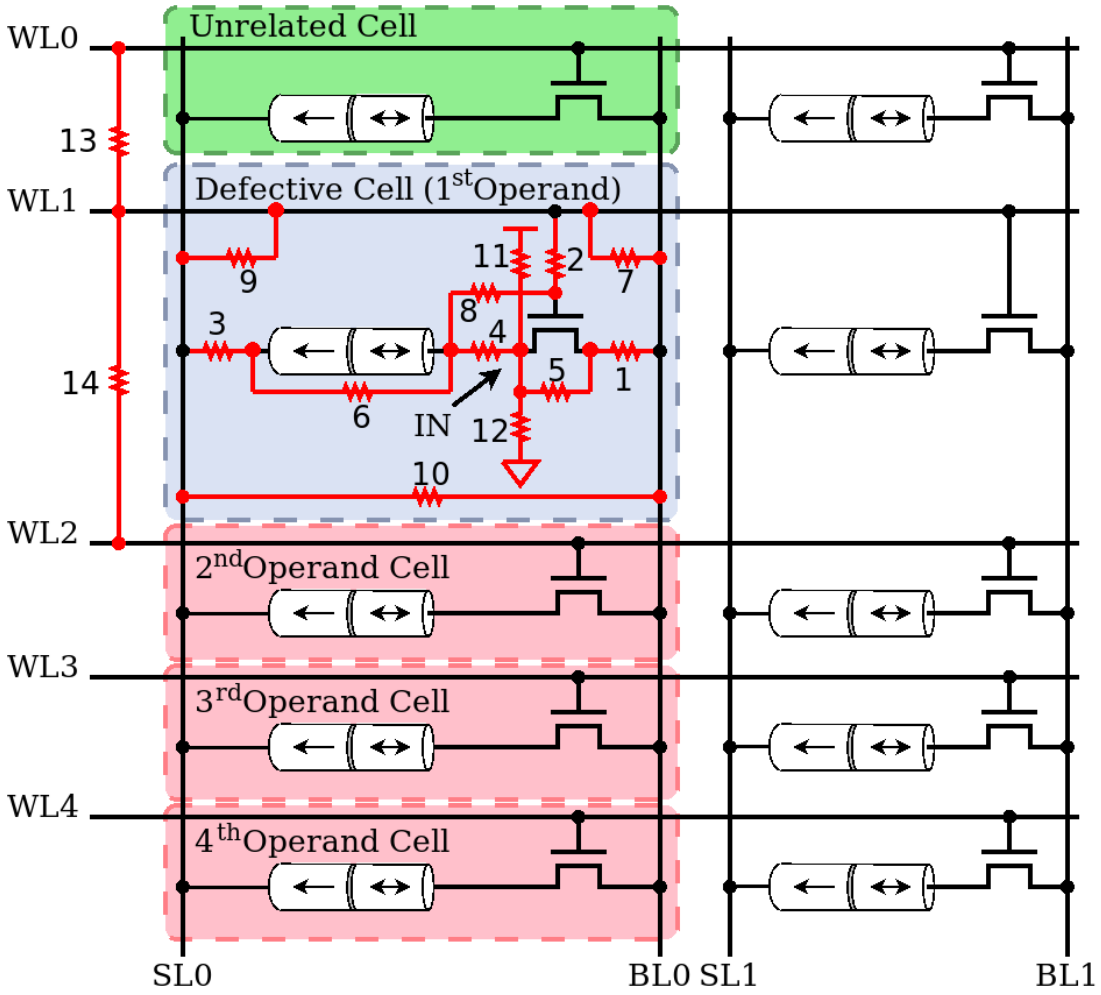



Figure 5.2.: Schematic of defect injection methodology.

node in the bit-cell (IN), where the access transistor is connected to the MTJ [21, 74]. First, the defect free circuit is simulated to obtain the correct outputs of the AND4 operation. Next, resistances are inserted for each of the defects as depicted in Fig. 5.2. The resistance values are then swept to find the range of resistances for which faults are observed. We only consider single defects, so there is only one resistance inserted and swept at a time.

As a result, we get the limits of the proper working condition of the simulated array. For the neuromorphic operation, a result different from the expected logical AND operation result is a fault. For instance, for BL open, the output of the AND operation is always '1'. From a resistive point of view, this can be seen as a cell in a very high resistance state, lowering the current sum of on the bit-line and consequently biasing the output of the sense amplifier to '1'. In general, each of the simulated defect with resulting faults has the ability to either increase or decrease the current sum and therefore influence the output of sense amplifier (comparator) accordingly.

Several STT-MRAM related defects are also covered by our defect injection. Manufacturing process variation on the oxide thickness and the RA product can lead to varying resistances in the 'P'/'AP' states. A low (high) value of RA or oxide thickness results in lower (higher) values of the 'P' and 'AP' resistances. These are covered by SL (3) defect injection, which effectively increases the resistance of the cell and the SL - IN (6) which bypasses the cell and effectively decreases the cell resistance. The stuck-at faults described in Section 2.4 can be spotted with regular STT-MRAM memory test and need no further attention in our fault evaluations.

#### 5.2.1.3. Fault Analysis Results

Table 5.5 shows the resistance limits for the AND4 operation for shorts, opens and coupling defects. For open defects, the resistance shown in the table is the minimum resistance above which the output is faulty. Similarly, for the short and coupling defects, the resistance corresponds to the maximum resistance below which the output is faulty. For instance, as per Table 5.5, an open in the BL with a resistance of greater than 15k Ω results in an IANDF for the input operand configuration (0,0,0,0). In the fault-free case, this would result in the output '0'. But as per the table, a resistive defect of more than 15kΩ leads to the faulty result ('1'). For subsequent memory testing steps, the lowest resistance of a defect has to be considered for Opens and the highest defect resistance for Shorts. Again, as an example, the smallest possible injection for the BL open is 600Ω. So, to guarantee a fault free behavior of the AND4 operation in all possible operand combination, the resistive fault must not be larger than these 600Ω.

### 5.2.2. Conclusion

STT-MRAM is a promising technology for hardware realization of neuromorphic computing. The multiple possible defects specific to this technology make it necessary to evaluate their impact on this specific field. In this chapter we performed an extensive evaluation of these defects on a majority operation, which can be used as a neuromorphic threshold transfer function. The results of defect characterization can be used to generate and grade proper testing patterns for neuromorphic operations.

