

Evaluation of Memristor-Based Computer Architectures

Bridging the Gap between Devices and Systems

Dissertation

to obtain the academic degree

Doktor-Ingenieur (Dr.-Ing.)

of the Faculty of Computer Science and Electrical Engineering
at the University of Rostock

submitted by

Daniel Günther Reiser

born on October 23, 1995 in Wertheim, Germany

Rostock, October 24, 2025

https://doi.org/10.18453/rosdok_id00005772

Gutachter:

- Prof. Dr.-Ing. Marc Reichenbach, Universität Rostock,
Institut für Angewandte Mikroelektronik und Datentechnik
- Prof. Dr. rer. nat. habil. Christian Wenger, BTU Cottbus-Senftenberg,
Institut für Physik
- Prof. Dr.-Ing. habil. Christian Haubelt, Universität Rostock,
Institut für Angewandte Mikroelektronik und Datentechnik

Jahr der Einreichung: 2025

Jahr der Verteidigung: 2026

Declaration of Authorship

I hereby declare that I have written this thesis independently and entirely on my own and that I have used only the sources and aids indicated. I have clearly indicated any passages in the work, that have been taken from other works or the internet, either verbatim or in substance, by citing the source. This thesis has not previously been submitted for examination in the same or a similar version.

Rostock, October 24, 2025

Daniel Günther Reiser

Acknowledgement

This dissertation developed over several years and across several institutions. I began my doctoral studies in 2019 at Friedrich-Alexander University Erlangen-Nürnberg, continued at BTU Cottbus-Senftenberg, and completed the work at the University of Rostock. I am grateful for the experiences and people who shaped this time.

I would like to express my sincere gratitude to Prof. Dr.-Ing. Marc Reichenbach for his supervision, trust, and guidance. His scientific curiosity, vision, and continuous support have greatly influenced the direction and quality of this work.

I thank my former colleagues from FAU Erlangen-Nürnberg and BTU Cottbus-Senftenberg for their collaboration during the early stages of this research. A special thanks goes to my current colleagues at the Chair of Integrated Systems, University of Rostock, who consistently give their best and still manage to be supportive, patient, and understanding. I have greatly appreciated the openness, discussions, and team spirit within our group.

My appreciation also extends to the colleagues and staff of the Institute of Applied Microelectronics and Computer Engineering, and especially to the institute director, for their continued support and for warmly welcoming our newly established chair. I would further like to acknowledge our project partners, particularly from Fraunhofer Institute for Integrated Circuits IIS, Technical University of Munich, and IHP - Leibniz Institute for High Performance Microelectronics, for the constructive and enjoyable collaborations. In particular, I thank the colleagues from IHP for providing the measurement and characterization data of the RRAM devices used in this work.

Outside the academic environment, I am thankful to my friends and family for their interest, encouragement, and the many moments that provided balance to everyday research. Finally, I would like to thank my parents, my sister, and my girlfriend for their constant support and understanding throughout the years. Your confidence in me made this work possible.

Abstract

Memristive devices are considered a key enabler for future memory and computing architectures, promising high density, non-volatility, and energy efficiency. However, their adoption is challenged by pronounced variability, complex switching dynamics, and limited support for realistic system-level evaluation. Traditional device-level simulators provide accuracy but lack scalability, while behavioral models enable speed but neglect essential device physics. This gap prevents a technology-aware exploration of memristor-based architectures.

This thesis addresses the challenge by introducing the Discrete Stochastic Model (DStM), a novel variability-aware, measurement-driven model for system-level simulation of memristive subsystems. DStM captures both time-invariant and time-dependent variability derived from experimental data while maintaining computational efficiency. Compared to existing system-level models, it achieves up to 84% higher accuracy while offering two orders of magnitude faster simulation speed.

To integrate these models into architectural evaluation, the thesis further presents the Memristor System Simulation (MeSSi) framework, a virtual prototyping platform developed in this work. Built on SystemC, MeSSi provides modular and reusable building blocks for memristor arrays, peripheral circuits, and controllers. This modularity enables the exploration of diverse scenarios, from register files and caches to in-memory computing modules. Through case studies, the thesis demonstrates that incorporating variability fundamentally alters system-level conclusions: strategies that appear reliable under idealized models often fail when realistic distributions are considered. The thesis also explores hardware acceleration of memristor simulations on Field Programmable Gate Arrays (FPGAs). A proof-of-concept shows that stochastic device dynamics can be efficiently offloaded, alleviating simulation bottlenecks and paving the way for large-scale evaluation of memristive systems.

Taken together, the contributions of this thesis establish a new methodology for variability-aware evaluation of memristor-based architectures. By bridging device-level measurements with scalable system-level simulation, this work enables technology-aware design flows that can guide the reliable and efficient integration of memristors into future computing systems.

Zusammenfassung

Memristive Bauelemente gelten als Schlüsseltechnologie für zukünftige Speicher- und Rechnerarchitekturen, da sie eine hohe Speicherdichte, Nicht-Volatilität und Energieeffizienz versprechen. Ihre praktische Nutzung wird jedoch durch ausgeprägte Variabilität, komplexe Schaltvorgänge sowie den Mangel an realistischen Systemsimulationen erschwert. Klassische physikalische Simulationen liefern zwar eine hohe Genauigkeit, sind jedoch kaum skalierbar. Abstrakte Verhaltensmodelle ermöglichen zwar schnelle Simulationen, vernachlässigen dabei jedoch wesentliche physikalische Effekte. Diese Lücke verhindert bislang eine technologiebewusste Bewertung memristorbasierter Architekturen.

In dieser Arbeit wird daher das Discrete Stochastic Model (DStM) vorgestellt, ein neu entwickeltes, messdatengetriebenes Modell für die Systemsimulation von Memristor-Subsystemen, das die Variabilität in deren Verhalten berücksichtigt. DStM erfasst sowohl zeitinvariante als auch zeitabhängige Variabilität auf Basis experimenteller Daten und behält gleichzeitig eine hohe Recheneffizienz bei. Im Vergleich zu bestehenden Modellen für die Systemsimulation erreicht es eine um bis zu 84% höhere Genauigkeit bei einer um zwei Größenordnungen schnelleren Simulation.

Zur Einbindung dieser Modelle in die Architekturbewertung wurde zudem die in dieser Arbeit entwickelte Memristor System Simulation (MeSSi) Plattform für virtuelle Prototypen geschaffen. Sie basiert auf SystemC und stellt modulare, wiederverwendbare Bausteine für Memristor-Arrays, Peripherieschaltungen und Controller bereit. Dadurch lassen sich unterschiedliche Anwendungsfälle untersuchen, von Registerfiles und Caches bis hin zu In-Memory-Computing-Modulen. Die Fallstudien zeigen, dass die Berücksichtigung von Variabilität die Systembewertung grundlegend verändert: Strategien, die unter idealisierten Annahmen zuverlässig erscheinen, versagen häufig bei realistischer Betrachtung. In dieser Arbeit wird zudem die Hardware-Beschleunigung der Memristorsimulation auf Field Programmable Gate Arrays (FPGAs) untersucht. Ein Proof of Concept zeigt, dass sich stochastische Zustandsdynamiken effizient in digitaler Hardware abbilden lassen. Dadurch wird die Simulationsdauer erheblich reduziert und Studien in großem Maßstab ermöglicht.

Die Beiträge dieser Arbeit umfassen somit die Entwicklung des DStM-Modells und der MeSSi-Plattform als neue Methodik zur Evaluierung memristorbasierter Architekturen unter Berücksichtigung von Variabilität. Durch die Verknüpfung von Messdaten und skalierbarer Systemsimulation wird ein technologiebewusster Entwurfsprozess ermöglicht, der die zuverlässige und effiziente Integration von Memristoren in zukünftige Rechnersysteme unterstützt.

Contents

Contents	XI
List of Figures	XIII
List of Tables	XVII
List of Abbreviations	XXI
1 Introduction	1
1.1 Background on Memristors	2
1.2 Research Goals and Contributions	4
1.3 Overview of the Author's Work	6
1.4 Distinction from Related Work	7
1.5 Thesis Structure	8
2 Fundamentals of Memristor Modeling and System-Level Simulation	9
2.1 Overview of the Memory Landscape	9
2.2 Memristor Devices and Applications	12
2.3 Integration of Memristors into Computer Systems	19
2.4 Memristor Modeling Approaches	25
2.5 System-Level Frameworks for Memristors	28
3 A Memristor Model targeted at the System Level	31
3.1 Factors Influencing Device Behavior	31
3.2 The DStM Model Concept	34
3.3 Implementation	39
3.4 Evaluation	43
3.5 Discussion of the DStM	53
4 Memristor-Based System Simulation Framework	55
4.1 Background: SystemC for Virtual Prototyping	56
4.2 Architecture of the MeSSi Platform	57
4.3 Implementation	60
4.4 Modeling Extra-Functional Properties	66
4.5 Summary of the MeSSi Platform	69

5	Case Studies using the Virtual Prototyping Framework	71
5.1	Ultra-Low-Power ECG Classification Architecture for Wearable Devices . . .	72
5.2	Analog In-Memory Computation of Neural Networks	80
5.3	Radiation-Resilient Memory Subsystem	88
5.4	RISC-V CPU with Integrated RRAM	97
5.5	Parameter Study of a Memristive Analog Neuron Circuit	103
6	Conclusion and Outlook	109
6.1	Conclusion	109
6.2	Outlook and Future Work	111
A	Statement on Writing Assistance	I
B	Bibliography	III
C	List of Peer-Reviewed Publications	XIX
D	List of Other Publications	XXI

List of Figures

1.1.1 Definition of the memristor as the fourth passive circuit element including its proposed circuit symbol. Adapted from [A14].	3
1.1.2 Schematic of a typical current-voltage response curve of a memristor.	4
1.1.3 A typical memory hierarchy of a modern computer system with potential areas for memristor integration highlighted. Adapted from [A24].	4
1.4.1 Structural overview of different modeling approaches of memristors with the goals of this thesis highlighted in red.	8
2.1.1 Classification of semiconductor memories. Adapted from [A41, A42].	10
2.2.1 (a) Schematic of a Phase change memory (PCM) device. Adapted from [A9, A52, A67, A77]. (b) Schematic of a Resistive RAM (RRAM) device. Adapted from [A9, A54, A69]. (c) Schematic of a Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM) device. Adapted from [A9, A52, A71, A72]. (d) Schematic of a Ferroelectric Tunnel Junction (FTJ) memory device. Adapted from [A9, A19, A78].	14
2.2.2 Illustration of the three principal modes of memristor operation using a RRAM device as an example: (a) analog mode, (b) binary (1-bit) mode, and (c) multi-level mode. Adapted from [A81].	15
2.2.3 Schematic overview of the <i>discrete storage</i> application of memristors.	17
2.2.4 Schematic overview of the <i>in-memory matrix-vector multiplication</i> application of memristors.	18
2.2.5 Schematic overview of the <i>logic in-memory</i> application of memristors.	18
2.2.6 Schematic overview of the <i>PUF in-memory</i> application of memristors.	19
2.3.1 Conceptual representation of a Y-diagram adapted for memristor-based systems. Adapted from [A106].	21
2.3.2 Qualitative example of cycle-to-cycle variability in memristor current-voltage curves. Adapted from [A128].	25
2.4.1 Qualitative comparison of different memristor models from literature.	26
3.1.1 Overview of key factors that determine memristive device behavior.	32
3.2.1 Overview of the workflow of the DStM.	36
3.2.2 Examples of measured conductance values and fitted normal distributions for multiple target states in IHP 250 nm CMOS-integrated RRAM devices.	37
3.4.1 Comparison of measured and modeled conductance distributions for four conductance states in IHP 250 nm RRAM devices.	45

3.4.2 Comparison of measured and modeled cumulative distributions for IHP 250 nm RRAM devices.	46
3.4.3 Confusion matrices for the classification of four RRAM conductance states: (a) sampled from measurement data, (b) predicted using the DStM model, and (c) predicted using an idealized variability model.	47
3.4.4 Comparison of per-operation simulation speed for the DStM memristor model. Left: Baseline implementation simulating ideal target conductances (no effects active). Right: Full implementation including realistic device variability (<i>variability</i> effect active).	49
3.4.5 Write access simulation speed of all evaluated memristor models as a function of the number of simulated devices.	50
3.4.6 Read access simulation speed of all evaluated memristor models as a function of the number of simulated devices.	51
3.4.7 Simulation speed of a combined access for a single-device simulation.	53
3.4.8 Simulation speed of a combined access for a simulation of 1024 devices.	54
4.0.1 Overview of the SystemC-based MeSSi platform including the DStM.	55
4.2.1 Block diagram of the MeSSi virtual prototyping platform.	58
4.3.1 The principle of discrete storage element operation implemented in MeSSi.	63
4.3.2 The principle of in-memory matrix-vector multiplication using analog MAC operations implemented in MeSSi.	65
5.1.1 Sampling and processing scheme of the continuous ECG signal. Adapted from [B4].	73
5.1.2 Hierarchical structure of the digital accelerator core for convolutional layer inference. Adapted from [B4].	74
5.1.3 Comparison of different power-domain designs for low-duty-cycle signal classification in embedded systems. a) Block diagram of an architecture with a single power-domain. b) Block diagram of the dual power-domain architecture employed for this case study. Adapted from [C1].	76
5.1.4 System-level block diagram of the ECG classification computer architecture. Adapted from [B4].	77
5.2.1 Conductance distributions of three mapping approaches used to obtain a differential state of +50 pS.	82
5.2.2 Block diagram of the in-memory computing platform configuration used in this case study. Adapted from [B6].	83
5.2.3 DNN inference accuracy for different mapping strategies. Adapted from [B6].	85
5.2.4 DNN inference accuracy over time for the most robust mapping strategy under different programming algorithm variants and operating temperatures. Adapted from [B6].	86
5.3.1 TETRISC SoC architecture with the RRAM-based memory bank extension and HiRel Controller structure. Adapted from [B7].	89
5.3.2 Overview of the co-simulation approach, combining memory-domain fault modeling with application-domain reliability analysis. Adapted from [B7].	91
5.3.3 Detailed simulation flow of the radiated RRAM-based memory subsystem within the MeSSi platform.	92

5.3.4 Simulated Bit Error Rate (BER) for different protection schemes and radiation intensity levels. Adapted from [B7].	93
5.4.1 Modeling flow from device measurements to FPGA emulation.	98
5.4.2 Block diagrams of the implemented emulation blocks. Adapted from [B8]. . .	99
5.4.3 Evaluation flow from QEMU-based execution traces to FPGA-based memristor emulation and BER estimation.	100
5.5.1 Coefficient of variation of RRAM conductance variability as a function of the target conductance. Adapted from [B11].	105
5.5.2 Simulation flow for the variability tolerance evaluation of the memristive analog neuron circuit.	106

List of Tables

2.1.1 Comparison of conventional and emerging memory technologies. Adapted from [A9, A21, A52–A56]	11
2.5.1 Comparison of selected system-level memristor frameworks	28
4.3.1 Example mapping from 2-bit input to conductance state index x	62
4.5.1 Key features of the MeSSi platform.	70
5.1.1 Energy consumption per classification window with and without power-down. Adapted from [B4].	77
5.1.2 Relevant MeSSi features for this case study.	79
5.2.1 Relevant MeSSi features for this case study.	87
5.3.1 Parameters of the exponential radiation model $G = a \cdot b^n$, where n is the discrete radiation intensity level index.	93
5.3.2 Relevant MeSSi features for this case study.	96
5.4.1 Average properties of the implemented simulation blocks, based on multiple synthesis runs and simulated number of devices.	99
5.4.2 Relevant MeSSi features for this case study.	102
5.5.1 Relevant MeSSi features for this case study.	108

List of Abbreviations

1T1R	1-transistor-1-RRAM
ADC	Analog-to-Digital Converter
AFib	Atrial fibrillation
AI	Artificial Intelligence
BER	Bit Error Rate
BL	Bitline
CBRAM	Conductive Bridge Random-Access Memory
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
D+N	DNN+Neurosim Model
DAC	Digital-to-Analog Converter
DRAM	Dynamic Random-Access Memory
DStM	Discrete Stochastic Model
ECG	Electrocardiogram
EDA	Electronic Design Automation
ESL	Electronic System Level
FPGA	Field-Programmable Gate Array
FTJ	Ferroelectric Tunnel Junction
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HfO	Hafnium Oxide
HIL	Hardware-in-the-Loop
IGVVA	Incremental Gate Voltage with Verify Algorithm
IHP	IHP GmbH - Leibniz-Institut für innovative Mikroelektronik
IMC	In-Memory Computing
IoT	Internet of Things

ISPVA	Incremental Step Pulse with Verify Algorithm
LFSR	Linear-feedback Shift Register
LID	Linear Ion Drift
LUT	Look-up Table
MAC	Multiply Accumulate
MeSSi	Memristor System Simulation
MVM	Matrix Vector Multiplication
NMC	Near-Memory Computing
NN	Neural Network
OxRAM	Oxide-based Random-Access Memory
PCM	Phase-Change Memory
PE	Processing Element
PUF	Physical Unclonable Function
QMR	Quad-modular Redundancy
RAM	Random-Access Memory
RRAM	Resistive Random-Access Memory
RTL	Register-Transfer Level
SEU	Single-event Upset
SoC	System-on-Chip
SPE	Solar Particle Event
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random-Access Memory
STT-MRAM	Spin-Transfer Torque Magnetoresistive Random-Access Memory
TETRISC	TETra Core System based on RISC-V
TIA	Transimpedance amplifier
TiO	Titanium Oxide
TLM	Transaction Level Model
TMR	Triple-modular Redundancy
UD	User-Defined
VIM	Variability Intensity Model
WL	Wordline
ZnO	Zinc Oxide

Chapter 1

Introduction

In recent years, technological advances have significantly reshaped the way we live and work. Automation, artificial intelligence (AI), and the Internet of Things (IoT), a vast network of interconnected smart devices, have become integral components of modern society [A1]. Tasks that previously required human intervention are now increasingly handled autonomously by machines, significantly enhancing industrial productivity and streamlining daily life. Technologies such as autonomous driving systems, smart home devices, voice assistants, and health trackers exemplify how widespread this shift has become. However, the rapid growth and complexity of these innovations have placed immense demands on the underlying computing hardware and architectures [A2].