| Defective Cell | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| Remaining Operands | | 0,0,0 | 1,0,0 | 1,1,0 | 1,1,1 | 0,0,0 | 1,0,0 | 1,1,0 | 1,1,1 |
| Opens | BL (1) | 15kΩ | 4.8kΩ | 2.7kΩ | 600Ω | 15kΩ | 6kΩ | 1kΩ | NF |
| | WL (2) | 9MΩ | 2MΩ | 8MΩ | 40kΩ | 4MΩ | 7MΩ | 30kΩ | NF |
| | SL (3) | NF | NF | 14kΩ | 3kΩ | NF | NF | 14kΩ | NF |
| | IN (4) | NF | NF | 16kΩ | 3kΩ | NF | NF | 13kΩ | NF |
| Shorts | BL - IN (5) | NF | NF | NF | NF | NF | NF | NF | NF |
| | SL - IN (6) | NF | NF | NF | NF | NF | NF | NF | 30kΩ |
| | WL -BL (7) | 4MΩ | 5MΩ | 6MΩ | NF | 5MΩ | 7MΩ | 19MΩ | NF |
| | WL - IN (8) | 4MΩ | 4MΩ | 8MΩ | 20MΩ | 5MΩ | 8MΩ | 30MΩ | NF |
| | WL - SL (9) | 1.8MΩ | 3MΩ | 4MΩ | 15MΩ | 3MΩ | 5MΩ | 16MΩ | NF |
| | BL - SL (10) | NF | NF | NF | NF | NF | NF | NF | 30kΩ |
| | IN - VDD (11) | 4MΩ | 5MΩ | 8MΩ | 26MΩ | 5MΩ | 9MΩ | 26MΩ | NF |
| | IN - GND (12) | NF | NF | NF | NF | NF | NF | NF | 70kΩ |
| Coupling | WL1 - WL0 (13) | 20Ω | 20Ω | 20Ω | 20Ω | 20Ω | 20Ω | 20Ω | NF |
| | WL1 - WL2 (14) | NF | NF | NF | NF | NF | NF | NF | NF |

Table 5.5.: Defect injection results for four-bit in-memory AND operation. If the operand configuration of the defective cell and the remaining operands result in a fault with a specific Open Defect (Short/Coupling), the minimal (maximal) injected resistance to provoke the fault is given. If the defect injection does not cause the operation to result in a fault, hence the observed output is the same as the expected output, it is marked as *no fault* (NF).

## 5.3. Testing Resistive Memory based Neuromorphic Architectures using Reference Trimming

In this chapter, we continue to elaborate on the possible defects and faults of CiM-based ANN hardware, specifically their influence on threshold operations needed for neuromorphic operations. We show the exploitation of the resistive cell behavior to reduce the testing time significantly compared to pure logic testing. Additionally, we take advantage of resistance trimming, to reduce the write operations during the test sequence even further. We chose STT-MRAM to evaluate our resistive testing approach, as it is one of the more mature technologies with a comparable low LRS/HRS ratio and thus the most vulnerable technology to defects which can cause failures in neuromorphic hardware. By selecting the most demanding technology, we are able to generalize our extensive defect injection simulation based analysis to the other more relaxed memory types. These resistive defects typically result in faulty behavior of the cell during a neuromorphic operation. Conventional memory tests are not sufficient to spot the CiM-specific faults [75], the same is therefore true for CiM-based neuromorphic memories, used to implement efficient ANN hardware.

The rest of the chapter is organized as follows. Section 5.3.1 discusses related work. Section 5.3.2 gives our general simulation approach and simulation results followed by Section 5.3.3 with the discussion of our baseline and the derivation of our proposed test generations. Section 5.3.4 concludes the chapter.

### 5.3.1. Related Work

Testing of emerging resistive memory technologies are subject to many studies. Specifically testing of conventional memory operations is discussed in [21, 74, 18, 50, 90]. Some of them focus on a single technology, e.g. STT-MRAM [74] or RRAM [18, 50], whereas others have a broader view on multiple resistive memory technologies [90]. Testing of CiM-capable architectures is discussed in [35] with extensive fault modeling, an RRAM core array and a focus on the memory periphery. STT-MRAM specific defect modeling and test generation in presented in [75]. However, there are also pure CMOS based CiM design: In [90] the testing of an 8T SRAM cell for in-memory computing is presented. Also, there are test concepts to realize CiM operations with resistive memories but without the conventional RAM architecture, e.g. Memristor Rationed Logic [32]. In [38] the general approach of testing the functionality of a neuromorphic circuit is compared to the sole structural testing of the circuitry.

As neuromorphic architectures are very different to regular memories and can heavily utilize CiM-operations, it is necessary to specifically test them and potentially optimize these tests for this neuromorphic scenario. None of the previous work have addressed the specific need of test generation for CiM-based neuromorphic architectures. We therefore show the derivation of a test generation to detect all (single) faults in a CiM-based neuromorphic memory architecture. This is done by extensive fault injection evaluation and re-purposing the resistive trimming circuit.

### 5.3.2. Fault Analysis and Test Requirements for Resistive Neuromorphic Hardware

In this section we have a closer look at the test requirements for neuromorphic hardware, specifically the implications of using non-volatile resistive memory technologies for their implementation. We also use STT-MRAM as a proof of concept because it is the most sensible technology in terms of the sense margin due to a very low LRS/HRS ratio. To derive our proposed test sequences, we use extensive technology-aware defect injection simulations. For illustration purposes, we use the threshold operations with four inputs and generalize to the full array afterwards to derive the test sequence. As write faults can easily be handled by conventional memory test pattern, we focus on the testing of synaptic operation faults (during inference) which are different from conventional memory read faults [63, 40].

#### 5.3.2.1. Defect Injection Framework for Resistive Neuromorphic Arrays

Our defect injection framework is based on a 8x4 STT-MRAM core array, setup to perform four-bit CiM operations by enabling four word-lines at the same time. Figure 5.2 from the last chapter shows the relevant part of the core array, specifically the cell with the injected defects, the three additional operand cells and one unrelated cells. The fault injection is performed by inserting resistors at the marked locations. With this approach we are able to simulate opens (locations 1-4) and shorts (locations 5-14).

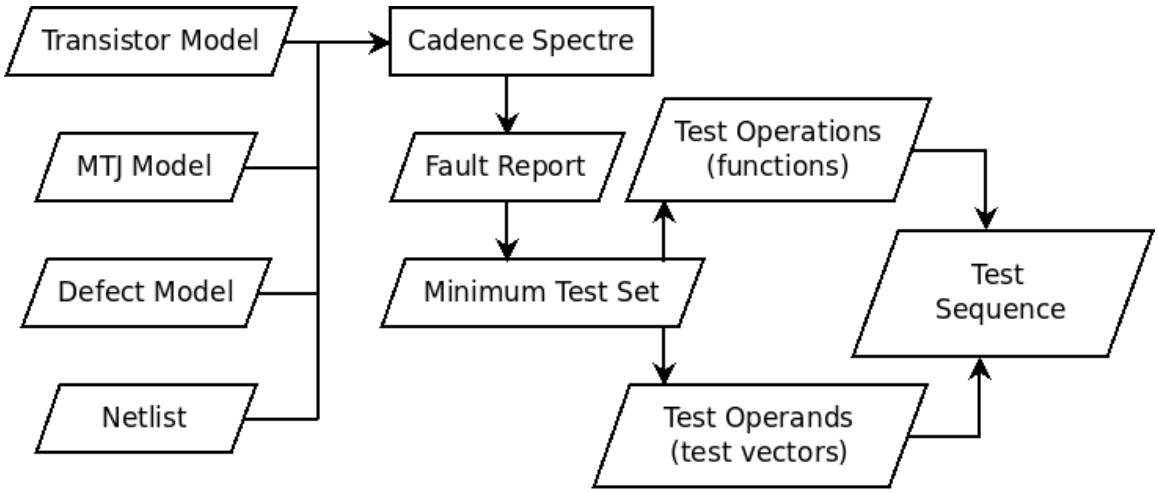



Figure 5.3.: Defect Injection, Test Coverage and Marching Test Sequence Generation Flow.

| Parameter | Value |
|---|---|
| VDD and Temperature | 1.2V and 27°C |
| MTJ radius | 20 nm |
| Free/Oxide layer thickness | 1.3/1.48 nm |
| RA | 7.5 $\Omega\mu m^2$ |
| TMR @ 0V | 150% |
| 'AP'/'P' resistance | 15 kΩ/6 kΩ |

Table 5.6.: MTJ parameters and simulation setup

We performed all of our simulations with *Cadence Virtuoso* and *Cadence Spectre*. The TSMC 40nm Low Power SPICE models were used to model the CMOS devices of our evaluated architecture. Furthermore, an MTJ compact model as described in [59] was used as the resistive memory cell. The rest of our simulation parameters can be found in Table 5.6. Our general workflow for the test generation is shown in Figure 5.3.

#### 5.3.2.2. Neuromorphic Test Requirements

Depending on the operand configuration, the sense margin to differentiate between the states of the activated memory cells changes. As shown in Figure 2.8 for four cells, the difference between $R_{MIN1}$ and the "0000"/"1000" states (SM1) is significantly smaller than the difference between $R_{MIN4}$ and the "1110"/"1111" states (SM4). To generalize this behavior, we can see that the high resistive state combinations are easier to distinguish than the low resistive ones.

In general, writing to a resistive memory cell is demanding more resources in terms of energy and delay than executing a read or a neuromorphic inference operation. This gives an incentive to reduce the number of writing operations during the March test sequence. Please note that in the normal March sequence for conventional CMOS memories, there is no difference in read and write operation on test time. However, for testing neuromorphic hardware based on resistive memories, the objective of test time minimization can be translated into reducing the number of write operations. Therefore, we try to activate the faulty behavior which is otherwise activated by specific operand combinations (i.e.,

requiring write operation in the test sequence) by shifting their respective operation reference with the help of reference trimming and thus changing the effective sense margin. With this concept it is possible to test different defects under the same input combinations without changing the operands. As shown in Figure 2.8 we have evaluated three operations with shifted references. The first is a lower $MIN^4 1$ operation (MIN1L), which is set in the middle of the original $R_{MIN1}$ and the resistance of all cells in LRS. The second and third are lower and higher versions of the $MIN^4 4$ operation (MIN4L/MIN4H), which are offset by the original $MIN^4 1$ sense margin SM1, as indicated by the SM1 arrows.

#### 5.3.2.3. Fault analysis results based on reduced sense margins

We have performed extensive defect injection and fault analysis for different threshold functions ($MIN^N m$) with normal and reduced sense margins. Figure 5.4 shows the faults with the highest sensitivity with regards to the different $MIN^4 m$ operations under test for each defect. The top four graphs depict the most restricting test patterns for open faults. Therefore, if a circuit has to be able to perform all MIN operations, the most dominated open defect which corresponds to lowest possible resistive defect has to be tested. In case of e.g. an open on the bit-line [see the top left image in Figure 5.4 - (1)BL] the smallest possible defect to lead to a fault is sensitized by the MIN4 operation with an operand input of '0111'. Likewise, minimum resistive defects can be found for opens on the word-line, source-line and the internal node.

The remaining graphs show the results of the evaluated short defects. Here the most dominated fault results from the largest resistive defect (ideally, the resistance between two unconnected wires should be infinite). So by testing for the operation with the highest resistance of short defects, all operations with lower defect resistances are covered.

### 5.3.3. Neuromorphic Test Sequence Generation

Based on the neuromorphic fault analysis results presented in the previous section, here we derive different test sequences to detect all single faults for all threshold functions. We start by looking at the problem from pure logic testing perspective and then consider the technology-specific mapping based on resistive memories. Finally, we take advantage of reference trimming to derive the optimal test time.

#### 5.3.3.1. Full Logic Test

Let $MIN^N m$ be the threshold operation to evaluate if at least m bit out of N ($m \leq N$) are set. From a pure logic testing perspective of all possible threshold (MIN) operations, each threshold function $MIN^N m$ requires two sets of test inputs. The first set of test inputs generate the input combination at the exact threshold value (with $m$ 1's and $N - m$ 0s) to detect all (single) stuck-at-0 faults on the output as well as the inputs set to 1. The second set of test patterns correspond to the input combinations just below the threshold (with $m - 1$ 1s and $N - m + 1$ 0s) to detect all (single) stuck-at-1 faults, on the output as well as the inputs set to 0. However, as the upper threshold test of one operation is the lower

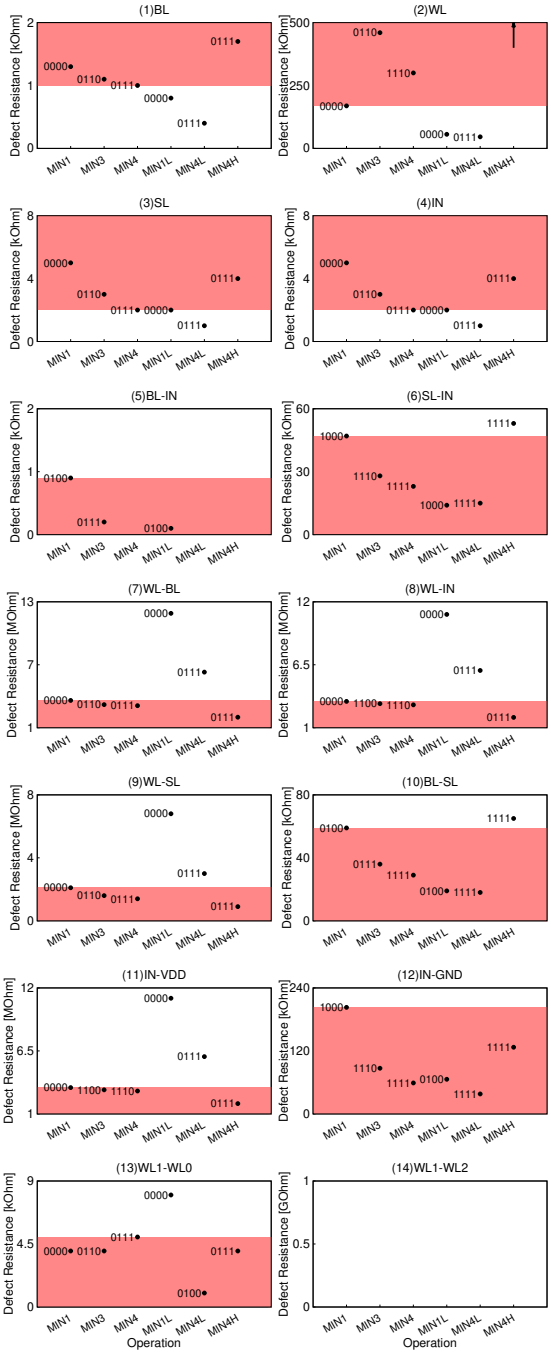



Figure 5.4.: Fault injection results for the four-bit operations: Each injected fault is shown with the most restricting input configuration (depicted next to the respective point) for each evaluated threshold operation. A defect in the red marked area results in at least one faulty behavior during the standard MIN1-MIN4 operations (MIN4L and MIN4H are not considered essential operations). Note, that the MIN4 operations with a bridging defect between BL and IN (5) and no operation with a bridging defect between WL1 and WL2 (14) resulted in an observable fault.

threshold test of the next threshold, half of the test patterns can be shared with testing the previous threshold function $MIN^N m$-1.