Implementing these advanced technologies necessitates computer systems characterized by high performance, energy efficiency, and scalability. Traditional computing architectures, however, increasingly struggle with meeting these demands. For instance, AI applications require substantial computational power to handle intricate tasks, while IoT environments emphasize the importance of low-power operation and real-time data processing. Furthermore, these systems generate vast volumes of data, necessitating rapid, efficient processing capabilities. Conventional architectures, limited by inherent constraints such as power consumption, memory bandwidth, and processing speed, are thus inadequate for future needs. Consequently, new computing solutions capable of overcoming these limitations and providing superior performance are urgently required [A3, A4].

To address these challenges, researchers are actively exploring alternative computing paradigms, each designed to mitigate specific shortcomings of conventional systems [A5]. Dataflow-driven architectures, for example, enhance efficiency by emphasizing optimized data movement rather than traditional instruction-driven processing [A6]. Near-memory computing (NMC) and in-memory computing (IMC) significantly reduce latency and energy consumption by performing computations directly within or close to memory units, thus minimizing data movement [A7]. Normally-off computing paradigms exploit idle phases to completely power down systems, dramatically reducing overall energy consumption [A8]. Among various emerging technologies facilitating these novel paradigms, memristors have garnered particular attention [A9].

Memristors, broadly understood as devices whose resistance depends on their electrical history, stand out not merely as an incremental improvement over existing memory devices, but as a fundamentally new circuit element that enables a shift in how computation itself can be organized [A10]. Just as the transistor reshaped the trajectory of electronics, the memristor combines memory and logic in a single device, collapsing traditional boundaries between storage and processing [A11]. This unique property directly addresses the von Neumann bottleneck and opens the door to computing paradigms that are qualitatively different from those possible with conventional memories [A12].

Memristors, characterized by their non-volatility, scalability, high-density storage capability, and inherently energy-efficient operation, present a promising path toward realizing advanced computing architectures. Their unique combination of properties directly supports the practical implementation of IMC, NMC, and normally-off computing, making them strong candidates as foundational components for the next generation of computing systems, particularly those addressing the intensive requirements of AI and IoT applications [A13]. However, fully leveraging this potential requires not only advances in fabrication of such devices, but also new *methodologies for evaluating and integrating memristors into complete computing systems*. Addressing this gap is the central motivation of this thesis.

1.1 Background on Memristors

The modern scientific interest in memristors began with their theoretical introduction by Leon Chua in 1971 as the fourth fundamental circuit element alongside resistors, inductors, and capacitors [A10]. Chua defined the memristor as a device establishing a direct relation between electric charge and magnetic flux. This completed the symmetry of circuit theory. This definition together with the proposed circuit symbol is shown in Figure 1.1.1. Unlike conventional passive elements, the memristor is characterized by a state-dependent resistance, termed *memristance*, which depends on the history of electrical excitation.

Interest in memristors surged after the work of Strukov et al. [A14], who reported a resistive switching device in titanium dioxide and explicitly identified it as a practical realization of Chua’s memristor. These devices display a distinctive *pinched hysteresis loop* in their current-voltage (i - v) characteristics, which is widely regarded as a hallmark signature of memristive behavior (Figure 1.1.2). However, this identification sparked an ongoing and sometimes heated debate. While some researchers embraced resistive random-access memory (RRAM) as the long-sought fourth circuit element, others rejected this view, arguing either that such devices do not fulfill Chua’s original definition [A15] or that the memristor itself does not merit recognition as a fundamental circuit element in the first place [A16]. In this latter perspective, the memristor is seen not as a missing building block of circuit theory, but as a derived, composite behavior of existing elements.

To reconcile theory and experiment, Chua himself later generalized the concept in the form of *memristive systems* [A17], defined as a broader class of devices whose resistance depends on past electrical activity, without requiring strict adherence to the original charge-flux relation. Within this more general framework, practical devices such as RRAM [A14], phase-change memory (PCM) [A18], and ferroelectric tunnel junctions (FTJ) [A19] can all be considered memristive systems.

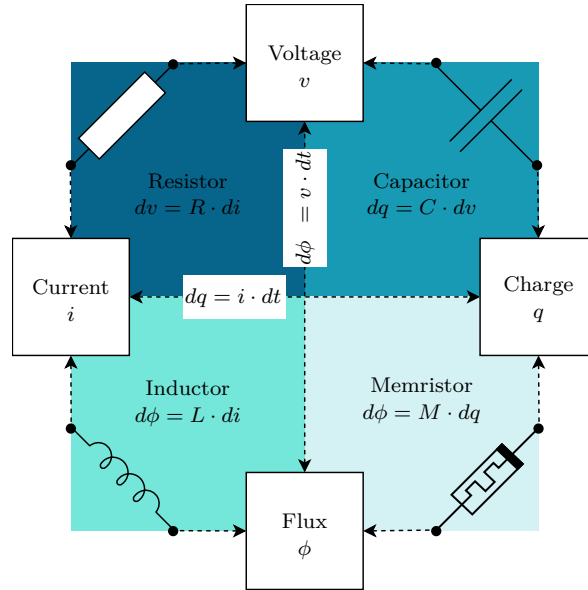


Figure 1.1.1: Definition of the memristor as the fourth passive circuit element including its proposed circuit symbol. Adapted from [A14].

As a result, the terminology in the literature is inconsistent. In some communities, the term *memristor* is used almost synonymously with resistive switching devices (i.e., RRAM). In others, it is employed more broadly as an umbrella term for all devices that exhibit memristive behavior (memristive systems), encompassing PCM, FTJ, and RRAM alike. This definition is supported by Chua, who states, that “Resistance switching memories are memristors” [A20]. Throughout this thesis, the latter convention is adopted: **the term “memristor” is used to denote the general category of devices with memory-dependent resistance**, while specific technologies will be explicitly referred to by their individual names.

This broader view emphasizes that memristors represent not merely a single device type, but an entire class of memory technologies that share the ability to co-locate storage and computation. They have generated significant attention because of their potential to address limitations inherent to conventional memories [A9, A21]. This becomes apparent when looking at a conventional memory hierarchy of computing systems depicted in Figure 1.1.3. Unlike traditional memory technologies such as Static Random Access Memory (SRAM), Dynamic Random Access Memory (DRAM), or Flash memory, which are optimized for specific roles within distinct hierarchy levels, memristors offer the potential to span multiple hierarchy levels simultaneously. They can achieve a favorable combination of speed, non-volatility, scalability, density, and local integrability [A22]. Moreover, their compatibility with established semiconductor CMOS fabrication processes facilitates seamless integration into existing semiconductor manufacturing workflows [A23]. These traits make them ideally suited for innovative computing paradigms like in-memory computing, near-memory computing, and normally-off computing architectures. Collectively, these properties position memristors as disruptive elements that could reshape or even collapse the traditional separation between computation units and memory.

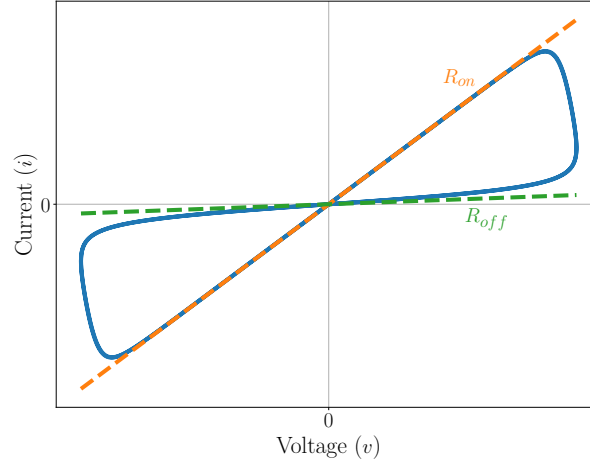


Figure 1.1.2: Schematic of a typical current-voltage response curve of a memristor.

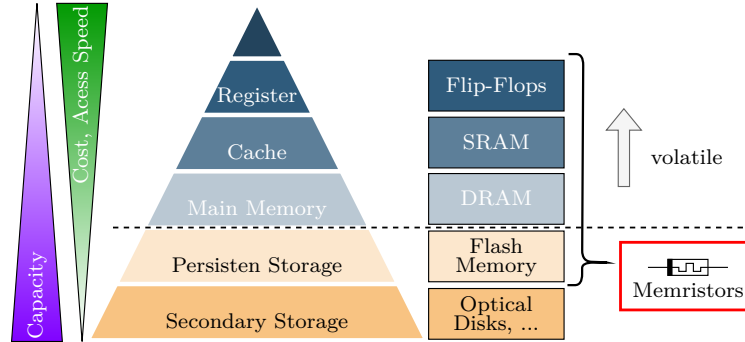


Figure 1.1.3: A typical memory hierarchy of a modern computer system with potential areas for memristor integration highlighted. Adapted from [A24].

However, despite these compelling advantages, memristor technology still faces substantial challenges [A25]. Variability, both intrinsic and due to fabrication inconsistencies, poses reliability concerns [A26]. Additionally, standardized modeling methods and well-developed integration practices are currently lacking, complicating the design process of computer systems including memristors [A27]. Finally, the atomic-scale dynamics underlying memristive behavior are not yet fully understood [A28, A29]. Addressing these challenges is crucial for the widespread adoption and practical application of memristor-based technologies.

1.2 Research Goals and Contributions

The integration of memristor-based devices into modern computing architectures promises substantial improvements in energy efficiency, non-volatile storage capabilities, and the realization of novel computing paradigms. However, the absence of fast, accurate, and variability-aware device models, together with the lack of flexible system-level evaluation tools, has

hindered realistic architectural evaluation and slowed adoption. This thesis addresses these challenges by defining and pursuing two main research goals:

1. **Goal 1:** Develop and implement a computationally efficient, statistically accurate memristor model derived directly from empirical measurements, capturing device variability under diverse environmental and programming conditions.
2. **Goal 2:** Create a flexible, modular evaluation framework, comprising a SystemC-based virtual prototyping platform and real-time FPGA IP blocks, that supports end-to-end co-simulation of memristor-based systems, from device behavior to complete architectures.

The primary scientific contributions of this thesis are as follows:

C1: High-Speed, Measurement-Driven System-Level Memristor Model

As a foundation for the modeling work, a comprehensive review and analysis of the key factors influencing memristive device behavior is first conducted. The insights gained from this study are systematically incorporated into the subsequent model design to ensure physical relevance and general applicability. Building upon this groundwork, a novel empirical, variability-aware memristor model called *DStM* is introduced, achieving up to two orders of magnitude speedup over the fastest state-of-the-art model under realistic conditions, while delivering statistical accuracy improvements of up to 84% compared to existing system-level approaches. The model derives its parameters directly from measured device data across operating conditions and programming algorithms, without reliance on a priori physical assumptions or manual parameter fitting. It is readily extensible to new devices through straightforward updates to measurement datasets, and is compatible with system-level simulation environments.

C2: Modular, Technology-Aware Virtual Prototyping Platform

Building upon the proposed model, a generic and extensible virtual prototyping platform called *MeSSi* is developed in SystemC, supporting various operation principles of the memristors. Its modular architecture enables the integration of diverse memristor types, peripheral circuits, and digital components, and supports both detailed and abstract simulation. Beyond functional correctness, the platform supports configurable modeling of extra-functional properties at the system level, including energy and latency estimation, enabling comprehensive evaluation of technology-architecture trade-offs.

C3: Validation through Case Studies and Hardware Acceleration Concepts

The proposed modeling and evaluation methodology is validated through representative case studies that span different operating principles and application scenarios. These studies demonstrate how incorporating device variability fundamentally changes system-level conclusions, highlighting the necessity of variability-aware evaluation for realistic design. In addition, hardware acceleration concepts are introduced as a proof-of-concept to alleviate simulation bottlenecks and enable scalable exploration of larger systems. Together, these validation efforts confirm both the accuracy and the practical relevance of the presented contributions.

Collectively, the contributions C1-C3 establish a coherent methodology and toolchain for technology-aware design of memristor-based computing systems. By bridging device-level characterization with full-system simulation and hardware-in-the-loop evaluation, this thesis enables architects to explore realistic designs, quantify performance-energy trade-offs, and accelerate the transition from device research to system deployment. These results lay a robust foundation for ongoing and future research in memristor-based computing architectures.

1.3 Overview of the Author's Work

The simulation of variability in memristors within computer architecture design was initially explored for oxide thickness variability in RRAMs (a specific type of memristor) by Johannes Knödtel, Markus Fritscher, Dietmar Fey, Marco Breiling, Marc Reichenbach, and the author of this thesis [B1, B2] and later continued by Fritscher et al. [A30]. That work focused on a systolic-array-based digital accelerator for electrocardiogram classification using neural networks with integrated local RRAM memory. The author of the thesis at hand contributed the accelerator architecture concept and developed the framework for parametrizing the architecture and mapping operations to hardware. Follow-up studies on the RRAM memory blocks used inside the architecture [B3] and the framework for developing the accelerator, utilizing normally-off computing, were later published [B4]. The systolic-array-based accelerator was also implemented on an FPGA to evaluate memory throughput on high-bandwidth-memory [B5].

The concept of a measurement-based, technology-aware memristor model for system-level design was developed by the author in collaboration with Marc Reichenbach, Tommaso Rizzi, Andrea Baroni, Markus Fritscher, Christian Wenger, Cristian Zambelli, and Davide Bertozzi. This approach was first presented as a discrete, technology-aware framework covering different operating and environmental conditions of RRAM devices [B6]. The implementation and evaluation of this model were carried out by the author, with device measurements and statistical parameter extraction performed by Andrea Baroni and Tommaso Rizzi at *IHP GmbH - Leibniz-Institut für innovative Mikroelektronik*. This model was subsequently incorporated by the author of the thesis at hand into a SystemC-based virtual prototyping platform, enabling estimation of extra-functional properties [B7]. Integration with other system components was jointly explored with Junchao Chen (system-on-chip platform with adaptive error correction for space environment) and Johannes Knödtel (CPU instruction set architecture simulation, fault injection, and analysis). Measurement data for these studies was provided by Andrea Baroni.

Building on this foundation, the modeling concept was later used by the author of the thesis at hand to realize real-time FPGA-based memory simulation blocks [B8]. These were applied to simulate a RISC-V CPU with RRAM-based register files and caches. The simulation block concept was developed together with Johannes Knödtel, Liliia Almeeva, Jianan Wen, Miloš Krstić, and Marc Reichenbach, with implementation and evaluation led by the author and support from Liliia Almeeva. Error rate evaluation for the RISC-V CPU was performed by Johannes Knödtel. Another measurement-based model was used to implement an FPGA-based IMC simulation platform, including programming variability [B9, B10]. The author of the thesis at hand helped with the conceptualization and writing of that work.

The simulation framework by the author of this thesis has also been employed to predict RRAM device variability under varying environmental conditions for an analog neuron circuit study at discrete operating points [B11]. Here, the author conceptualized and developed the variability prediction model, while Sebastian Jenderny contributed the analog neuron circuit model and conducted simulation and evaluation.

Finally, the concept of the measurement-based modeling approach was also applied in further works by Fritscher et al., including a RISC-V accelerator core [A31], fault-aware neural network training [A32], and RRAM-based CMOS standard cell modeling for a RISC-V CPU design [B12]. In the latter work, the author contributed to the conceptualization of the underlying modeling approach.

1.4 Distinction from Related Work

Memristor models can be broadly classified into three methodological categories. *Analytical models* emphasize fundamental physical mechanisms and aim for high fidelity at the device or material level [A33]. *Empirical models*, in contrast, abstract from measured device behavior and focus on reproducing observed input-output characteristics [A33]. *Compact models* combine these perspectives by embedding empirical assumptions into simplified, physics-inspired formulations, making them particularly suitable for circuit-level design and simulation [A27]. It should be noted that this taxonomy is not unique; some works distinguish, for example, between physics-based versus phenomenological approaches, or deterministic versus stochastic formulations. The classification adopted here, however, provides a balanced view that highlights the methodological distinctions most relevant for this thesis. Figure 1.4.1 provides an overview of these approaches and situates the modeling path pursued here. While many prior models emphasize continuous, physics-based formulations suited to device or circuit-level simulations, the present work adopts a discrete, functionality-driven modeling strategy explicitly tailored to efficient system-level evaluation.

Several system-level simulation frameworks for memristors have been proposed in prior literature [A33]. Nevertheless, existing frameworks typically exhibit two significant limitations:

1. They often rely on analytical models originally developed for device-level accuracy, resulting in substantial performance overhead and slower simulations at the system level [A27, A34]. Alternatively, they employ idealized models characterized by generic and simplified behaviors, lacking the realism needed for accurate system-level evaluations [A35, A36].
2. Most frameworks are designed around specific applications, imposing constraints on the architectures they can evaluate, thus limiting comprehensive design-space exploration [A37–A40].

In contrast, the platform introduced in this thesis uniquely combines a discrete, empirical modeling approach specifically optimized for rapid system-level simulation and flexible evaluation. It offers a universal and modular framework capable of seamless integration and co-simulation with other system-level components, enabling realistic behavior representation without sacrificing simulation performance or architectural versatility.

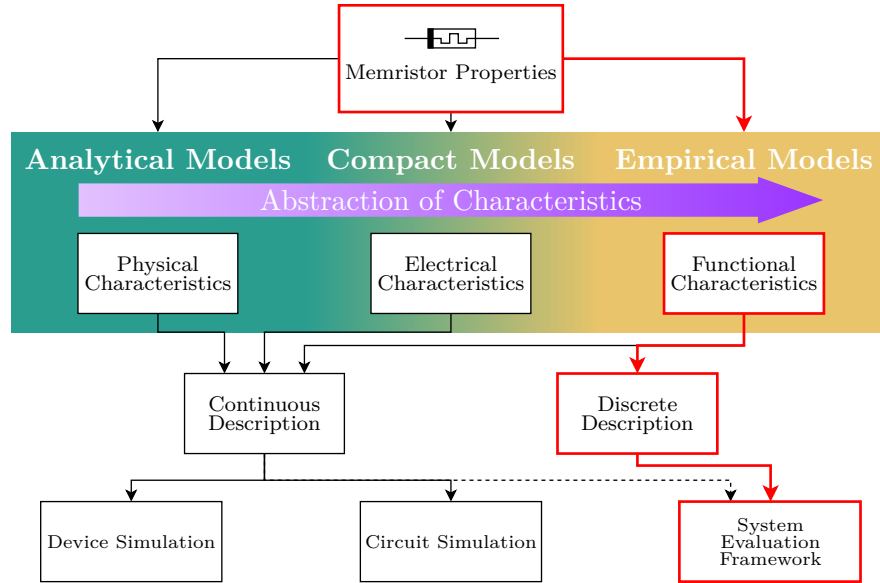


Figure 1.4.1: Structural overview of different modeling approaches of memristors with the goals of this thesis highlighted in red.

1.5 Thesis Structure

This thesis systematically develops and evaluates the modular virtual prototyping framework for memristor-based computer architectures. The remainder of the thesis is structured as follows:

- **Chapter 2** introduces the fundamentals of memristive technologies, addressing their key characteristics, modeling challenges, and integration issues in modern computer architectures.
- **Chapter 3** presents the novel discrete, empirical modeling approach developed specifically for efficient and accurate system-level simulations.
- **Chapter 4** details the modular virtual prototyping framework, demonstrating how the developed discrete model is seamlessly integrated and leveraged for rapid system-level evaluation.
- **Chapter 5** evaluates the framework through practical case studies, highlighting its effectiveness and demonstrating the advantages of employing the proposed modeling approach in realistic scenarios.
- **Chapter 6** summarizes the contributions of this thesis and discusses future research opportunities arising from the presented work.

Chapter 2

Fundamentals of Memristor Modeling and System-Level Simulation

Having outlined the motivation and core contributions of this thesis, this chapter provides essential foundational knowledge necessary to understand memristors and their integration into computer architectures. Initially, fundamental concepts of memristive devices, including their defining properties, application areas and modeling methodologies, are introduced. Subsequently, existing approaches to system-level simulation and evaluation platforms from relevant literature are examined. This discussion sets the groundwork to contextualize and highlight the significance of the contributions presented in later chapters.

2.1 Overview of the Memory Landscape

The diversity of semiconductor memory technologies reflects decades of innovation aimed at balancing speed, density, energy efficiency, and cost. To set the stage for the comparison of conventional and memristive memories pursued in this thesis, first a classification of existing technologies by their underlying principles, is provided in Figure 2.1.1.

The landscape can broadly be divided into volatile and non-volatile categories. Within these categories, individual technologies are grouped by their operational mechanisms, such as charge storage, phase-change, magnetism, ferroelectricity, or resistance switching. Importantly, as discussed in Section 1.1, several of the emerging memory technologies in this classification can be described under the umbrella term *memristors*, i.e., devices whose resistance state can be modified and retained.

This thesis concentrates on the comparison of two main sets of technologies: (1) the most prominent conventional memories: **SRAM**, **DRAM**, and **Flash**, which form the foundation of today's computing systems. And (2) the leading emerging memories that fall into the memristor class: most notably **Phase-Change Memory (PCM)**, **Resistive RAM (RRAM)** (with its variants oxide-based resistive RRAM (OxRAM) and conductive-bridge

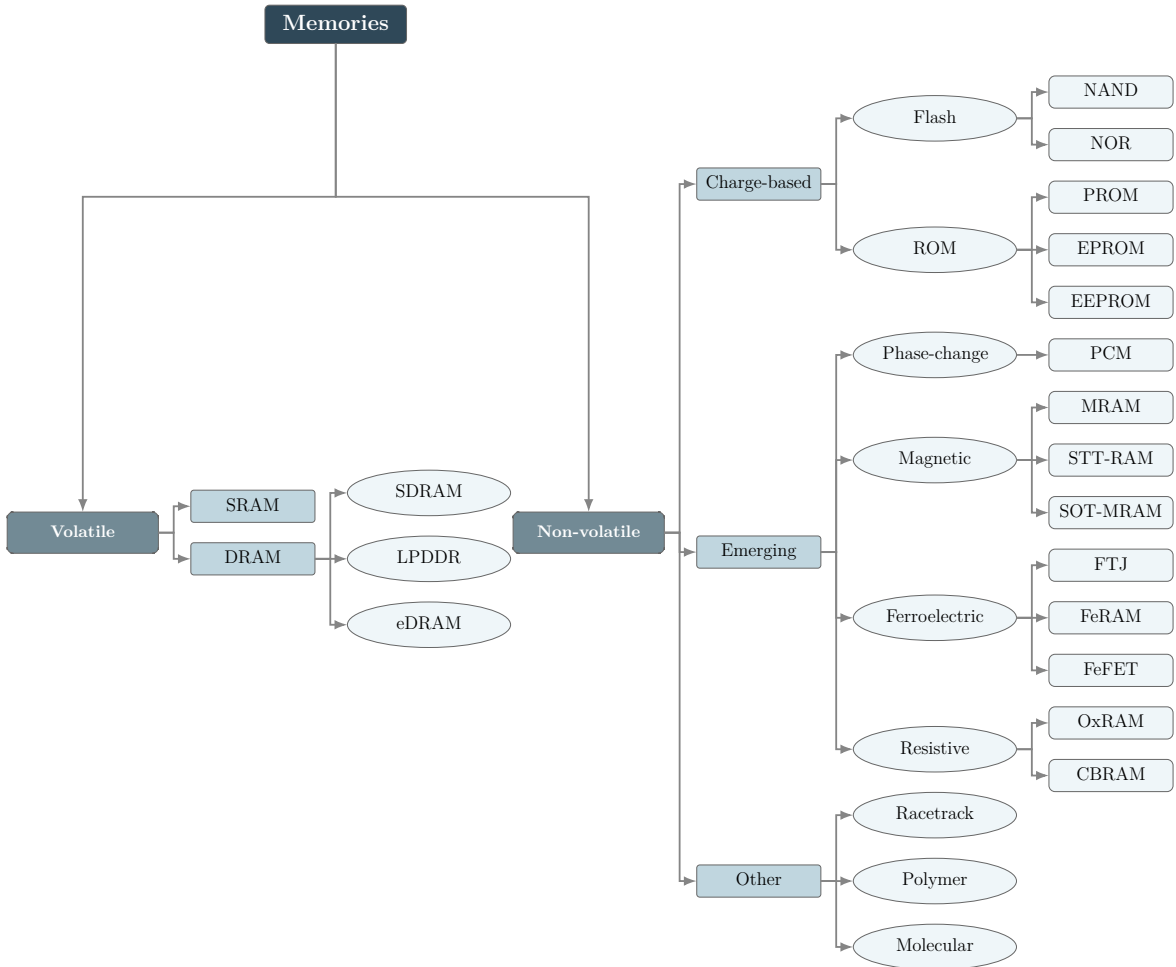


Figure 2.1.1: Classification of semiconductor memories. Adapted from [A41, A42].

RAM (CBRAM)), **Spin-Transfer Torque MRAM (STT-MRAM)**, and **Ferroelectric Tunnel Junctions (FTJ)**. These candidates are of particular interest due to their potential to combine high density, scalability, and non-volatility with performance characteristics approaching or even surpassing those of conventional random access memory (RAM). To facilitate clarity and consistency in the subsequent discussions, it is worth noting that throughout this thesis, **OxRAM** (oxide-based resistive RAM) and **CBRAM** (conductive-bridge RAM) are collectively referred to under the general term **RRAM** (Resistive Random Access Memory). While these two technologies differ in their underlying switching mechanisms (ionic movement of oxygen vacancies in oxides versus metallic filament formation in electrolytic materials), they share the same fundamental operating principle of resistive switching between distinct conductive states. Consequently, their device behavior, modeling approaches, and system-level implications are closely aligned. Treating them jointly as RRAM simplifies the terminology without loss of generality for the analyses presented in this work.

The figure also includes a number of other memory technologies, such as Ferroelectric RAM (FeRAM) [A43], early FeFET devices [A44], racetrack memory [A45], polymer memories

[A46], and molecular memories [A47]. While each of these offers unique advantages (e.g., FeRAM offers fast access times [A43] or racetrack memory provides a novel shift-register structure [A45]), their widespread adoption has so far been limited by scaling challenges, endurance issues, or integration complexity [A43, A48–A50]. Since the methodology developed in this thesis requires devices that can be reliably fabricated and experimentally characterized, the discussion will focus on memristive technologies that are currently mature enough to be considered for system-level integration.

First the conventional memories that dominate current architectures are reviewed. **SRAM** is primarily deployed in cache hierarchies due to its fast access and high endurance, though its relatively large cell size and static power consumption limit its density and efficiency [A51, A52]. **DRAM**, as the workhorse of main memory, achieves higher density and lower cost but requires refresh operations and faces scaling limitations [A51, A52]. **Flash**, derived from earlier ROM variants, dominates non-volatile storage with high density and persistence, though its relatively slow access, limited endurance, and integration challenges confine it to storage roles [A51, A52]. While conventional memories form the backbone of current architectures, they face intrinsic scaling and energy limitations. This has motivated the exploration of emerging memories, many of which can be unified under the memristor concept.

Consequently, emerging memories have attracted significant attention as possible next-generation solutions. **PCM**, **RRAM**, **STT-MRAM**, and **FTJ** devices offer non-volatility, energy efficiency, and high scalability, making them prime candidates for integration into future memory hierarchies and for enabling novel computing paradigms [A52]. Their comparative characteristics relative to conventional technologies are summarized in Table 2.1.1. Here qualitative indicators (L(ow)/M(edium)/H(igh) or + to +++) denote relative rankings across technologies based on literature surveys. Beneficial properties are highlighted in green, while non-beneficial properties are colored yellow and orange. It should be noted that the values presented in the table represent typical values found in the literature [A9, A21, A52–A56]. However, ongoing research, especially for emerging memories, but also continued innovation in conventional technologies, can lead to significant deviations from these typical characteristics.

Table 2.1.1: Comparison of conventional and emerging memory technologies. Adapted from [A9, A21, A52–A56]

Technology	Volatile	Density	Integration	Speed		Endurance	Power		Variance
				R	W		Static	W	
SRAM	yes	+	+++	H	H	H	H	L	L
DRAM	yes	++	++	M	M	H	M	L	L
NOR Flash	no	++	++	M	L	L	L	H	M
NAND Flash	no	+++	+	L	L	L	L	H	M
PCM	no	++++	++	M	L	M	L	M	H
RRAM	no	++++	+++	H	M	M	L	M	H
STT-MRAM	no	+++	++	H	M	H	L	M	H
FTJ	no	+++	++	H	M	M	L	M	H

Memristors, owing to their nanoscale dimensions and capability to store multiple discrete resistance states, can achieve significantly higher storage densities than conventional memory technologies [A52]. In particular, PCM [A57], RRAM [A54], and certain STT-MRAM [A58]

as well as ferroelectric devices [A59] exhibit multilevel programming capabilities, enabling multiple bits to be stored per cell within a compact device footprint. Their promising compatibility with Complementary Metal-Oxide-Semiconductor (CMOS) fabrication processes further supports the integration of data storage directly adjacent to processing units, paving the way for in-memory and neuromorphic computing architectures that are infeasible with traditional memory technologies [A18, A54, A55, A60]. Unlike SRAM or DRAM, which are volatile and require continuous power to retain information, memristive devices provide intrinsic non-volatility, ensuring data persistence even in the absence of power. Consequently, memristors uniquely combine the functional roles of working memory and storage, offering fast, localized, and energy-efficient data access.

However, variability remains a critical issue across most memristive technologies, manifesting in both time-independent and time-dependent forms [A27, A28, A61]. Time-independent variability primarily originates from manufacturing-related inconsistencies, including deviations in device geometry, material composition, and defect distributions. In some device classes, such as RRAM, additional variability arises during the electroforming process, which initializes a conductive path through the active layer [A29]. These imperfections can also lead to hard device failures, such as cells that are permanently stuck in one resistive state or become non-functional after fabrication [A62]. In contrast, time-dependent variability reflects stochastic and dynamic phenomena that occur during device operation. These include random telegraph noise, stochastic filament formation and rupture, drift or relaxation of resistance states, and gradual material degradation under electrical stress [A22, A61]. The combined effects of these mechanisms lead to a substantial spread in the achievable resistance levels. This variability (both intrinsic and dynamic) represents one of the primary challenges to achieving reliable and large-scale deployment of memristive technologies. Mitigation strategies such as redundancy, error correction, and adaptive programming algorithms are commonly employed to alleviate its impact on system-level performance. Another major challenge concerns device programming. Many memristive technologies require program-and-verify sequences, elevated voltage levels, or reset-before-write procedures to achieve precise resistance control [A63–A65]. These requirements increase both latency and energy consumption. Consequently, the read and write speed, energy efficiency, and endurance of memristive devices are strongly dependent on their underlying materials, switching mechanisms, and peripheral circuit designs. Unlike mature volatile memories such as SRAM and DRAM, for which performance metrics are well standardized, memristor-based devices exhibit a broad diversity of operational characteristics [A66]. This diversity compels system designers to carefully balance density, speed, energy consumption, and reliability when integrating memristors into emerging computing architectures.