**Lemma 5.3.1** *Given a memory with N rows, the test complexity $C_{full}$ for the logic test for all $MIN^N m$ operations with 100% single stuck-at coverage is $2^N$.*

***Proof**: Testing each $MIN^N m$ operation requires $C(N,m) + C(N,m-1)$ test patterns, in which $C(n,k) = n!/k!(n-k)!$. Therefore, to test all MIN operations the total number of test patterns is $\sum_{i=1}^{N} C(N,i) = 2^N$.*

This shows that pure logic testing of a neuromorphic array implementing all possible threshold functions requires all possible combinations of memory values with an exponential complexity.

#### 5.3.3.2. Minimal-Maximal Threshold Test

While the pure logic testing has exponential complexity, the fact that threshold functions are implemented by resistive memories can help to reduce the test complexity. One of the key observations from the fault injection results in Figure 5.4 is that all defective behaviors can be detected by only checking the two extreme threshold functions, $MIN^N 1$ and $MIN^N N$. This is because of the resistive behavior of the cell and the sensing mechanism to implement threshold functions, which allows us to reduce the test complexity significantly compared to the pure logic test. To confirm this, we characterized the defective behavior of $MIN^4 1$, $MIN^4 3$ and $MIN^4 4$ operations experimentally and evaluated the trend between $MIN^4 1$ and $MIN^4 4$. We can see from the results, that the minimum test set can be generated by only using the operations $MIN^4 1(0000)$, $MIN^4 1(0100)$, $MIN^4 1(1000)$ and $MIN^4 4(0111)$.

**Algorithm 1** Full Generic Test Sequence (8N = 6N(W) + 2N(R))

W0, $MIN^N 1\_0$,
**for** i = 1 to N **do**
  W0(i-1), W1(i), $MIN^N 1\_1$,
**end for**
$W1$,
**for** i = 1 to N **do**
  W1(i-1), W0(i), $MIN^N N\_1$,
**end for**

From this we can generate a generic non-marching test sequence as shown in Algorithm 1. By interpreting the four-bit operations as N-bit threshold operations we can formulate $MIN^N 1(0000)$ as $MIN^N 1$ on all zero-cells. Therefore, we start by writing the entire array to zero (W0) and perform a single $MIN^N 1$ operation which expects a 0 as the operation result. $MIN^4 1(0100)$ and $MIN^4 1(1000)$ can be interpreted as a "running 1" test, which performs the $MIN^N 1$ operation after each step. This is not the usual behavior of marching tests, as the neuromorphic operation is performed in lockstep with the write operation. We formulate this as a loop, which uses the current cell address as an index, sets the cell at the previous address to 0, the next cell to test to 1 and then performs the $MIN^N 1$ operation

which expects a 1 as a result. $MIN^4$4(0111) can consequently be interpreted as a "running 0" and a $MIN^N N$ operation performed after each step. We therefore write the entire array to 1 (W1) and perform the same steps as above with adjusted writing operations and $MIN^N N$ which expects a result of 1.

Assuming that a "running 1/0" write will take 2N write operations (toggle two operands per step), the entire test will have a complexity of 8N+1 (plus one single $MIN^N$1 operation). More specifically, the write complexity is 6N while performing 2N+1 neuromorphic read (inference) operations.

#### 5.3.3.3. Minimal Write Test using Trimming

In the previous section, we showed that the threshold operations are most sensitive in their minimum/maximum threshold operation ($MIN^N$1 and $MIN^N N$). With this in mind, we propose to use resistance trimming to mimic the sense margins of other operations. For example, as shown in Figure 2.8 by decreasing $R_{MIN4}$ to $R_{MIN4L}$, we can emulate a sense margin for $MIN^4$4 operations and operands, which are based on the sensitivity of the $MIN^4$1 operation.

**Algorithm 2** Write Optimized Generic Test Sequence (4N = 3N(W) + 1N(R))

W0, $MIN^N 1L\_0$,
**for** i = 1 to N **do**
  W0(i-1), W1(i), $MIN^N$1_1,
**end for**

By further decreasing the minimum sense margin, which is usually available with trimming circuitry, it is possible to activate particular defects under certain threshold operation and operand values, which are otherwise only detected with different threshold operations under certain operands. This allows us to avoid writing sequences to create other operand values and instead change the sense margin with the same inputs to detect faults. For this purpose, we can sensitize more faults with the $MIN^N$1 operation by reducing the sense margin to half, as indicated by SM0 in Figure 2.8, and eliminate the need for $MIN^N N$ operation and other operand values. The minimum test set can thus be generated with only $MIN^4 1L$(0000), $MIN^4$1(0100) and $MIN^4$1(1000).

A full generic test derived the same way as shown in the previous subsection is shown in Algorithm 2. The test complexity for this algorithm is 4N, with 3N write operations and N+1 neuromorphic read operations. This test is therefore able to reduce the test complexity by another 50% compared to our previous maximal threshold test.

### 5.3.4. Conclusion

In this chapter we investigated the testing of neuromorphic arrays based on resistive memories. We particularly used reference resistance trimming to vary the sense margin of the threshold operations to activate faults only detectable under different threshold operations and operands, to further reduce the test time. With this approach we generated

a test sequence, which is able to detect all single faults in all threshold operations used in neuromorphic-enabled resistive memories.

# 6. Conclusion and Perspectives

STT-MRAM offers many advantages compared to regular memories like SRAM or DRAM. However, using STT-MRAM to perform in-memory operations leads to unique problems, which are not present in SRAM/DRAM based CiM architectures.

Typically, manufacturing test is used to maintain a high quality output of the manufacturing process. Yet, testing takes a lot of resources in terms of time and equipment, so the reduction of either of these two frees up test capacity to be used for CiM-specific testing. In Chapter 3 we presented two approaches to use low temperature test data for trim-setting adjustments and to reduce test time by skipping cells during the trim-search, respectively. Our results show that these two methods are able to remove the need for high temperature test, which unnecessarily put the device-under-test under stress, and to reduce the average test time significantly.

Further, we showed the implementation of an oscillation based in-memory computing concept in Chapter 4. This concept allows for the computation of any symmetric boolean logic function during the runtime, without the restriction of design time decisions. This is in contrast to other concepts, where typically operations are fixed by the chip designer when choosing e.g. appropriate reference resistors.

Chapter 5 first showed the necessity for CiM-specific testing on basic binary operations by injecting resistive faults on cell level before running a CiM operation. It then showed the development of a testing algorithm to find defects based on this resistive defect model, which was able to find all relevant defects. It then considered more complex in-memory calculations using a higher operand number which in turn increased the possible failure space.

## 6.1. Outlook

Currently, STT-MRAM is based on the MTJ as the main storage device. From this, the defect model considered in this thesis has been derived. However, the next generation of spintronic devices, namely Spin-Orbit-Transfer (SOT) devices, which introduce a new writing scheme but still use magnetic tunneling for their resistive evaluation, are already pushing to the center of attention. Many concepts laid out in this thesis are only relying on the magnetic tunneling, so they might also prove valuable for the testing of SOT-MRAM.