2.2 Memristor Devices and Applications

Memristive devices encode information by modulating their resistance states in response to applied electrical stimuli, making them promising candidates for various computing applications. To effectively integrate memristive devices into system-level designs, an understanding of their fundamental operational principles and distinctive properties is essential. This section provides an overview of the prominent emerging non-volatile technologies, Phase-Change Memory (PCM), Resistive RAM (RRAM), Spin-Transfer Torque MRAM (STT-MRAM), and

Ferroelectric Tunnel Junctions (FTJ), each representing a specific technological realization of memristive behavior, exploiting distinct physical or chemical mechanisms to control their resistance state. Figure 2.2.1 shows a schematic overview of the operating principles of the four types including an indicator, which property is proportional to the conductance G of the device. In the following subsections, these technologies are discussed as representative implementations of memristive systems, illustrating the diverse physical mechanisms through which resistance can depend on prior electrical history.

Phase-Change Memory (PCM) encodes information by exploiting reversible phase transitions between amorphous (high-resistance) and crystalline (low-resistance) states in chalcogenide alloys. Rapid, high-current pulses induce localized melting followed by rapid quenching to form the amorphous state, whereas lower-intensity, longer pulses allow recrystallization. PCM provides moderate latency, excellent scalability, and inherent non-volatility. Notably, PCM cells exhibit characteristic pinched hysteresis loops in their current-voltage response due to changes in the crystalline fraction during programming, highlighting their memristive nature. However, challenges include managing their higher programming energy and resistance drift over time, which affect data stability [A18, A57, A67, A68].

Resistive Random-Access Memory (RRAM) represents one of the most widely studied implementations of memristive behavior, utilizing the reversible formation and rupture of nanoscale conductive filaments within a dielectric medium to modulate resistance. Variants such as Oxide-based RRAM (OxRAM) and Conductive-Bridge RAM (CBRAM) differ primarily by filament material: oxygen vacancy chains in OxRAM and metallic filaments (formed via metal ion migration) in CBRAM. RRAM offers rapid switching, lower operating voltages, scalability, and multilevel storage capabilities. However, the stochastic nature of the filament formation process is a key characteristic of this particular memristor class. It introduces significant variability, requiring careful statistical modeling approaches to accurately simulate realistic device behavior at the system level [A9, A21, A54, A69, A70].

Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM) stores data through magnetic tunnel junctions (MTJs), where the resistance state depends on the relative magnetic orientation of ferromagnetic layers. Switching between these states is achieved via spin-polarized currents that transfer angular momentum without external magnetic fields. STT-MRAM features low latency, high endurance, and non-volatility, positioning it as a potential replacement for traditional caches and main memory elements [A9, A21, A52, A71, A72]. Although STT-MRAM's resistance switching mechanism differs fundamentally from filament-based or phase-change mechanisms, it aligns with the generalized definition of memristive systems adopted in this thesis due to the state-dependent resistance governed by magnetic history [A73].

Ferroelectric Tunnel Junctions (FTJ) employ ferroelectric polarization modulation within ultrathin insulating layers sandwiched between conductive electrodes. The resistance state depends on tunneling probabilities influenced by polarization orientation, thereby exhibiting memristive switching behavior [A59, A74–A76]. FTJs provide exceptionally low

switching energy, compact cell sizes, high ratios between conducting and non conduction states (On-Off-ratio), and ultrafast switching.

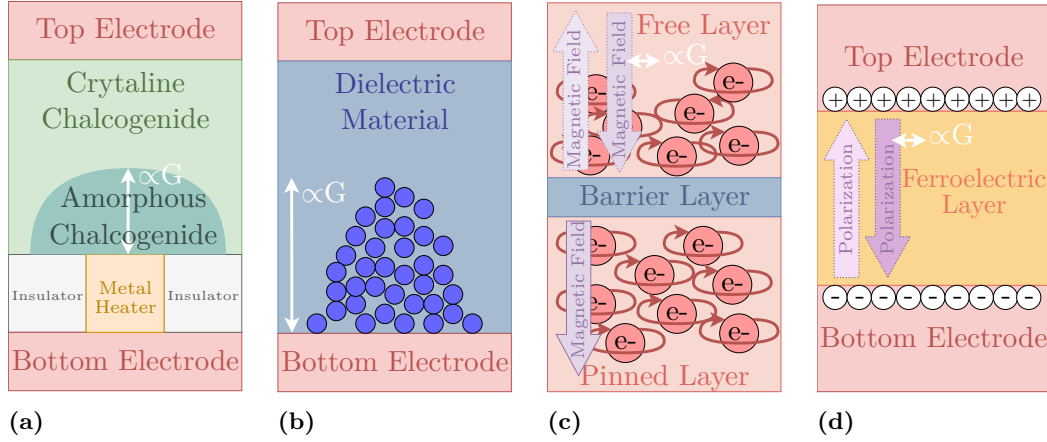


Figure 2.2.1: (a) Schematic of a Phase change memory (PCM) device. Adapted from [A9, A52, A67, A77]. (b) Schematic of a Resistive RAM (RRAM) device. Adapted from [A9, A54, A69]. (c) Schematic of a Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM) device. Adapted from [A9, A52, A71, A72]. (d) Schematic of a Ferroelectric Tunnel Junction (FTJ) memory device. Adapted from [A9, A19, A78].

Modes of Memristor Operation

Memristors can be operated in several distinct modes, depending on the intended application and system requirements. In analog operation, the continuous resistive tuning is exploited, and memristive devices act as non-volatile, continuously tunable resistors whose conductance depends on prior stimuli. This mode is commonly used to emulate synapses [A79] or for analog signal processing [A80]. However, in most digital systems and even in analog in-memory computing, memristive devices are typically used in a discrete fashion. The simplest case is binary (1-bit) operation, where the device switches between a low-resistance state (LRS) and a high-resistance state (HRS), representing logical '1' and '0' [A9]. More advanced is multi-level operation, in which the device is programmed to one of several predefined conductance states, allowing multiple bits of information to be stored in a single cell. Multi-level operation can be realized by programming the device into several distinct conductance states [A54]. In filamentary devices such as RRAM, this is often achieved by limiting the maximum current during switching [A81], whereas in PCM [A82] or ferroelectric devices [A83], the programmed state is defined by the amplitude and duration of the applied pulses. The practical number of achievable states depends on the conductance range of the device, the stability of the intermediate states, and the ability to reliably distinguish states in the presence of variability.

In discrete operation modes, cells are typically paired with, or incorporate, an access device (such as a transistor or selector) to control current flow, isolate individual cells, and, where required, enforce current compliance [A9]. In these modes, there is also a clear distinction between programming (write) and read operations [A52]. Read operations apply a small sensing bias chosen to avoid disturbing the stored state, while programming operations require larger electrical stimuli (either higher voltages, longer pulses, or higher currents) to

induce a state change. The exact voltage and current levels depend on the specific memristive technology. The resulting current is measured with a sense amplifier for binary reads or an analog-to-digital converter (ADC) for multi-level reads. For reliable multi-level operation, these programming pulses must be carefully controlled, often using a sequence of pulses interleaved with intermediate readouts to iteratively approach the target conductance state (for example, using the Incremental Step Pulse with Verify Algorithm, ISPVA, for RRAM devices [A81]). Limiting the maximum electrical stimulation during programming is important for stabilizing intermediate states and preventing excessive stress in devices such as RRAM [A84] or PCM [A82]. Compliance can be provided by the access device or driver circuitry. The chosen operation mode thus has significant implications for circuit design, application domain, and system-level modeling.

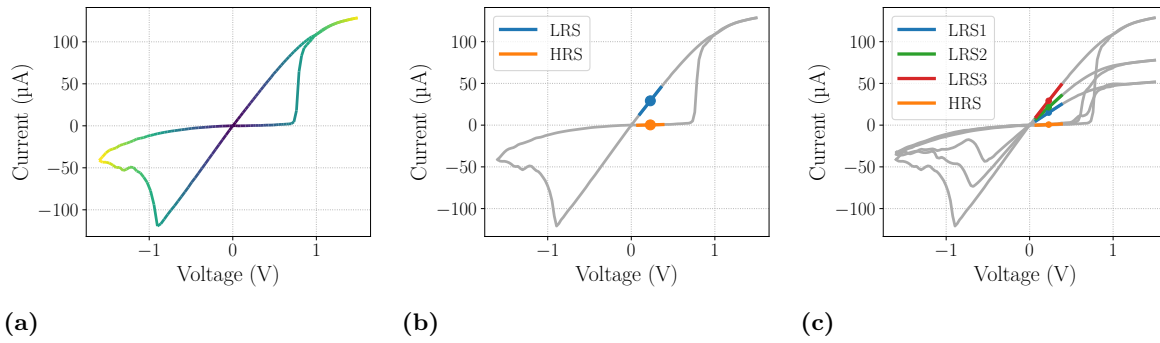


Figure 2.2.2: Illustration of the three principal modes of memristor operation using a RRAM device as an example: (a) analog mode, (b) binary (1-bit) mode, and (c) multi-level mode. Adapted from [A81].

Figure 2.2.2 illustrates the characteristic i - v behavior of a RRAM device operated in different modes. In analog mode (a), the device responds continuously to the applied voltage, so any resistance within its full range can be observed, as determined by the surrounding analog circuitry.

In binary operation (b), only two well-separated resistance states are used: the low-resistance state (LRS) and the high-resistance state (HRS). The highlighted and colored points correspond to the actual operating points of the device during normal use. These states are typically read out using low-voltage pulses (around 0.2 V), which are sufficient to measure the conductance without disturbing the state [A81]. The intermediate region of the curve is traversed only during programming, as the device switches from one state to another, but does not correspond to a stable operating point.

In multi-level operation (c), several predefined conductance states are targeted, allowing the storage of multiple bits per cell. The multiple i - v curves shown do not occur simultaneously. Each curve corresponds to the device response after programming to a specific conductance state using a different compliance current. To reliably achieve the distinct multi-level states shown here, the current through the device has to be actively limited. Again, only the highlighted points represent the true operating states, while the intermediate points illustrate the transitions that occur during programming.

It should be noted that the curves shown were obtained from quasi-static voltage sweeps (up to 1.5 V) on RRAM devices for characterization purposes. In practical operation, however, programming is typically carried out with short, high-voltage pulses that often exceed the static switching threshold observed in these quasi-static measurements [A81]. These fast pulses are required to achieve rapid and reliable state transitions. Note, however, that other memristive technologies exhibit distinct current-voltage characteristics. Consequently, while the i - v curves provide an illustrative overview of accessible states and operation modes, the actual switching dynamics during programming can differ significantly, influenced by the inherent time-voltage tradeoff and the application of sophisticated programming algorithms.

Application Domains

Memristors, owing to their intrinsic non-volatility and multi-level storage capabilities, have become promising components for a diverse array of applications. While initially explored as memory elements, their distinctive properties also facilitate novel computing paradigms beyond conventional digital storage. Consequently, modeling memristors at the system level necessitates consideration of the properties utilized in different use cases, particularly for those applications which closely interface or integrate with traditional digital computer systems. This section categorizes key application domains of memristors, elaborating on their underlying principles and highlighting representative implementations.

Discrete Storage Elements A primary application of memristors lies in discrete data storage, where a predefined set of resistance states encodes information digitally. During programming, the device is driven to the desired resistance state through a sequence of controlled electrical pulses defined by a programming algorithm, ensuring reliable encoding of the target bit values. Reading the stored data involves converting the analog conductance value back into its corresponding digital representation. Notably, many memristors can store multiple bits per cell by distinguishing several discrete resistance levels. However, this multi-bit capability often requires more sophisticated read-out circuitry to accurately resolve the finer gradations between states. The programming process of memristors is inherently more intricate than that of conventional memory technologies. Unlike binary-level switching in SRAM or DRAM, or the high-voltage charge-injection mechanisms used in flash memory, the multi-level operation of memristive devices that support it (such as RRAM, PCM and certain ferroelectric devices) typically relies on a digital-to-analog converter (DAC) and iterative programming algorithms to control changes in resistance precisely. Despite these challenges, memristor-based storage elements capitalize on their non-volatility and combine advantages from multiple established memory types: they enable local CMOS integration comparable to SRAM, achieve high data density and low-power operation similar to DRAM, and offer the non-volatility characteristic of NAND flash. Consequently, they can mitigate data-transfer bottlenecks within the memory hierarchy and between memory and processing units, positioning them as competitive alternatives to conventional memory technologies, albeit with trade-offs in control and programming complexity. [A9, A52] Figure 2.2.3 illustrates a typical memristor-based memory array, where specific devices (highlighted in red) are addressed and selected during read-out to retrieve their stored values.

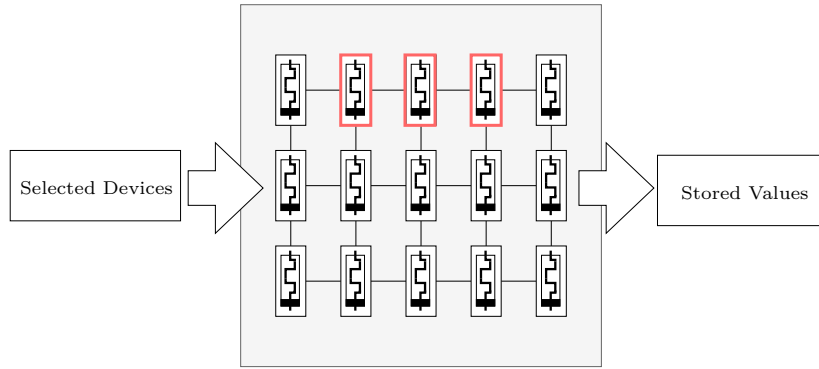


Figure 2.2.3: Schematic overview of the *discrete storage* application of memristors.

In-Memory Computing In the domain of in-memory computing, memristors are employed to perform analog computations: most notably matrix-vector multiplications, which are central operations in neural network accelerators. In such systems, each memristor stores a conductance value (discrete or analog) and behaves like an analog element during computation. Input voltages are applied across a crossbar array of memristors, and the resulting output currents, which reflect the weighted sum of these inputs, constitute the analog results of the multiply-accumulate (MAC) operation. These analog outputs are subsequently converted into digital form for further processing. This hybrid analog-digital approach harnesses the inherent speed and massive parallelism of analog computing while maintaining compatibility with conventional digital systems. Although traditional memory technologies such as SRAM, DRAM, and Flash have been explored for in-memory computing, they suffer from inherent drawbacks, including high power consumption (SRAM/DRAM), slow write speeds, and limited endurance (Flash). Memristive technologies, by contrast, offer a compelling alternative due to their non-volatility, scalability, and potential for low-energy, high-speed operation. Nevertheless, several challenges persist. Most notable are the issues of computational accuracy, integration complexity, inter-cell interference, leakage currents, and the design of efficient analog-to-digital conversion circuits. [A85, A86] Figure 2.2.4 illustrates a typical memristor-based crossbar array used for in-memory matrix-vector multiplication. During operation, the input vector is converted into voltage signals and applied to the array, where the memristor conductances represent the stored matrix weights. The resulting column currents correspond to the weighted sums (products of inputs and stored weights), which form the analog outputs of the computation.