In addition, the knowledge gained through the development of a mature manufacturing process of STT-MRAM leads to a more detailed understanding of manufacturing defects. This allows to refine the defect model with defects like tunneling barrier pinholes, which result from an imperfect tunneling layer manufacturing, or sidewall deposition defects. The findings of this thesis can be used as a starting point in the research of these new defect models with respect to in-memory computing.

# Part III.

# Appendix

# Bibliography


[1] S. T. Ahmed, M. Hefenbrock, C. Münch, and M. B. Tahoori. "Neuroscrub: Mitigating retention failures using approximate scrubbing in neuromorphic fabric based on resistive memories". In: *2021 IEEE European Test Symposium (ETS)*. IEEE. 2021, pp. 1–6.

[2] S. T. Ahmed, M. Hefenbrock, C. Münch, and M. B. Tahoori. "NeuroScrub+: Mitigating Retention Faults Using Flexible Approximate Scrubbing in Neuromorphic Fabric Based on Resistive Memories". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022).

[3] S. T. Ahmed, M. Mayahinia, M. Hefenbrock, C. Münch, and M. B. Tahoori. "Process and Runtime Variation Robustness for Spintronic-Based Neuromorphic Fabric". In: *2022 IEEE European Test Symposium (ETS)*. IEEE. 2022, pp. 1–2.

[4] H. Akinaga and H. Shima. "Resistive Random Access Memory (ReRAM) Based on Metal Oxides". In: *Proceedings of the IEEE* 98.12 (2010), pp. 2237–2251.

[5] A. Antonyan, S. Pyo, H. Jung, and T. Song. "Embedded MRAM Macro for eFlash Replacement". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018, pp. 1–4.

[6] M. Bansal, M. Kumar, M. Sachdeva, and A. Mittal. "Transfer learning for image classification using VGG19: Caltech-101 image data set". In: *Journal of Ambient Intelligence and Humanized Computing* (2021), pp. 1–12.

[7] F. Bernard-Granger, B. Dieny, R. Fascio, and K. Jabeur. "SPITT: A magnetic tunnel junction SPICE compact model for STT-MRAM". In: *Proceedings of the MOS-AK Workshop of the Design, Automation & Test in Europe (DATE)*. 2015.

[8] X. Bi, H. Li, and X. Wang. "STT-RAM Cell Design Considering CMOS and MTJ Temperature Dependence". In: *IEEE Transactions on Magnetics* 48.11 (Nov. 2012), pp. 3821–3824. DOI: 10.1109/TMAG.2012.2200469.

[9] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. "Quantum machine learning". In: *Nature* 549.7671 (2017).

[10] D. R. Bild, S. Misra, T. Chantemy, P. Kumar, R. P. Dick, X. S. Huy, Li Shangz, and A. Choudhary. "Temperature-aware test scheduling for multiprocessor systems-on-chip". In: *International Conference on Computer-Aided Design*. 2008, pp. 59–66. DOI: 10.1109/ICCAD.2008.4681552.

[11] R. Bishnoi, F. Oboril, and M. B. Tahoori. "Design of Defect and Fault-Tolerant Nonvolatile Spintronic Flip-Flops". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.4 (Apr. 2017). ISSN: 1063-8210. DOI: 10.1109/TVLSI.2016.2630315.

[12] R. Bishnoi, L. Wu, M. Fieback, C. Münch, S. M. Nair, M. Tahoori, Y. Wang, H. Li, and S. Hamdioui. "Special Session – Emerging Memristor Based Memory and CIM Architecture: Test, Repair and Yield Analysis". In: *2020 IEEE 38th VLSI Test Symposium (VTS)*. 2020, pp. 1–10.

[13] E. M. Boujamaa et al. "A 14.7Mb/mm2 28nm FDSOI STT-MRAM with Current Starved Read Path, 52Σ/Sigma Offset Voltage Sense Amplifier and Fully Trimmable CTAT Reference". In: *IEEE Symposium on VLSI Circuits*. June 2020, pp. 1–2. DOI: 10.1109/VLSICircuits18222.2020.9162803.

[14] C. Braun, S. Halder, and H. J. Wunderlich. "A-ABFT: Autonomous Algorithm-Based Fault Tolerance for Matrix Multiplications on Graphics Processing Units". In: *DSN*. June 2014. DOI: 10.1109/DSN.2014.48.

[15] G. W. Burr et al. "Neuromorphic computing using non-volatile memory". In: *Advances in Physics: X* 2.1 (2017), pp. 89–124. DOI: 10.1080/23746149.2016.1259585.

[16] G. W. Burr et al. "Phase change memory technology". en. In: *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena* 28.2 (2010), pp. 223–262. ISSN: 2166-2746, 2166-2754. DOI: 10.1116/1.3301579.

[17] C.-H. Chen et al. "Reliability and magnetic immunity of reflow-capable embedded STT-MRAM in 16nm FinFET CMOS process". In: *Symposium on VLSI Technology*. 2021, pp. 1–2.

[18] C.-Y. Chen et al. "RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme". In: *IEEE Transactions on Computers* 64.1 (2014), pp. 180–190.

[19] Y. Chen, T. Krishna, J. S. Emer, and V. Sze. "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks". In: *JSSC* 52.1 (2017). ISSN: 0018-9200. DOI: 10.1109/JSSC.2016.2616357.

[20] A. Chintaluri, A. Parihar, S. Natarajan, H. Naeimi, and A. Raychowdhury. "A Model Study of Defects and Faults in Embedded Spin Transfer Torque (STT) MRAM Arrays". In: *IEEE 24th Asian Test Symposium (ATS)*. Nov. 2015. DOI: 10.1109/ATS.2015.39.

[21] A. Chintaluri et al. "Analysis of Defects and Variations in Embedded Spin Transfer Torque (STT) MRAM Arrays". In: *ETCAS* (2016), pp. 319–329.

[22] F. Cüppers, S. Menzel, C. Bengel, A. Hardtdegen, M. von Witzleben, U. Böttger, R. Waser, and S. Hoffmann-Eifert. "Exploiting the switching dynamics of HfO2-based ReRAM devices for reliable analog memristive behavior". In: *APL Materials* 7.9 (2019), p. 091105. DOI: 10.1063/1.5108654.

[23] J. M. Daughton. "Magnetic tunneling applied to memory (invited)". In: *Journal of Applied Physics* 81.8 (1997), pp. 3758–3763. DOI: 10.1063/1.365499. eprint: https://doi.org/10.1063/1.365499. URL: https://doi.org/10.1063/1.365499.

[24] G. Dearth and T. D. Bissett. *DMA controller for memory scrubbing*. US Patent 5,588,112. Dec. 1996.

[25] J. DeBrosse, T. Maffitt, Y. Nakamura, G. Jan, and P.-K. Wang. "A fully-functional 90nm 8Mb STT MRAM demonstrator featuring trimmed, reference cell-based sensing". In: *2015 IEEE Custom Integrated Circuits Conference (CICC)*. 2015, pp. 1–3. DOI: 10.1109/CICC.2015.7338359.