Memristor-Based Logic Memristor-based logic circuits also build upon discrete storage mechanisms. In these systems, memristors represent binary information, and logic operations are executed by configuring networks of memristive devices whose resistive states directly encode logic values. The functionality of such circuits is inherently flexible, as reprogramming the memristor states allows the same hardware to realize different logic functions. Although the underlying physical processes may involve continuous (analog) changes in resistance, the resulting outputs are ultimately interpreted in digital form. This dual role of acting both as data storage and as active computational elements highlights the versatility of memristors in bridging memory and logic domains. These logic operations are frequently integrated directly within memory arrays, enabling a form of in-memory computation in which data

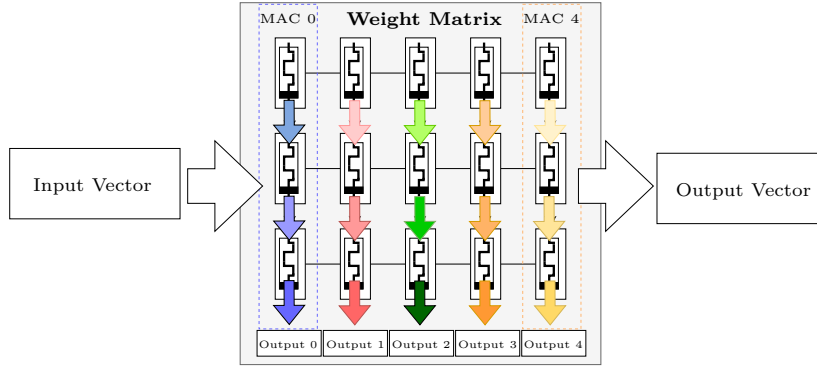


Figure 2.2.4: Schematic overview of the *in-memory matrix-vector multiplication* application of memristors.

processing occurs at the storage site itself. This co-location of logic and memory helps to reduce both latency and energy consumption by minimizing data movement. Alternatively, memristor-based logic can also be implemented as standalone circuits, separate from memory arrays. By interconnecting fundamental memristor-based logic gates, it becomes possible to construct complex digital building blocks such as adders, multipliers, and multiplexers, thereby supporting a broad range of computational tasks. [A11, A87, A88] Figure 2.2.5 illustrates a memristor-based array configured for in-memory logic operations. In this example, memristor states encode logical '0' or '1'. During operation, a voltage pulse is applied, which, depending on the input states, either switches the output memristor to a new resistance level or leaves it unchanged. By adjusting pulse amplitudes, switching thresholds, and cascading multiple devices, a wide range of stateful logic functions can be realized.

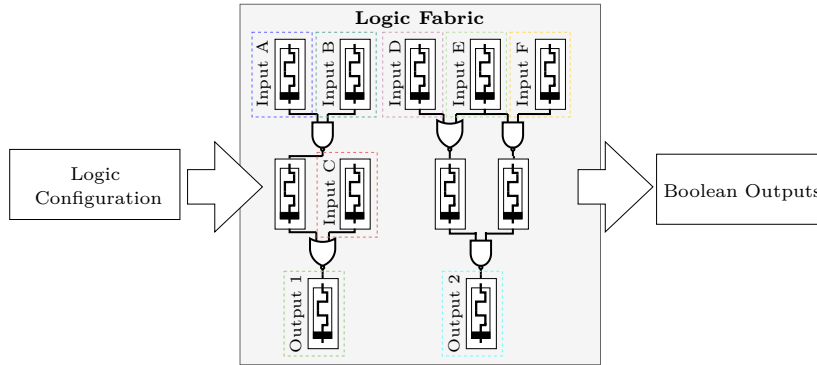


Figure 2.2.5: Schematic overview of the *logic in-memory* application of memristors.

Physical Unclonable Functions Physical unclonable functions (PUFs) exploit the intrinsic variability and uniqueness of memristor characteristics to generate unpredictable yet reproducible responses. In PUF implementations, the time-independent variability of resistance states (utilized either in discrete or analog form) serves as a robust source of entropy for secure key generation. As in other applications, the analog responses of memristors are digitized to ensure compatibility with conventional digital security protocols. [A89–A91] Beyond deterministic key generation, the inherent stochastic behavior and time-dependent

variability of memristors can also be harnessed for true random number generation, further extending their relevance in hardware security contexts [A92, A93]. Together, these mechanisms provide powerful means of enhancing the security and trustworthiness of electronic systems. Figure 2.2.6 illustrates a typical memristor-based array used as a PUF. In response to a given challenge, specific devices are addressed and selected (highlighted in red). Owing to their intrinsic stochastic characteristics, each device exhibits a distinct response profile, such that every particular combination of selected devices and applied challenge produces a unique yet reproducible output pattern.

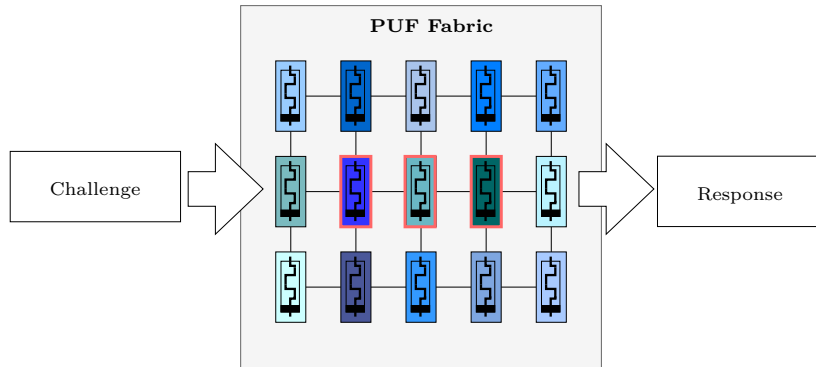


Figure 2.2.6: Schematic overview of the *PUF in-memory* application of memristors.

Analog Circuit Applications In certain applications, the continuous analog nature of memristors is exploited directly. Their nonlinear behavior, evident from the pinched hysteresis in the i - v response, together with a continuous range of resistance values, forms the basis for analog signal processing directly in the analog domain, without intermediate digital conversion. One prominent application is in the emulation of biological neurons, where spiking neural networks leverage memristors to mimic synaptic behavior. In these networks, the accumulation of input voltage pulses gradually modifies the memristor conductance, effectively replicating the adaptive potentiation and depression dynamics of biological synapses. This functionality is often implemented within memory arrays and thus represents a form of analog in-memory computing [A94–A96]. Beyond neural emulation, memristors have been utilized in a range of other analog applications. These include the development of memristor-based analog oscillator circuits [A97], digital-to-analog converters [A98], and phase-frequency detectors [A99]. Together, these applications underscore the versatility of memristors in analog circuit design, offering efficient alternatives to traditional analog components through their unique nonlinear behavior and continuous tunability.

2.3 Integration of Memristors into Computer Systems

In the application domains discussed in the previous section, memristors excel by leveraging their unique properties. However, the functionality provided by memristor-based components, such as memory modules or accelerator cores, represents only one part of a complete computer architecture. Their effective use therefore requires integration into broader system designs, which introduces distinct challenges beyond the device and circuit levels. Addressing these

challenges first requires an understanding of modern system design methodologies and how they can be adapted to accommodate emerging device technologies.

Modern computer systems are highly complex, integrating diverse functionalities that cannot be managed efficiently through traditional gate- or register-transfer level (RTL) abstractions [A100, A101]. To cope with this complexity, the *Electronic System-Level (ESL)* design paradigm has emerged, emphasizing overall functionality, performance, and communication structures over low-level circuit details [A102, A103]. In the following ESL is referred to simply as *system-level design*.

By raising the abstraction of entry above RTL and gate-level descriptions, system-level design enables higher productivity, shorter design times, and improved reliability in complex integrated systems. This top-down methodology begins with functional requirements and performance targets, which are progressively refined into implementable architectures. At each refinement step, functionality, timing, and resource allocation are modeled with increasing accuracy, while verification ensures consistency across abstraction levels. Ultimately, the system description is refined into circuit-level implementations and, finally, mapped to device-level characteristics [A100].

System-level design offers several key benefits that are directly relevant for memristor integration. Functional partitioning clarifies the distribution of tasks between hardware and software, ensuring efficient utilization of specialized memristor-based components. Abstraction of the communication infrastructure highlights data flow and interfaces, thereby facilitating the integration of heterogeneous components into scalable architectures. High-level performance and power models enable early exploration of system behavior, allowing bottlenecks to be identified before costly design stages. Finally, parallel development and early verification reduce integration risks, an aspect particularly critical when introducing novel device technologies [A103].

It should be noted that system-level design has historically been developed with a strong focus on *digital* systems, reflecting the dominance of digital hardware in System-on-Chip (SoC) and embedded design flows. The integration of analog and mixed-signal components, however, has gained increasing attention. Recent behavioral analog modeling frameworks demonstrate that the methodology can be generalized to heterogeneous systems [A104]. This aspect is particularly relevant for memristive technologies, which inherently exhibit continuous, analog switching behavior and thus challenge a purely digital view of system abstraction.

Abstraction Levels in Computer Design

A foundational framework for describing hardware design abstraction is provided by the Gajski-Kuhn chart or Y-diagram [A105]. The Y-diagram conceptualizes system design along three interrelated domains: *behavioral*, *structural*, and *physical*. Each domain can be represented at multiple abstraction levels, ranging from the overall system level down to the device and material level. Movement along one axis corresponds to refinement within a domain (e.g., behavioral refinement from high-level functionality to detailed timing), while movement across axes represents transformations between domains, such as mapping behavior onto structure or structure onto physical realization [A106]. This framework illustrates

how complex systems evolve from abstract functional descriptions toward concrete physical implementations.

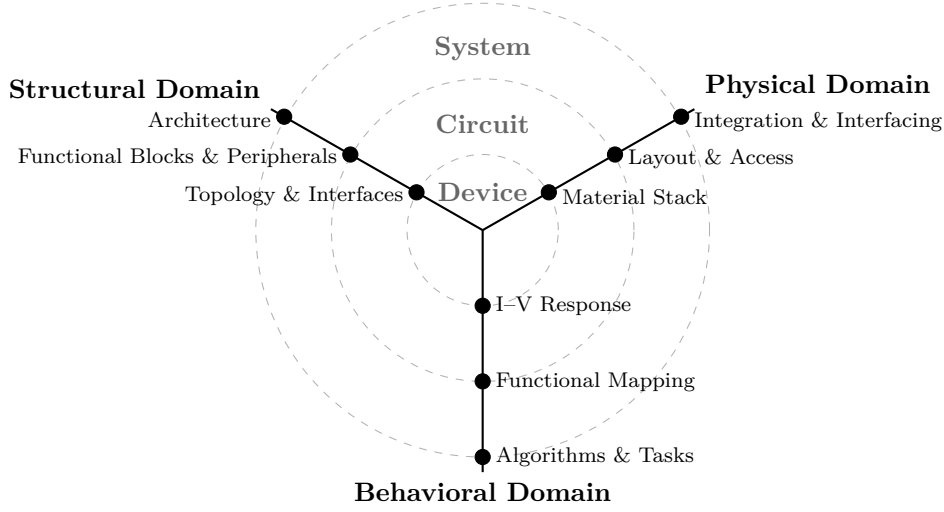


Figure 2.3.1: Conceptual representation of a Y-diagram adapted for memristor-based systems. Adapted from [A106].

Within this general design framework, the hardware design process is commonly organized into a hierarchy of abstraction levels, each emphasizing different design concerns [A107]:

- **System-level:** Captures functionality, concurrency, communication, and partitioning between hardware and software.
- **Register-transfer level (RTL):** Describes functionality in terms of registers and clocked data transfers, typically using hardware description languages (HDLs) such as VHDL or Verilog.
- **Gate-level:** Specifies circuits as networks of Boolean logic gates and flip-flops.
- **Circuit-level:** Describes the electrical behavior of transistors, interconnects, and analog/digital building blocks.
- **Device-level:** Examines the underlying physics and material properties of devices, including fabrication processes and physical limitations.

For the purpose of analyzing memristor-based systems, these levels are consolidated into three categories in this thesis: the *system level*, *circuit level*, and *device level*. This simplification reflects the unique position of memristive technologies within the Y-diagram framework. Unlike conventional digital elements, memristors exhibit strong coupling between the *behavioral* and *physical* domains and their analog, state-dependent switching dynamics cannot be adequately represented at RTL or gate-level abstractions. Consequently, the main design challenge lies in bridging device- and circuit-level behavior to system-level functionality, motivating modeling approaches that incorporate device characteristics within higher-level simulations and co-design methodologies. Figure 2.3.1 illustrates a Y-diagram adapted by the author of this thesis for memristor-based systems, which serves as a conceptual reference for the abstraction hierarchy used throughout this thesis.

Tools and Virtual Prototyping

Tools such as SystemC, transaction-level modeling (TLM), and virtual prototyping platforms are central to system-level design. They provide flexible environments for high-level modeling, simulation, refinement, and validation [A102, A103, A108]. In parallel, computer-aided design (CAD) and computer-aided engineering (CAE) tools for synthesis, verification, and layout ensure continuity with traditional digital design flows. Together, these toolchains accelerate the design cycle and enhance overall design quality by enabling iterative optimization early in the process [A107].

Challenges in Integrating Memristors

While memristors enable promising new computing paradigms, their integration into modern computer systems introduces several challenges. On the device and circuit levels, fabrication and circuit design issues must be addressed [A28, A48, A59, A70, A109]. At the system level, memristor behavior must be modeled in an abstract yet faithful manner to evaluate reliability and performance within larger architectures. A major difficulty arises from the continuous, history-dependent behavior of memristive devices, compounded by variability in their resistance states.

Memristors fundamentally differ from traditional digital components, as their operation is governed by continuous, analog resistance changes rather than discrete switching events. This intrinsic behavior complicates their integration into established digital design flows, which assume deterministic binary switching, standard-cell-based synthesis, and mature CAD tool support. For memristors, these assumptions no longer hold: their analog nature blurs the boundary between digital and analog design, suitable device libraries and primitives remain largely unavailable, and verification must account for stochastic switching and device-level variability. Consequently, extensions to existing design flows, supported by advanced virtual prototyping platforms, are required to enable realistic and efficient design exploration [A25].

The main characteristics of memristors that complicate their integration are:

- *Analog Behavior*: Instead of binary switching, memristors exhibit gradual resistance variations, complicating the representation of their interfaces to conventional digital components [A110].
- *Variability*: Due to material imperfections, process variations, and stochastic switching, memristors often show significant variability and random switching behavior [A28]. Phenomena such as drift, aging, and retention loss further complicate modeling, as these effects evolve dynamically over time and depend on both conductance state and programming history [A27].
- *Unresolved Mechanisms*: The fundamental physical processes underlying resistance switching of many types of memristors remain only partially understood, complicating efforts to provide clear descriptions and accurate models [A26, A28, A29, A111].

Variability in Memristors

Understanding variability is crucial for assessing the reliability and predictability of any memory technology [A112]. However, in the case of memristive devices, variability plays a fundamentally different role than in conventional memories [A61]. Many existing works on memory modeling and system-level reliability rely on assumptions and mitigation strategies derived from well-established technologies such as SRAM, DRAM, and Flash [A113–A117]. These approaches map device-level properties to system-level behavior under the premise that the sources of variation are well understood and statistically stable. In contrast, memristors represent a paradigm shift in both device physics and operational mechanisms. The origins of variability in memristors (such as filamentary dynamics, stochastic ion migration, and material inhomogeneity) differ fundamentally from the variations in traditional memories. As a result, classical system-level approaches cannot be directly applied, and new frameworks are required to capture and manage memristor-specific variability.

To highlight this distinction, this section first reviews the typical sources and characteristics of variability in conventional memory technologies, before contrasting them with the unique variability mechanisms observed in memristors.

SRAM Variability In SRAM, variability primarily originates from microscopic differences in the transistors that make up each cell. One major factor is the random placement of dopant atoms in the channel region of the transistors. Since only a few atoms are present at nanometer scales, even slight differences in their number or location can shift the threshold voltage at which a transistor switches on. If the two transistors forming the storage latch of an SRAM cell have mismatched thresholds, the cell becomes less stable and more prone to errors when reading or writing. Another important contributor is the random trapping and release of charge carriers at defects in the transistor’s gate oxide. This process, often called random telegraph noise, causes the effective threshold voltage of a transistor to fluctuate over time. Such temporal variability further reduces the stability margin of the SRAM cell. The combined effect of these mechanisms is a reduction in the static noise margin, a measure of how robustly the cell maintains its state against disturbances. Lower margins translate into a higher probability of read and write failures, especially as devices are scaled down in size [A118–A121].

DRAM Variability In DRAM, the dominant source of variability is the retention time of the individual memory cells. Each cell stores information as charge on a tiny capacitor, which gradually leaks away. Due to small manufacturing differences, some capacitors lose their charge much faster than others. To avoid data loss, the refresh rate of the entire chip must be set according to the weakest cells, even though most cells could retain their charge for much longer. This mismatch not only reduces energy efficiency but also becomes more critical as device dimensions shrink. Some cells even exhibit unpredictable retention behavior, where their charge loss rate can suddenly change from one refresh cycle to the next. This phenomenon, known as variable retention time, makes it difficult to guarantee reliability through static refresh policies. To counteract such issues, modern DRAM devices rely on error correction codes and adaptive refresh strategies [A122]. A second source of variability arises in the sense amplifiers, which are responsible for detecting the minute voltage difference that distinguishes a stored ‘0’ from a ‘1’. If the transistors inside the amplifier have slightly

different characteristics, they may misinterpret the stored value or require longer to resolve it [A123].

Flash Variability In Flash memory, variability is mainly tied to how reliably each cell can be programmed and how long it can retain its charge. A Flash cell stores information by trapping electrons in a thin insulating layer. Over time and repeated use, tiny imperfections accumulate in this layer. As a result, different cells tolerate a different number of program/erase cycles before they fail. This endurance variation means that some cells wear out much earlier than others, complicating the task of distributing write operations evenly across the memory. Modern 3D NAND Flash introduces additional process-induced variation because memory cells are stacked in multiple layers. Subtle differences between layers can cause some rows of cells to degrade faster or store charge less reliably than others. Finally, charge leakage over time leads to variable retention behavior, causing some cells to lose their stored information much sooner than expected. Together, these mechanisms make variability in Flash a key challenge for both endurance and data retention [A124, A125]. Another significant source of variability arises from the distribution of programmed charge levels. When cells are programmed, the target threshold voltage is not always reached exactly, leading to a spread in programmed states. As the device ages, this distribution broadens, and the states of neighboring levels can begin to overlap, increasing the risk of read errors [A126].

Memristor Variability Compared to conventional memory technologies, memristive devices such as RRAM, PCM, STT-MRAM, and FTJ exhibit substantially higher and more complex variability. The key reason is that information in these devices is stored and processed directly through their electrical conductance states. Any fluctuation in the underlying switching processes therefore translates directly into uncertainty in the stored or computed value. This is in sharp contrast to SRAM, DRAM, or Flash, where circuit-level design margins and decades of process refinement keep variability within manageable bounds and thus largely invisible at the system level. For memristors, variability affects not only reliability in digital storage but also directly limits the achievable precision in analog or multi-level applications, where conductance must be programmed and read with high accuracy [A61, A127].

Memristor variability arises from multiple sources and is, in this work, broadly categorized as follows:

- **Time-Invariant:** This class encompasses sources of variability that are fixed after device fabrication and remain constant during operation. The most prominent example is *device-to-device variability*, which arises from fabrication-induced differences between nominally identical devices. Other forms include *batch-to-batch* and *within-array* variability, which are collectively referred to as spatial variability. These variations are typically modeled as static distributions of device parameters that define the baseline spread in device characteristics across a population.
- **Time-Dependent:** This class comprises sources of variability that evolve during device operation. Key examples include *cycle-to-cycle variability*, referring to stochastic differences in device behavior between consecutive programming events, and *temporal drift* or *retention loss*, in which device properties change over time even without

external stimuli. The underlying mechanisms are often inherently stochastic, such as filamentary switching processes, and are critical considerations when modeling device reliability and endurance.

- **Environmentally induced variability** acts as an orthogonal modifier to both categories. Factors such as temperature, applied voltage, and ambient conditions can either exacerbate or mitigate both time-invariant and time-dependent variability.

The consequences of this pronounced variability are manifold, including reduced reliability in digital storage, limited precision in multi-level memory and analog computing, and challenges in fabricating uniform material stacks. Mitigation strategies are therefore essential for making memristor-based computing viable. Figure 2.3.2 qualitatively illustrates the spread in the current-voltage characteristics of a RRAM device, highlighting the impact of time-dependent variability observed across multiple switching cycles.

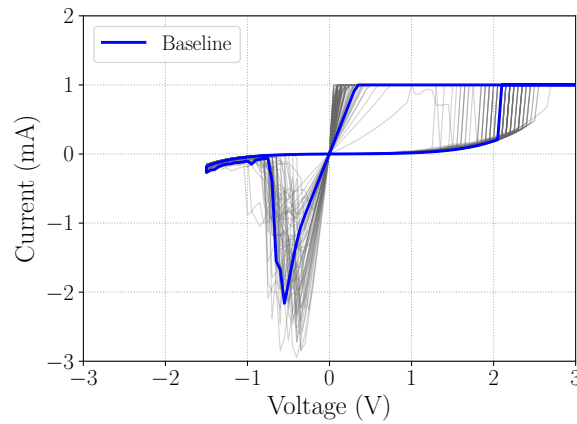


Figure 2.3.2: Qualitative example of cycle-to-cycle variability in memristor current-voltage curves. Adapted from [A128].

2.4 Memristor Modeling Approaches

Accurate modeling of memristor behavior at the system level requires a careful balance between physical realism and computational efficiency. Despite significant progress in understanding memristive devices, existing modeling efforts have predominantly followed a bottom-up approach, starting from physical principles and gradually abstracting toward higher levels [A129]. While this strategy has provided valuable insight into switching mechanisms and material properties, it has also revealed important limitations: the underlying physical processes are still not fully understood (see Section 2.3), and therefore even physics-based models require calibration against experimental data. Moreover, models developed from device-level considerations often struggle to scale efficiently to system-level simulations, where thousands or millions of devices interact dynamically. As a result, there remains a clear gap between highly detailed but slow device-level models and fast but inaccurate empirical models. Bridging this gap requires a modeling approach that directly incorporates device measurements, achieving high accuracy without excessive computational complexity - an aspect this section will contextualize in relation to existing work.

Figure 2.4.1 presents a qualitative comparison of execution speed, accuracy, and level of detail for several notable models from the literature. These models can typically be described as analytical, compact, or empirical. However, the boundaries between these categories are fluid rather than strict, and many models occupy positions in between, combining features of multiple types. Their realizations may use either continuous or discrete descriptions (compare Figure 1.4.1). This comparison was primarily compiled from analyses of the models and their respective publications, supplemented by relevant evaluations reported in previous studies [A130–A134]. For reference, the figure also includes the development of physical prototypes, categorized as *Exact* but slow, since fabricating actual devices can take several months. Finally, an estimated optimal model for system-level evaluations is indicated and labeled *Goal*. It represents a very fast model that maintains high accuracy and a sufficient level of detail for system-level studies.

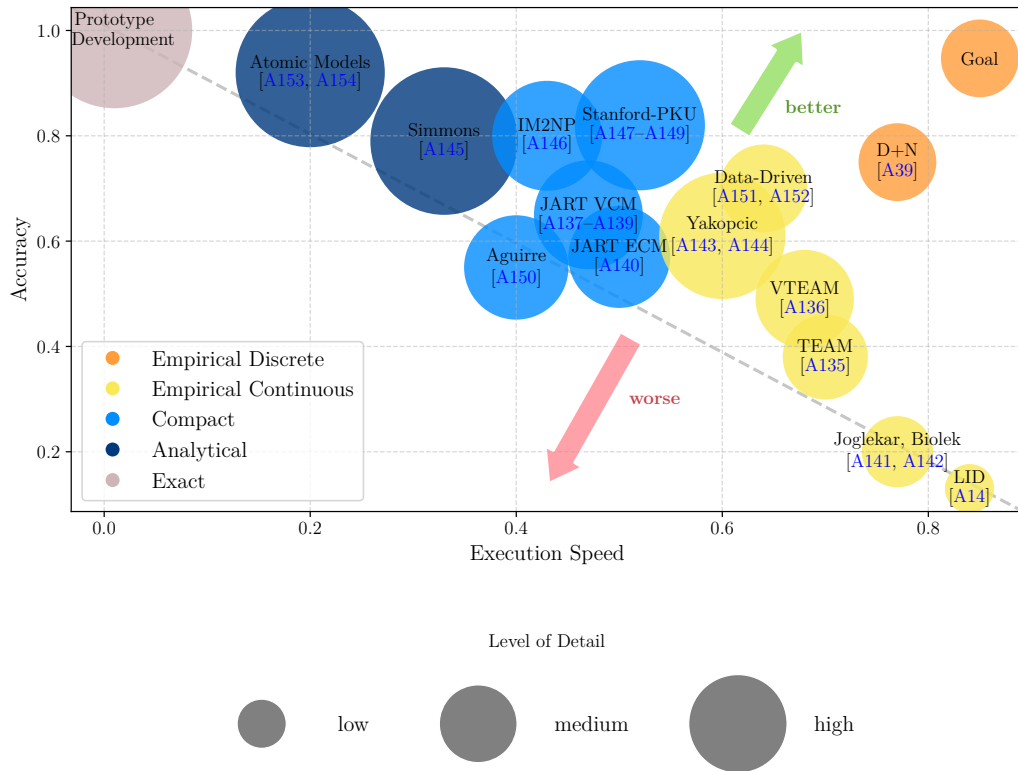


Figure 2.4.1: Qualitative comparison of different memristor models from literature.

Analytical Models provide the most accurate and detailed description by modeling the underlying physics, often down to the atomic level or the specific physical processes within the device. When calibrated against real measurements, these models can accurately predict low-level device characteristics and offer in-depth insights into phenomena such as ionic transport, filament formation, and defect states [A153, A154]. They are particularly valuable in the development and evaluation of new memristor devices. However, due to their computational intensity and focus on a single device type, these models are less practical for large-scale simulations and system-level integration.

Compact Models strike a balance between computational speed, level of detail and accuracy. By encapsulating key physical behavior in an efficient form, these models integrate seamlessly into larger electrical simulation frameworks. Typically used at the circuit level, they allow circuit designers to simulate moderate-sized memristive components by approximating the current-voltage characteristics of a memristor. The most notable physical compact model, used in many circuit and even small system design works, is the Stanford-PKU model [A147–A149]. It is widely adopted because of its accurate, experimentally validated, and, for a physics-based model, relatively computationally efficient description that captures the analog switching behavior with low parameter complexity. However, despite the balanced approach compact physical models can still be too computationally intensive for system-level simulations involving thousands of memristive devices. Furthermore, as they are often tailored to specific devices and materials, and given the incomplete understanding of certain memristor behavior, these models require calibration against measurement data to achieve acceptable accuracy.

Empirical Models capture memristor behavior from a functional perspective, prioritizing simulation speed and ease of use. Important examples are the TEAM [A135] and VTEAM [A136] models, the Biolek and Joglekar non-linear ion-drift model [A141, A142] and the HP linear ion drift (LID) model [A14]. While these models are well-suited for high-level simulations, they often sacrifice physical realism and accuracy when representing the true behavior of the device. For example, the linear ion drift model (one of the first models introduced by researchers at HP Labs following the initial memristor fabrication) is exceptionally simple and fast to simulate. However, it only provides a rough approximation of the hysteresis behavior and overlooks several effects present in real devices. Moreover, empirical models frequently neglect the impact of both time-dependent and time-invariant variability, which must then be accounted for separately to achieve realistic device behavior.

Empirical Discrete Models represent memristors using a predefined set of discrete resistance or conductance states, typically determined by programming algorithms. Only these discrete states are treated as relevant operating points, disregarding the analog transitions between them. This approach significantly increases simulation speed and simplifies system-level integration. Despite their promise for system-level simulations, such models have not been researched very thoroughly. The most prominent usage of this modeling approach is found in NeuroSim and its variants (such as DNN+NeuroSim)[A39], an application-specific tool for simulating the inference of neural networks on memory crossbar arrays. In this tool, a discrete memristor model (D+N) is integrated in a high-level simulation using predefined resistance states to represent the data of the neural network. Additionally, a static distribution can be applied on top to account for variability. This is a solid basis. However, since the variability effects are entirely theoretical, they do not adequately reflect real-world memristor behavior. It exemplifies how such approaches often sacrifice accuracy for speed. A more advanced discrete modeling strategy, by contrast, could leverage variability data from real measurements, preserving simulation speed while substantially improving accuracy at the most relevant operating points.

In summary, current memristor modeling approaches reveal a persistent trade-off between detail, speed, and accuracy. Analytical and compact models capture rich physical behavior but

are **too slow** for large-scale or architectural simulations. Empirical and discrete models, while efficient, lose essential device characteristics and rely on **oversimplified** or synthetic representations of variability. Compounding this issue, even the most advanced physical models must ultimately be fitted to measurement data to account for partially unknown operational influences, creating a **disconnect between device-level reality and the abstractions** used in system design. This motivates the development of a new, measurement-driven modeling approach that reverses the conventional direction: rather than fitting models to match data, it derives the model directly from real device measurements. Such a strategy preserves the statistical and physical characteristics observed experimentally while maintaining computational tractability. Consequently, it can provide a much stronger foundation for accurate and scalable system-level evaluation and bridge the gap between detailed device behavior and abstract system-level representations. Such a model will be introduced in Chapter 3.

2.5 System-Level Frameworks for Memristors

Various simulation frameworks for memristor-based system design have been developed. Table 2.5.1 summarizes the most notable frameworks today. These tools are predominantly targeted at memory arrays and, even more commonly, at in-memory matrix-multiplication accelerators for neural network inference or training. The NeuroSim framework actually exists in multiple variants for both inference and training of neural networks. However, since these share the same core principles, it is represented here as a single entry.

Table 2.5.1: Comparison of selected system-level memristor frameworks

Name	Scope	Memristor Model	Variability	Speed	Integrability
NVSim [A40]	Mem. array	⁻¹	None	++	Standalone
NVMain [A155]	Mem. array	⁻¹	None	++	Standalone
MemTorch [A38]	NN accel.	Multiple ²	Idealized, UD ³	⁻⁴	Standalone
NeuroSim [A156]	NN accel.	Discrete Empirical	Idealized, UD ³	+++	Standalone
MNSIM [A157]	NN accel.	Discrete Empirical	Idealized, UD ³	++	Standalone
RxNN [A158]	NN accel.	Discrete Empirical	Idealized	+++	Standalone
SySCIM [A37]	NN system	Yakopcic (SC-AMS) ⁵	Idealized, UD ³	++	SystemC
ACCLAIM [A159]	NN system	⁻⁶ (SC-AMS) ⁵	Idealized	++ ⁷	SystemC
MHSim [A160]	NN system	Discrete Empirical	Idealized, UD ³	++	ZSim

¹ Only extra-functional properties modeled (energy, latency, area).

² Model types: LID, VTEAM, Stanford-PKU, Data-driven, custom (via generic interface).

³ UD = user-defined: normal distribution with target conductance as mean and user-specified σ .

⁴ Speed depends on selected model.

⁵ Continuous model implemented in SystemC-AMS.

⁶ Memristor model not disclosed.

⁷ Speed assumed based on similarity to SySCIM.

Across all frameworks, two principal memristor modeling strategies are observed. Some tools adapt continuous device models, originally developed for device- or circuit-level studies, for use at higher abstraction levels (as in MemTorch or SySCIM). Others employ discrete empirical, lookup-table-based models, where only the target conductance state is stored, resulting in much faster simulation (as in NeuroSim, MNSIM, RxNN, or MHSim).

MemTorch distinguishes itself through a flexible and extensible model interface that enables both the integration of existing device models and the implementation of custom ones. Nevertheless, it provides little guidance on model selection or parameter fitting, which can undermine the realism and comparability of simulation outcomes. Apart from NVSim and NVMain, which are limited to array-level memory metrics, most tools nominally account for device variability. In practice, however, this is typically restricted to simple, idealized randomization: conductance states are varied according to a normal distribution centered on their mean values, with a user-defined relative standard deviation σ (marked in the table as *idealized* and *user-defined (UD)*, respectively). Such variability models are rarely grounded in physical mechanisms or operational conditions, and users are given no guidance in selecting appropriate standard deviations. This underscores both the lack of standardized approaches to modeling variability and the broader difficulty of capturing realistic device behavior in system-level simulations.

In terms of speed, frameworks that use discrete empirical models achieve high simulation throughput, making them suitable for large-scale workloads, while those employing continuous models (e.g., implemented in SystemC-AMS, like in SySCIM and ACCLAIM) incur greater computational overhead. Most frameworks (NVSim, NVMain, MemTorch, NeuroSim, MNSIM, RxNN) operate as standalone tools, producing results that must be manually transferred into broader system-level design environments. Only SySCIM and ACCLAIM offer native integration with digital simulation through SystemC, though this comes at the cost of slower, more complex mixed-signal co-simulation, since their memristor models are implemented in SystemC-AMS.