[26] B. Del Bel, J. Kim, C. H. Kim, and S. S. Sapatnekar. "Improving STT-MRAM density through multibit error correction". In: *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2014, pp. 1–6.

[27] Q. Dong et al. "A 1-Mb 28-nm 1T1MTJ STT-MRAM With Single-Cap Offset-Cancelled Sense Amplifier and In Situ Self-Write-Termination". In: *IEEE Journal of Solid-State Circuits* 54.1 (2019), pp. 231–239. DOI: 10.1109/JSSC.2018.2872584.

[28] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam. "ShiDianNao: Shifting vision processing closer to the sensor". In: *ISCA*. June 2015. DOI: 10.1145/2749469.2750389.

[29] H. A. Du Nguyen, L. Xie, M. Taouil, R. Nane, S. Hamdioui, and K. Bertels. "On the implementation of computation-in-memory parallel adder". In: *TVLSI* 25.8 (2017), pp. 2206–2219.

[30] M. Durlam, P. Naji, M. DeHerrera, S. Tehrani, G. Kerszykowski, and K. Kyler. "Non-volatile RAM based on magnetic tunnel junction elements". In: *IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. 2000, pp. 130–131. DOI: 10.1109/ISSCC.2000.839718.

[31] M. Durlam et al. "A 1-Mbit MRAM based on 1T1MTJ bit cell integrated with copper interconnects". In: *IEEE JSSC* (2003), pp. 769–773. ISSN: 1558-173X. DOI: 10.1109/JSSC.2003.810048.

[32] A. Emara et al. "Testing of memristor ratioed logic (MRL) XOR gate". In: *ICM*. 2016, pp. 181–184.

[33] D. Fan and S. Angizi. "Energy Efficient In-Memory Binary Deep Neural Network Accelerator with Dual-Mode SOT-MRAM". In: *ICCD*. 2017.

[34] M. Fieback, C. Münch, A. Grebegiorgis, G. C. Medeiros, M. Taouil, S. Hamdioui, and M. Tahoori. "PVT Analysis for RRAM and STT-MRAM-based Logic Computation-in-Memory". In: *IEEE European Test Symposium*. 2022.

[35] M. Fieback, S. Nagarajan, M. Taouil, and S. Hamdioui. "Testing Computation-in-Memory Circuits". In: *ITC*. 2019, pp. 1–6.

[36] M. Fieback, M. Taouil, and S. Hamdioui. "Testing Resistive Memories: Where are We and What is Missing?" In: *ITC*. 2018, pp. 1–9.

[37] P.-E. Gaillardon, L. Amaru, A. Siemon, E. Linn, R. Waser, A. Chattopadhyay, and G. De Micheli. "The Programmable Logic-in-Memory (PLiM) computer". In: *DATE*. 2016, pp. 427–432.

[38] A. Gebregiorgis and M. B. Tahoori. "Testing of Neuromorphic Circuits: Structural vs Functional". In: *2019 IEEE International Test Conference (ITC)*. 2019, pp. 1–10.

[39] A. Gebregiorgis, L. Wu, C. Münch, S. Rao, M. B. Tahoori, and S. Hamdioui. "Special Session: STT-MRAMs: Technology, Design and Test". In: *2022 IEEE 40th VLSI Test Symposium (VTS)*. IEEE. 2022, pp. 1–10.

[40] S. Hamdioui, M. Fieback, S. Nagarajan, and M. Taouil. "Testing Computation-in-Memory Architectures Based on Emerging Memories". In: *2019 IEEE International Test Conference (ITC)*. 2019, pp. 1–10.

[41] K. Hofmann, K. Knobloch, C. Peters, and R. Allinger. "Comprehensive statistical investigation of STT-MRAM thermal stability". In: *Symposium on VLSI Technology: Digest of Technical Papers*. June 2014, pp. 1–2. DOI: 10.1109/VLSIT.2014.6894367.

[42] H. Honigschmid et al. "Signal-Margin-Screening for Multi-Mb MRAM". In: *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*. 2006, pp. 467–476. DOI: 10.1109/ISSCC.2006.1696079.

[43] K.-H. Huang and J. A. Abraham. "Algorithm-Based Fault Tolerance for Matrix Operations". In: *IEEE Transactions on Computers* C-33.6 (June 1984), pp. 518–528. ISSN: 0018-9340. DOI: 10.1109/TC.1984.1676475.

[44] M. Imani, Y. Kim, and T. Rosing. "Mpim: Multi-purpose in-memory processing using configurable resistive memory". In: *ASP-DAC*. 2017, pp. 757–763.

[45] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan. "Computing in Memory With Spin-Transfer Torque Magnetic RAM". In: *TVLSI* (2018), pp. 470–483.

[46] C.-J. Jhang, C.-X. Xue, J.-M. Hung, F.-C. Chang, and M.-F. Chang. "Challenges and Trends of SRAM-Based Computing-In-Memory for AI Edge Devices". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.5 (May 2021), pp. 1773–1786. ISSN: 1558-0806. DOI: 10.1109/TCSI.2021.3064189.

[47] J.-Y. Jou and J. A. Abraham. "Fault-Tolerant Matrix Operations On Multiple Processor Systems Using Weighted Checksums". In: *Real-Time Signal Processing VII*. Vol. 0495. 1984. DOI: 10.1117/12.944013.

[48] D. Kadetotad et al. "Parallel architecture with resistive crosspoint array for dictionary learning acceleration". In: *JETCAS* 5.2 (2015).

[49] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz. "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM". In: *ICASSP*. 2014. DOI: 10.1109/ICASSP.2014.6855225.

[50] S. Kannan et al. "Sneak-path testing of crossbar-based nonvolatile random access memories". In: *IEEE Trans. Nanotechnol.* (2013).

[51] A. D. Kent and D. C. Worledge. "A new spin on magnetic memories". In: *NNANO* 10 (2015). ISSN: 1748-3387. DOI: 10.1038/nnano.2015.24.

[52] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. "Convergence properties of the Nelder–Mead simplex method in low dimensions". In: *SIAM Journal on optimization* 9.1 (1998), pp. 112–147.

[53] K. Lee et al. "28nm CIS-Compatible Embedded STT-MRAM for Frame Buffer Memory". In: *2021 IEEE International Electron Devices Meeting (IEDM)*. 2021, pp. 2.1.1–2.1.4. DOI: 10.1109/IEDM19574.2021.9720537.

[54] H. Li, P. Huang, B. Gao, X. Liu, J. Kang, and H. .-. Philip Wong. "Device and Circuit Interaction Analysis of Stochastic Behaviors in Cross-Point RRAM Arrays". In: *TED* 64.12 (2017), pp. 4928–4936.

[55] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie. "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories". In: *DAC*. 2016.

[56] X. Liu, D. Mazumdar, W. Shen, B. D. Schrag, and G. Xiao. "Thermal stability of magnetic tunneling junctions with MgO barriers for high temperature spintronics". en. In: *Applied Physics Letters* 89.2 (July 2006), p. 023504. DOI: 10.1063/1.2219997. (Visited on 11/19/2020).

[57] S. Matsunaga, J. Hayakawa, S. Ikeda, K. Miura, T. Endoh, H. Ohno, and T. Hanyu. "MTJ-based Nonvolatile Logic-in-memory Circuit, Future Prospects and Issues". In: *DATE*. 2009, pp. 433–435. ISBN: 978-3-9810801-5-5.