MHSim, which couples NeuroSim with the CPU simulator ZSim [A161], is notable for supporting system-level co-simulation of memristor accelerators and CPUs. However, the tight coupling to ZSim makes the integration of custom system components, such as DSPs or digital accelerators, difficult.

In summary, despite recent progress, there remains a lack of standardized, physically validated memristor and variability models for system-level simulation, and most frameworks offer limited compatibility with general-purpose full-system design flows. This highlights the ongoing need for unified, extensible, and realistic frameworks to enable rigorous evaluation of memristor-based computing architectures.

Chapter 3

A Memristor Model targeted at the System Level

Accurately modeling memristor behavior at the system level requires striking a careful balance between physical realism and computational efficiency. As discussed in Section 2.4, due to the complex switching mechanisms, material dependencies, and external influences affecting memristive devices, fully physics-based models are often too detailed and computationally expensive for large-scale system simulations, and they still need to be fitted to device measurements for high-accuracy results. Furthermore, the absence of standardized programming protocols and the inherent trade-offs among access time, stability, energy consumption, and endurance further complicate direct abstraction from device-level behavior. These challenges are compounded by significant variability and the influence of operating conditions such as temperature and applied voltage schemes. Consequently, a system-level model must selectively capture the most critical factors influencing memristor performance while abstracting away low-level physical details.

To address these challenges, the Discrete Statistical Model (DStM) was conceived, refined, and extended throughout this work at hand. Its key aim is to bridge the gap from device-level characterization to system-level simulation by providing a model that is both accurate and computationally efficient, with just the right level of abstraction for integration into virtual prototyping environments such as SystemC-based toolflows. The following chapter introduces the conceptual foundations and formal description of the DStM.

3.1 Factors Influencing Device Behavior

The behavior of memristive devices is governed by a complex interplay of physical, technological, and operational factors. Understanding these influences is essential for the development of accurate and reliable system-level models. Based on an extensive literature study conducted by the author of the thesis at hand, the most important factors affecting device behavior were identified, analyzed, and systematically categorized. Figure 3.1.1 illustrates the resulting classification developed by the author, which summarizes the key domains influencing memristive behavior. These include device physics and operating principles, materials

and composition, device structure and scaling, fabrication techniques, peripheral circuitry and system integration, as well as external operating conditions. In the following, each of these categories is discussed in more detail, with particular emphasis on those factors that are most relevant for abstraction at the system level.

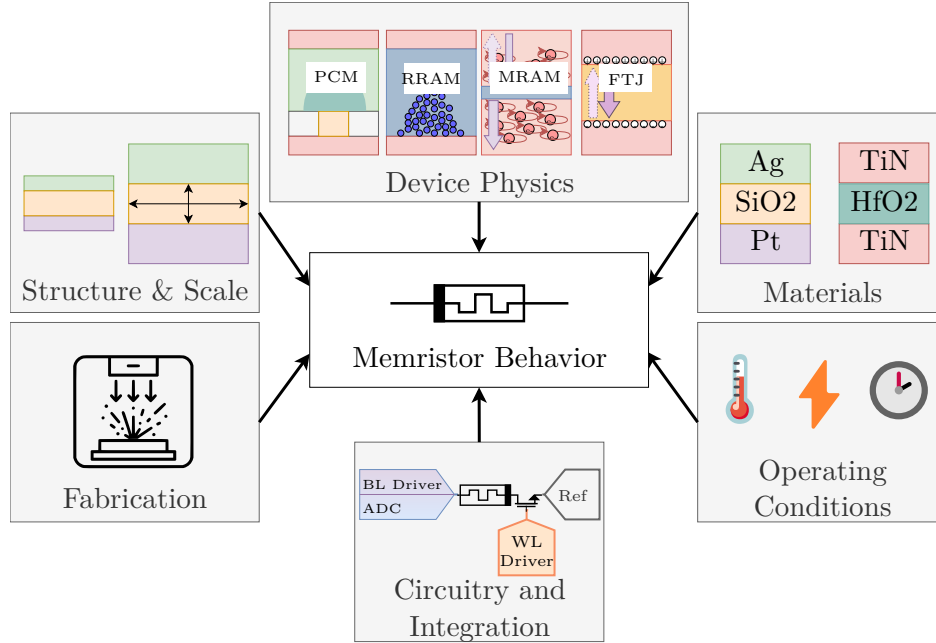


Figure 3.1.1: Overview of key factors that determine memristive device behavior.

Device Physics and Operating Principles The switching behavior of memristive devices is governed by physical mechanisms such as atomic rearrangement, ion migration, spin dynamics, and polarization switching (compare Section 2.2). These processes not only define core properties like switching speed, endurance, retention, and programming thresholds, but are also inherently stochastic. This leads to device-to-device and cycle-to-cycle variability [A61]. Factors such as switching polarity, voltage amplitude, and pulse duration needed to control the behavior of the devices influence the typical device response and its variability [A63, A162]. For system-level modeling, these principles must be abstracted to capture the dependence of device state and reliability on input conditions, while reflecting the intrinsic variability imposed by the underlying physics.

Materials and Composition While the underlying switching principle, such as modulation of oxygen vacancies in RRAM, defines the general operation of a memory device, the specific choice of materials, their composition, and their arrangement within the device stack have a profound impact on actual device behavior and variability. For a given switching mechanism, a wide range of materials can be employed (e.g., TiO₂, HfO₂, and ZnO in RRAM), each introducing distinct electrical characteristics, switching thresholds, and variability profiles. Additionally, variations in thickness, stoichiometry, doping, and layer sequencing can further modulate device performance and the degree of stochasticity observed. For example, HfO₂-based RRAM typically shows narrower switching distributions than TiO₂ devices due

to more stable filament formation. Electrode selection and interface engineering (e.g., buffer layers, interfacial defect control) play equally crucial roles, affecting not only endurance and retention but also the consistency of switching events. As a result, even devices based on the same physical principle can exhibit widely differing operational and reliability characteristics due to their material choices. For system-level modeling, it is essential to abstract these material-dependent effects to account for their influence on programming behavior, endurance, and variability. [A29, A163, A164]

Device Structure and Scaling The geometrical design and physical dimensions of memristive devices critically shape their electrical behavior, scalability, and reliability. Parameters such as active layer thickness, electrode area, and aspect ratio directly influence switching voltage, programming energy, and speed. For instance, in RRAM, the oxide thickness and device cross-section strongly affect filament dynamics, contributing to variability and endurance limits. In PCM, the size of the phase-change region determines programming current and the stability of resistance states. As devices are scaled to nanometer dimensions, variability is further amplified due to enhanced stochastic effects and, in some cases, the emergence of quantum phenomena. Ultra-small structures also complicate manufacturability, often impacting yield and long-term stability. For system-level modeling, it is essential to abstract these structural dependencies, as they directly affect switching variability, endurance, and the access characteristics of large-scale memory arrays. [A165, A166]

Fabrication Techniques How memristive devices are fabricated has a major impact on their performance, reliability, and variability. A key step in fabricating memristive devices is deposition, the process of forming thin material layers onto a substrate to build up the device stack. Common deposition techniques include sputtering (physically 'blasting' material onto a surface), atomic layer deposition (which builds films one atomic layer at a time for high uniformity), and chemical vapor deposition (where chemical reactions form layers from gases). The choice of technique determines how smooth, pure, and consistent the device layers are, which directly impacts device performance and reliability. For example, in RRAM, the way the oxide layer is deposited sets how evenly oxygen vacancies are distributed, which in turn affects how reliably the device can switch states. In phase-change memory, the thickness and composition of the switching material must be tightly controlled to ensure the device can reliably change between phases. After deposition, thermal treatments like annealing (heating the material under controlled conditions) are often used to improve the arrangement of atoms, reduce defects, and make interfaces more stable. Differences in how devices are fabricated, between manufacturers or even across batches, can result in significant differences in switching voltages, endurance, and long-term data retention. In some cases, fabrication defects can result in device faults, producing non-functional cells that are unable to switch or retain information as intended. For system-level modeling, it is important to account for both the variability and the possibility of non-functional cells introduced by fabrication processes, as these factors affect the consistency and reliability of large memory arrays. [A167, A168]

Operating Conditions Beyond the materials and structure, memristive device behavior is strongly influenced by external conditions. Temperature affects how easily ions or atoms move within the device, shifting switching thresholds and data retention, but high temperatures can also speed up wear and long-term degradation [A169, A170]. Humidity and air

exposure can lead to chemical changes, material degradation, or unstable interfaces, especially in oxide-based devices or with reactive electrodes, raising variability and risk of failure over time [A171]. Radiation, such as encountered in space, may introduce defects or change material composition, sometimes causing permanent device malfunction [A172]. Mechanical stress, from packaging or operational vibration, can alter defect migration paths and affect both performance and device lifetime [A173]. Each of these factors introduces additional variability, that must be considered for applications in harsh or long-lifetime environments. For system-level modeling, they represent important sources of operational uncertainty and possible accelerated aging, depending on how and where the devices are used.

Peripheral Circuitry and System Integration Memristive devices are typically embedded within circuits and/or large arrays, often paired with access transistors (as in 1T1R configurations) to enable individual cell selection and mitigate unwanted current paths. However, the properties of access transistors themselves can introduce additional variability and influence device operation, with effects that may be sensitive to environmental conditions like temperature. Programming and readout circuitry further impact device behavior. Digital-to-analog converters (DACs) used for programming can limit the precision of conductance tuning, while analog-to-digital converters (ADCs) used for reading can introduce quantization error, noise, and offset errors, all contributing to variability in observed device states. Noise in the programming pulses or in the peripheral circuits can result in imprecise switching, degrading both retention and endurance. As array sizes increase, IR drop (voltage drop across resistive interconnects) becomes significant, causing the actual voltage at each device to differ from the intended value. This leads to non-uniform programming, read errors, and increased variability across the array. Additionally, leakage currents and unintended conduction paths ('sneak paths') can distort device operation, further complicating reliable programming and sensing, especially in passive crossbar architectures. The specific programming algorithm, such as pulse sequence, duration, and verify-write schemes, can also affect device variability and retention characteristics. For system-level modeling, it is important to account for these influences as they can amplify or mask intrinsic device variability, ultimately affecting the performance and reliability of the full system. [A28, A162, A174, A175]

3.2 The DStM Model Concept

To put the DStM presented in this work into perspective, it is first important to understand the general mathematical description of a memristor. According to [A17] a general voltage controlled memristor can be described by two equations:

$$i(t) = G(x, t) \cdot v(t) \quad (3.1)$$

Equation 3.1 describes the current-voltage relationship for a memristor with the state dependent Ohm's law. Where $v(t)$ is the applied voltage, $i(t)$ is the current through the memristor, $G(x, t)$ is the conductance of the memristor (memductance), which depends on the internal state variable or vector of state variables x and the time t .

$$\frac{dx}{dt} = f(x, v(t), t) \quad (3.2)$$

Equation 3.2 describes the evolution of the internal state of the memristor. Here x is the internal state variable (or vector of state variables) that represents, for example, the position of a boundary, ionic concentration, or filament length, while $f(x, v(t), t)$ is a function describing how the state evolves in response to the applied voltage and time.

The Discrete Statistical Model

As outlined in Section 3.1, developing a high-performance simulation model for memristors at the system level, while simultaneously accounting for the many factors that influence device behavior, is a significant challenge. Incorporating multiple influencing factors can make the state evolution function $f(x, v(t), t)$ increasingly complex and computationally demanding (see for example [A176]).

To address this issue, the Discrete Statistical Model (DStM) was developed by the author of this thesis. The DStM reduces overall complexity by representing and modeling discrete operating points of specific memristor devices, rather than attempting to capture all possible influences in a continuous and highly detailed fashion. This approach enables efficient simulation while still incorporating the key factors that impact device behavior.

The DStM can be viewed as a bridge between the *device level*, where stochastic conductance variations are directly observed, and the *system level*, where each memristor cell must be represented by a compact and computationally tractable state model. Each device state is therefore represented by a discrete index x , and its electrical behavior is statistically characterized through the probability distribution $\mathcal{D}_x$, which is derived from experimental data under well-defined operating conditions.

In accordance with the general memristor equations, the DStM can be formulated as follows:

$$x(t+1) = x_{\text{target}} \quad (3.3)$$

$$G(x, t) \sim \mathcal{D}_x \quad (3.4)$$

Here, $x(t)$ is the index of the current conductance state at time t , and x_{target} is the index of the targeted state set by a write operation. $G(x, t)$ represents the measured device conductance at state x , and the probabilistic term $G(x, t) \sim \mathcal{D}_x$ replaces detailed physical modeling with an empirically observed statistical description. In the DStM the conductance $G(x, t)$ is sampled from the probability distribution $\mathcal{D}_x$, which captures the statistical variability observed in measurements for state x . Because of the discrete operating points, only the discrete indices (integer) x for each simulated memristor cell needs to be stored, instead of an analog state variable such as $G(x(t), t)$. During a write operation, the internal state of a device (x) is updated to the targeted conductance state ($x \rightarrow x_{\text{target}}$). When a read operation is simulated, the output conductance is obtained by sampling from the distribution associated with the current state, $\mathcal{D}_x$.

Importantly, this description is not limited to a particular device type or fabrication methodology: by replacing the state-dependent distributions $\mathcal{D}_x$ with those obtained from different devices, technologies, or even from theoretical models, the DStM framework can be generalized to a wide range of memristor models. This flexibility makes the DStM applicable beyond

the specific devices studied in this thesis and enables adaptation to future technologies as new data or models become available.

Overview of the DStM Flow

The DStM is an empirical model, with the central idea that it can be defined entirely by measurements: all operational and fabrication influences, including peripheral circuitry and operating conditions, are implicitly captured by statistically analyzing real device data. For each combination of device type and operating condition, a unique set of measurements defines the statistical distributions, that form the basis of the model. The following subsection details how the empirical device data that define $\mathcal{D}_x$ in Eq. (3.3) are obtained and integrated into the DStM.

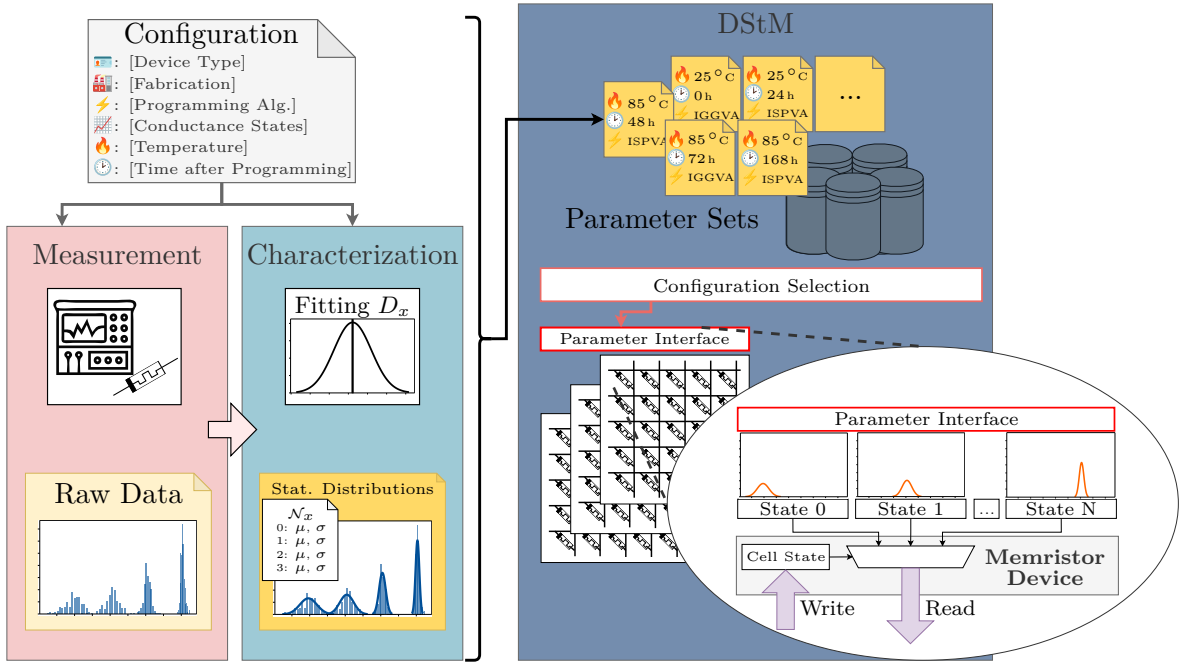


Figure 3.2.1: Overview of the workflow of the DStM.