[58] M. Mayahinia, C. Münch, and M. B. Tahoori. "Analyzing and Mitigating Sensing Failures in Spintronic-based Computing in Memory". In: *2021 IEEE International Test Conference (ITC)*. IEEE. 2021, pp. 268–277.

[59] A. Mejdoubi, G. Prenat, and B. Dieny. "A compact model of precessional spin-transfer switching for MTJ with a perpendicular polarizer". In: *MIEL*. 2012.

[60] D. Monroe. "Neuromorphic computing gets ready for the (really) big time". In: *Communications of the ACM* 57.6 (2014).

[61] R. H. Morelos-Zaragoza. *The art of error correcting coding*. John Wiley & Sons, 2006.

[62] S. Mullainathan and J. Spiess. "Machine learning: an applied econometric approach". In: *Journal of Economic Perspectives* 31.2 (2017).

[63] C. Münch and M. B. Tahoori. "Defect Characterization of Spintronic-based Neuromorphic Circuits". In: *2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS)*. 2020, pp. 1–4.

[64] C. Münch, R. Bishnoi, and M. B. Tahoori. "Multi-bit non-volatile spintronic flip-flop". In: *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2018, pp. 1229–1234.

[65] C. Münch, R. Bishnoi, and M. B. Tahoori. "Reliable in-memory neuromorphic computing using spintronics". In: *Proceedings of the Asia and South Pacific Design Automation Conference*. ACM. 2019.

[66] C. Münch, R. Bishnoi, and M. B. Tahoori. "Tolerating retention failures in neuromorphic fabric based on emerging resistive memories". In: *Asia and South Pacific Design Automation Conference*. 2020.

[67] C. Münch, N. Sayed, R. Bishnoi, and M. Tahoori. "A Novel Oscillation-Based Reconfigurable In-Memory Computing Scheme With Error Correction". In: *IEEE Transactions on Magnetics* 57.2 (2020).

[68] C. Münch and M. B. Tahoori. "Testing resistive memory based neuromorphic architectures using reference trimming". In: *Design, Automation & Test in Europe Conference & Exhibition*. 2021.

[69] C. Münch, J. Yun, M. Keim, and M. B. Tahoori. "MBIST-based Trim-Search Test Time Reduction for STT-MRAM". In: *IEEE VLSI Test Symposium*. 2022.

[70] C. Münch, J. Yun, M. Keim, and M. B. Tahoori. "MBIST-supported Trim Adjustment to Compensate Thermal Behavior of MRAM". In: *IEEE European Test Symposium*. 2021, pp. 1–6. DOI: 10.1109/ETS50041.2021.9465383.

[71] H. Naeimi, C. Augustine, A. Raychowdhury, S.-L. Lu, and J. Tschanz. "STTRAM SCALING AND RETENTION FAILURE." In: *Intel Technology Journal* 17.1 (2013).

[72] V. B. Naik et al. "Manufacturable 22nm FD-SOI Embedded MRAM Technology for Industrial-grade MCU and IOT Applications". In: *IEEE International Electron Devices Meeting*. 2019, pp. 2.3.1–2.3.4. DOI: 10.1109/IEDM19573.2019.8993454.

[73] V. B. Naik et al. "STT-MRAM: A Robust Embedded Non-Volatile Memory with Superior Reliability and Immunity to External Magnetic Field and RF Sources". In: *2021 Symposium on VLSI Technology*. 2021, pp. 1–2.

[74] S. M. Nair, R. Bishnoi, M. Tahoori, G. Tshagharyan, H. Grigoryan, and G. Harutyunyan. "Defect Injection, Fault Modeling and Test Algorithm Generation Methodology for STT-MRAM". In: *IEEE International Test Conference (ITC)*. 2018.

[75] S. M. Nair, C. Münch, and M. B. Tahoori. "Defect Characterization and Test Generation for Spintronic-based Compute-In-Memory". In: *2020 IEEE European Test Symposium (ETS)*. 2020.

[76] S. Nasrin, D. Badawi, A. E. Cetin, W. Gomes, and A. R. Trivedi. "MF-Net: Compute-In-Memory SRAM for Multibit Precision Inference Using Memory-Immersed Data Conversion and Multiplication-Free Operators". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.5 (May 2021), pp. 1966–1978. ISSN: 1558-0806. DOI: 10.1109/TCSI.2021.3064033.

[77] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori. "Evaluation of Hybrid Memory Technologies Using SOT-MRAM for On-Chip Cache Hierarchy". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.3 (Mar. 2015). ISSN: 0278-0070. DOI: 10.1109/TCAD.2015.2391254.

[78] V. Parmar and M. Suri. "Design Exploration of IoT centric Neural Inference Accelerators". In: *GLSVLSI*. 2018. DOI: 10.1145/3194554.3194614.

[79] A. L. N. Reddy and P. Banerjee. "Algorithm-based fault detection for signal processing applications". In: *IEEE Transactions on Computers* 39.10 (Oct. 1990), pp. 1304–1308. ISSN: 0018-9340. DOI: 10.1109/12.59860.

[80] D. Reis, M. Niemier, and X. S. Hu. "Computing in memory with FeFETs". In: *ISLPED*. 2018, 24:1–24:6.

[81] J. Reuben et al. "Memristive logic: A framework for evaluation and comparison". In: *PATMOS*. Sept. 2017, pp. 1–8. DOI: 10.1109/PATMOS.2017.8106959.

[82] N. Sayed, S. M. Nair, R. Bishnoi, and M. B. Tahoori. "Process variation and temperature aware adaptive scrubbing for retention failures in STT-MRAM". In: *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2018, pp. 203–208.

[83] V. Seshadri, K. Hsieh, A. Boroum, D. Lee, M. A. Kozuch, O. Mutlu, P. B. Gibbons, and T. C. Mowry. “Fast bulk bitwise AND and OR in DRAM”. In: *IEEE Computer Architecture Letters* 14.2 (2015), pp. 127–131.

[84] V. Seshadri et al. “Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology”. In: *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE. 2017, pp. 273–287.

[85] Y. Shih et al. “Logic Process Compatible 40-nm 16-Mb, Embedded Perpendicular-MRAM With Hybrid-Resistance Reference, Sub- $\mu$ A Sensing Resolution, and 17.5-nS Read Access Time”. In: *IEEE Journal of Solid-State Circuits* 54.4 (Apr. 2019), pp. 1029–1038. DOI: 10.1109/JSSC.2018.2889106.

[86] F. Shoucair. “Design Consideration in High Temperature Analog CMOS Integrated Circuits”. In: *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* 9.3 (1986), pp. 242–251. DOI: 10.1109/TCHMT.1986.1136646.

[87] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. “Relaxing non-volatility for fast and energy-efficient STT-RAM caches”. In: *IEEE 17th International Symposium on High Performance Computer Architecture*. IEEE. 2011.

[88] G. Snider, Appl, and Phys. “Computing with hysteretic resistor crossbars”. In: *Appl. Phys. A* (2005), pp. 1165–1172. ISSN: 0947-8396. DOI: 10.1007/s00339-004-3149-1. URL: https://link.springer.com/content/pdf/10.1007%7B%5C%%7D2Fs00339-004-3149-1.pdf%20http://link.springer.com/10.1007/s00339-004-3149-1.

[89] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky. “Logic design within memristive memories using memristor-aided loGIC (MAGIC)”. In: *IEEE TNANO* 15.4 (2016), pp. 635–650.

[90] T.-L. Tsai, J.-F. Li, C.-L. Hsu, and C.-T. Sun. “Testing of In-Memory-Computing 8T SRAMs”. In: *DFT*. 2019, pp. 1–4.

[91] K. Tsunoda et al. “Area dependence of thermal stability factor in perpendicular STT-MRAM analyzed by bi-directional data flipping model”. In: *IEEE International Electron Devices Meeting*. Dec. 2014, pp. 19.3.1–19.3.4. DOI: 10.1109/IEDM.2014.7047082.

[92] E. I. Vatajelu, L. Anghel, J. M. Portal, M. Bocquet, and G. Prenat. “Resistive and Spintronic RAMs: Device, Simulation, and Applications”. In: *IOLTS*. 2018, pp. 109–114.

[93] A. Velasquez and S. K. Jha. “Parallel boolean matrix multiplication in linear time using rectifying memristors”. In: *IEEE ISCAS*. 2016, pp. 1874–1877. ISBN: 978-1-4799-5341-7. DOI: 10.1109/ISCAS.2016.7538937. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7538937.

[94] L. Wei et al. “13.3 A 7Mb STT-MRAM in 22FFL FinFET Technology with 4ns Read Sensing Time at 0.9V Using Write-Verify-Write Scheme and Offset-Cancellation Sensing Technique”. In: *IEEE International Solid- State Circuits Conference - (ISSCC)*. 2019, pp. 214–216. DOI: 10.1109/ISSCC.2019.8662444.

[95] H.-. P. Wong, H. Lee, S. Yu, Y. Chen, Y. Wu, P. Chen, B. Lee, F. T. Chen, and M. Tsai. "Metal–Oxide RRAM". In: *Proceedings of the IEEE* 100.6 (2012), pp. 1951–1970.

[96] H.-. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson. "Phase Change Memory". In: *Proceedings of the IEEE* 98.12 (2010), pp. 2201–2227.

[97] H.-S. P. Wong and S. Salahuddin. "Memory leads the way to better computing". In: *Nature Nanotechnology* (Mar. 2015), pp. 191–194. DOI: 10.1038/nnano.2015.29.

[98] H. Wu et al. "First Experimental Demonstration of MRAM Data Scrubbing: 80 Mb MRAM with 40 nm junctions for Last Level Cache Applications". In: *2021 IEEE International Electron Devices Meeting (IEDM)*. 2021, pp. 2.3.1–2.3.4. DOI: 10.1109/IEDM19574.2021.9720539.

[99] L. Wu, M. Taouil, S. Rao, E. J. Marinissen, and S. Hamdioui. "Electrical Modeling of STT-MRAM Defects". In: *2018 IEEE International Test Conference (ITC)*. 2018.

[100] L. Xie et al. "Scouting Logic: A Novel Memristor-Based Logic Design for Resistive Computing". In: *ISVLSI*. 2017, pp. 176–181. DOI: 10.1109/ISVLSI.2017.39.

[101] X. Yin, M. niemier, and X. S. Hu. "Design and Benchmarking of Ferroelectric FET Based TCAM". In: *DATE*. Lausanne, Switzerland, 2017, pp. 1448–1453.

[102] S. Yu and P. Chen. "Emerging Memory Technologies: Recent Trends and Prospects". In: *IEEE Solid-State Circuits Magazine* (2016), pp. 43–56. ISSN: 1943-0590. DOI: 10.1109/MSSC.2016.2546199.

[103] J. Yun, B. Nadeau-Dostie, M. Keim, C. Dray, and M. Boujamaa. "MBIST Support for Reliable eMRAM Sensing". In: *2020 IEEE European Test Symposium (ETS)*. 2020, pp. 1–6.

[104] J. Yun, B. Nadeau-Dostie, M. Keim, L. Schramm, C. Dray, M. Boujamaa, and K. Gelda. "MBIST Supported Multi Step Trim for Reliable eMRAM Sensing". In: *2020 IEEE International Test Conference (ITC)*. 2020, pp. 1–5. DOI: 10.1109/ITC44778.2020.9325218.

[105] D. Zhang, L. Zeng, Y. Zhang, W. Zhao, and J. O. Klein. "Stochastic spintronic device based synapses and spiking neurons for neuromorphic computation". In: *Proc. IEEE/ACM Int. Symp. Nanoscale Architectures (NANOARCH)*. July 2016. DOI: 10.1145/2950067.2950105.

# List of Figures

# List of Tables

# A. Acronyms

| | |
|---|---|
| **1T1MTJ** | 1-transistor/1-MTJ |
| **ABFT** | Algorithm-Based Fault Tolerance |
| **ANN** | Artificial Neural Network |
| **AP-state** | Antiparallel-state |
| **BCH** | Bose-Chaudhuri-Hocquenhgem |
| **BE** | Bottom Electrode |
| **BL** | Bitline |
| **CBRAM** | Conductive Bridge Random Access Memory |
| **CCDF** | Complementary CDF |
| **CDF** | Cumulative Distribution Function |
| **CF** | Conductive Filament |
| **CiM** | Compute-in-Memory |
| **CiM-A** | CiM Array |
| **CiM-P** | CiM Periphery |
| **CMOS** | Complementary Metal-Oxide-Semiconductor |
| **CPU** | Central Processing Unit |
| **CUT** | Chip Under Test |
| **dppm** | defective parts per million |
| **DRAM** | Dynamic Random Access Memory |
| **ECC** | Error Correction Code |
| **FBC** | Fail Bit Count |
| **FeFET** | Ferrorelelectric Field Effect Transistor |
| **FIT** | Failure in Time |
| **FL** | Free Layer |
| **FoM** | Figure of Merit |
| **GPU** | Graphics Processing Unit |
| **HRS** | High Resistive State |
| **IANDF** | Incorrect AND Fault |
| **IORF** | Incorrect OR Fault |
| **IRF** | Incorrect Read Fault |
| **LRS** | Low Resistive State |
| **MBIST** | Memory Built-in Self Test |
| **MTJ** | Magnetic Tunneling Junction |
| **NF** | No Fault |
| **NVM** | Non-volatile Memory |
| **OxRAM** | Oxygen-Vacancy based Random Access Memory |
| **PCM** | Phase Change Memory |
| **P-state** | Parallel-state |
| **PDF** | Probability Density Function |
| **PV** | Process Variation |
| **RA** | Resistance-Area |
| **ReRAM** | Resistive Random Access Memory |

| | |
|---|---|
| **RF** | Reference Layer |
| **SL** | Sourceline |
| **SM** | Sense Margin |
| **SOT** | Spin-Orbit-Transfer |
| **SRAM** | Static Random Access Memory |
| **STT-MRAM** | Spin-Transfer-Torque Magnetic Random Access Memory |
| **TE** | Top Electrode |
| **TMR** | Tunneling Magnetoresistance Ratio |
| **WL** | Wordline |