Figure 3.2.1 illustrates the conceptual workflow of the DStM. Here, memristor devices are integrated into the model by defining a configuration specifying the device type and relevant operating conditions. Based on this configuration, measurements are performed on multiple devices, each programmed and read out multiple times to a range of conductance states. This process ensures statistically reliable data for each state and enables the characterization of both time-dependent and time-invariant variability effects. The choice of measurement strategy directly determines which sources of variability are captured in the model. To comprehensively represent both time-dependent and time-invariant variability, measurements should be performed across multiple devices, states, and repeated over time. If only specific effects are of interest, such as time-dependent variability, the measurement protocol can be adapted, for example, by repeatedly cycling the same devices to exclude device-to-device variations.

The collected measurement data is then characterized by fitting normal distributions to the conductance values observed in each target states ($\mathcal{D}_x = \mathcal{N}_x(\mu_x, \sigma_x)$). Normal distributions were chosen because they provided the best fit for the majority of the observed data of measurements from IHP RRAM devices (compare [A32, A177]). However, it is assumed that the same holds true for other memristor types, since conductance variations typically result from the sum of many small, independent factors and as described by the central limit theorem: the sum of many independent random variables tends toward a normal distribution [A178]. For some specific target states, alternative distributions might offer a slightly more accurate fit. However, in general, normal distributions are a suitable choice, especially when aiming to avoid the overhead and complexity of modeling multiple different distributions within a single simulation framework.

The parameters of the fitted distributions are stored in parameter files, each corresponding to a specific device and set of operating conditions. During simulation, the user selects the relevant configuration, and the model loads the corresponding parameters for the statistical distributions $\mathcal{D}_x$. Each simulated memristor cell is assigned a state index x , which is updated on write operations. When a read operation occurs, the conductance value is sampled from the distribution associated with the current state and returned as the output of the cell.

In summary, the workflow shown in Fig. 3.2.1 provides the practical link between the formal DStM definition in Eq. (3.3) and its use in system-level simulations, by defining all statistical parameters $\mathcal{D}_x$ and the corresponding operating conditions directly from measured device data.

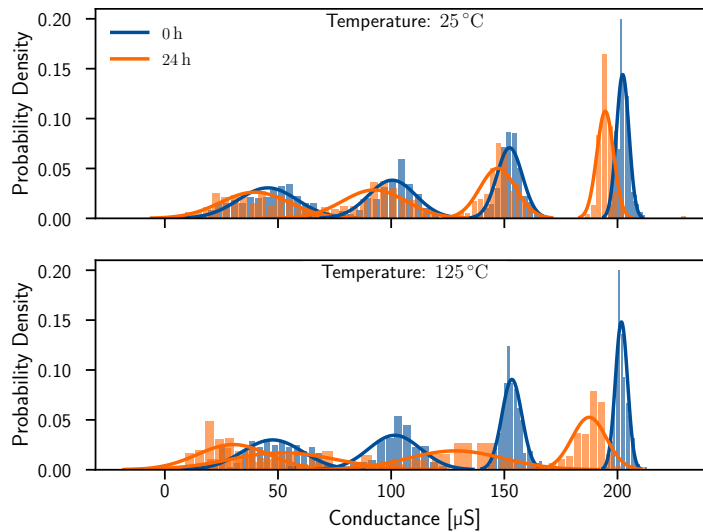


Figure 3.2.2: Examples of measured conductance values and fitted normal distributions for multiple target states in IHP 250 nm CMOS-integrated RRAM devices.

As an illustrative example, Figure 3.2.2 presents measurement data from IHP RRAM devices integrated in 250 nm CMOS technology [A179]. The histograms show the distribution of conductance values obtained for several programmed target states across multiple devices and programming cycles. Each distribution corresponds to a specific targeted conductance state. Shown are four different operation modes: 0 h and 24 h after programming, at a temperature

of 25 °C and 125 °C. The overlaid curves represent the fitted normal distributions for each state, as employed in the DStM.

This measurement-based approach allows the DStM to faithfully reflect real-world device variability and operational characteristics, while remaining computationally efficient for system-level simulation.

Modeling Assumptions and Limitations

The Discrete Statistical Model (DStM) developed in this work is based on a set of practical and physically motivated assumptions that determine its scope, applicability, and efficiency for system-level simulation. The key modeling assumptions are as follows:

1. Discrete Operation Mode

The DStM focuses explicitly on the discrete operation of memristors, where only a set of pre-defined conductance states (operation points) are considered. This approach is well suited for the simulation of applications such as non-volatile memory (NVM), multi-level storage, in-memory computing, memristor-based logic, and physical unclonable functions (PUFs), where information is encoded in distinct, well-separated resistance levels. Purely analog applications, where continuous switching dynamics are of central interest, are not targeted by this model. Even for analog circuits, the DStM can be used if only extreme states (e.g., maximum or minimum conductance) are relevant to the analysis. Continuous switching behavior (including device forming and gradual transitions) is not modeled explicitly. The internal state of each simulated cell is represented solely by the index of its current conductance state. Dynamic transitions between states are abstracted away. The model updates the state index on write events and samples the output value from the corresponding distribution on read events. The DStM assumes typical cell architectures such as 1T1R (one transistor, one memristor) or similar configurations that avoid sneak paths and enable individual cell access. The effects of leakage, wiring, interconnects, and selection device characteristics are considered only insofar as they influence the measured device behavior. More complex array-level effects are outside the direct scope of the model but can be incorporated via additional circuit models if required. This discretization greatly simplifies simulation and aligns with digital or quantized-use cases. However, it also limits the applicability of the model for continuous or analog memristive behavior, where gradual conductance evolution or pulse-dependent dynamics play a central role.

2. Captured Effects

All relevant effects, including programming algorithms, peripheral circuitry (e.g., access transistors, current-limiting schemes, ADCs), and both time-dependent (cycle-to-cycle, temporal drift) and time-invariant (device-to-device, spatial) variability, are implicitly incorporated by extracting the underlying statistics from empirical measurements taken under defined operating conditions. Set/Reset protocols, pulse timing strategies, compliance current/voltage constraints, readout circuit details, and other circuit-level influences are included to the extent that they affect the measured conductance states and their variability, though typically their contribution is much smaller than the intrinsic variability of the

memristor itself. Environmental variability (e.g., due to temperature or supply voltage) is also included as dictated by the measurement data. All observed variability, including that introduced during programming, is reflected in the distributions from which values are sampled. However, effects not represented in the measurement process, such as rare switching anomalies, long-term degradation, or array-level electrical coupling, remain outside the model's scope. Since all these effects are implicitly represented through the measurement data, the accuracy of the model depends critically on the completeness and quality of the underlying measurements. Any measurement errors, insufficient sampling, or incomplete coverage of operating conditions directly translate into modeling inaccuracies. Moreover, the parameterization of the DStM is explicitly coupled to the operating conditions under which the data were obtained. Each combination of device type, biasing scheme, and environmental conditions results in a distinct parameter set. While this coupling limits the validity of the model to the measured conditions, it also provides a clear and physically meaningful connection between simulation parameters and real devices: an advantage over traditional system-level models that rely on abstract fitting parameters without experimental grounding.

These assumptions ensure that the DStM provides a fast, empirically grounded, and broadly applicable abstraction of memristor devices for system-level studies, provided that suitable measurement data is available. Extra-Functional properties such as energy and latency are not part of the memristor model itself, but can be considered in a system-level framework. Same as the variability, these depend strongly on operating conditions, programming algorithms, and peripheral circuitry. Overall, these modeling assumptions and their corresponding limitations define the scope of validity for the DStM: it is optimized for system-level studies that require statistical realism and computational efficiency, but not detailed physical modeling of continuous switching or low-level parasitics.

3.3 Implementation

For system-level evaluations, memristor models must efficiently handle large device arrays and accurately capture device-level variability while maintaining computational performance. However, many existing implementations trade off realism for speed or lack seamless integration with system-level simulation frameworks. To address these limitations, the DStM was implemented in C++ as a high-performance, portable, and flexible solution for system-level simulation of memristor devices. The implementation aims to enable efficient simulation of large-scale memristor arrays, support both parallel and individual device operations, and ensure straightforward integration with virtual prototyping environments such as SystemC. The codebase is modular and extensible, allowing new device effects to be incorporated or the model to be adapted for specific research requirements. In addition to the C++ software-based simulator, a VHDL implementation of the DStM was developed for FPGA-based hardware emulation, which enables real-time hardware-in-the-loop testing (see Section 5.4). The following section focuses on the structure and design of the C++ implementation.

The DStM models memristor behavior by representing the state of each device as an index corresponding to one of several predefined conductance levels. This discretized representation reflects the physical characteristics of real memristor devices, which typically exhibit a limited number of stable conductance states determined by their material and circuit design.

To optimize simulation speed and minimize memory overhead, the set of available conductance states and their corresponding target values are defined at compile time. This approach eliminates the need for dynamic memory allocation during simulation and ensures deterministic behavior across large-scale device arrays. At the same time, flexibility is maintained by allowing only a subset of these predefined states to be used in specific simulations, enabling the model to adapt to different experimental or application scenarios.

Class Structure and Core Functionality

The C++ implementation of the DStM follows a modular class structure designed to balance extensibility, clarity, and computational efficiency. The implementation is built around three core classes:

- *memristors*: the main template class representing a collection of memristor devices, responsible for managing internal states and supporting read/write operations.
- *effect*: an abstract base class (interface) that defines the protocol for implementing behavioral effects (e.g., general variability, stuck devices, or specific variability sources).
- *variability*: a concrete implementation of *effect* that applies state-dependent variability as defined by the DStM. This class loads statistical distributions for each conductance state from measurement-based parameter files.

This modular design separates the responsibilities of state management and effect application, which provides several key advantages. By decoupling these components, the model can represent both ideal and non-ideal device behavior within the same framework. Specifically:

- Simulations without active effects correspond to ideal, variability-free operation, which serves as a useful baseline for comparison.
- Multiple effects can be combined and layered dynamically during simulation to study interactions between different variability sources.
- Effects can be enabled or disabled at runtime, enabling experiments such as simulating conductance drift by switching parameter sets after programming.
- Different sources of variability (e.g., time-dependent, device-to-device, stuck-at faults) can be implemented and evaluated independently or in combination.
- Write (programming) operations are computationally efficient, as only the internal state index is updated. Variability effects manifest only during read operations, where conductance values are generated on demand. This event-driven approach closely reflects physical device behavior, where time-dependent and time-invariant effects become apparent only upon measurement.

memristors Class

Efficiently simulating large collections of memristor devices poses two key challenges: managing thousands of device states without excessive memory overhead, and supporting flexible configurations ranging from single devices to full crossbar arrays. The *memristors* class ad-

dresses these challenges by providing a scalable container for device states and an interface for applying behavioral effects during simulation.

```

1  template<size_t HEIGHT, size_t WIDTH>
2  class memristors {
3  private:
4      std::array<uint8_t, WIDTH*HEIGHT> cell_states{};
5      std::vector<std::unique_ptr<effect::effect>> memristor_effects;
6  public:
7      memristors();
8      ~memristors();
9
10     void addEffect(std::unique_ptr<effect::effect> e);
11     void clearEffects();
12     void printActiveEffects();
13
14     constexpr size_t getSize();
15     static constexpr size_t getNumStates();
16
17     void write(size_t addr, std::span<const uint8_t> states);
18     void read(size_t addr, std::span<double> data) const;
19 };

```

Listing 3.1: Definition of the *memristors* class in the C++ implementation of the DStM.

The *memristors* class is a template, with *HEIGHT* and *WIDTH* as parameters. It can simulate an $n \times m$ array of devices, enabling efficient vectorized (parallel) access. But it can also simulate a collection of individual devices. Device states are stored as 8 bit integer indices in a flat array, representing the device's targeted conductance state. This approach supports high-performance simulations of both large arrays and independent devices. The class provides methods to add or remove effects at runtime and to print active effects for evaluations or debugging. The write method efficiently updates one or more internal states at specified addresses. The read method, if effects are active, delegates to the effect(s) to generate conductance values. If no effects are active, the indices are directly mapped to their nominal target conductance values, representing idealized operation.

effect Base Class

Accurately reproducing non-ideal device behavior requires the ability to represent a wide range of physical effects, such as device variability, faults, or drift, without altering the core simulation logic. The *effect* class addresses this challenge by providing an abstract interface for defining and applying such effects to memristor devices. It establishes a uniform protocol that allows new physical phenomena to be introduced as independent software modules while maintaining compatibility with the existing framework.

```

1 namespace effect
2 {
3     class effect
4     {
5     protected:
6         virtual void print(std::ostream& o) const = 0;
7     public:
8         virtual ~effect() = default;
9         virtual void on_read(
10             std::span<double>& read_data,
11             std::vector<uint8_t>& read_states
12         ) = 0;
13         friend std::ostream& operator<< (std::ostream& o, const effect& e){
14             e.print(o);
15             return o;
16         }
17     };
18 }

```

Listing 3.2: Definition of the abstract *effect* base class.

Derived classes implement the *effect* interface to model specific phenomena such as device variability or faults. The DStM specifies a generic *variability* effect derived from measurements, which includes all measurable variability sources (time dependent and time invariant). However, specific measurements can be conducted to isolate only specific sources of variability (e.g. only measuring the same device multiple times to isolate cycle-to-cycle variability). This effect can then be implemented in a different derived class. Derived classes can be registered as active effects inside the *memristors* class. The *print* method and « operator are provided for evaluation and debugging, printing specific information about the implemented effect. The *on_read* method is called by the *memristors* class during read operations. It receives the cell state indices of the devices to be read and fills the result container with the corresponding conductance values (applying the effect). For derived classes, implementing time dependent variability, the concrete implementation of the *on_read* method can generate a different value for each device for each function call. For derived classes, implementing time invariant variability, the concrete implementation of the *on_read* method can for example generate the same offset value for the same device every time the function is called, if only device-to-device variability is modeled.

variability Class

Accurately modeling memristor behavior requires capturing the statistical variability observed in physical measurements. Such variability arises both across devices (device-to-device differences) and over time (cycle-to-cycle or temporal fluctuations). The *variability* class provides a concrete implementation of the abstract *effect* interface to model these phenomena within the DStM. It realizes the general variability source defined by the model, encompassing both time-dependent and time-invariant components derived from experimental data.

```

1 namespace effect
2 {
3     class variability final : public effect
4     {
5     private:
6         std::array<chance, cfg_memristor_num_states> _variabilityGenerators;
7         bool _generatorsInitialized = false;
8         std::string _jsonFilePath;
9         void initGenerators();
10    protected:
11        void print(std::ostream& o) const override;
12    public:
13        explicit variability(const std::string& jsonFilePath);
14        ~variability() noexcept override = default;
15        void on_read(
16            std::span<double>& read_data,
17            std::vector<uint8_t>& read_states
18        );
19    };
20 }

```

Listing 3.3: Definition of the derived *variability* class.

The *variability* class models statistical variations in device conductance by sampling from per-state normal distributions defined in measurement-based parameter files (stored in JSON format). During construction, the *initGenerators* method loads and initializes the distribution parameters, creating one random generator for each discrete conductance state. The *on_read* method generates output conductance values by sampling from the appropriate normal distribution for each state. The underlying random number generation uses the 32-bit Mersenne Twister pseudorandom number generator (PRNG), transformed to a normal distribution using a so-called *random variate generation algorithm* inside the *chance* class. Here, the implementation of the random variate generation algorithm is actually depending on the C++ standard library implementation e.g. [A180].

Summary

Together, these classes form a modular and extensible framework that balances physical accuracy with computational efficiency. The *memristors* class provides a scalable foundation for representing large device arrays, while the *effect* interface defines a clear abstraction layer for incorporating additional behavioral phenomena without modifying the core simulation logic. Concrete implementations such as the *variability* class demonstrate how measurement-based effects can be seamlessly integrated into the model, enabling realistic and configurable system-level evaluations. This architecture ensures that new effects or device characteristics can be added with minimal code changes, supporting ongoing research into emerging device behaviors and system-level interactions.

3.4 Evaluation

This section evaluates the performance of the DStM in comparison to existing memristor models from the literature. The evaluation focuses on two main criteria: the accuracy of

the simulated device behavior with respect to measured data, and the computational efficiency (simulation speed) achieved by the implementation. Where appropriate, representative benchmarks and reference models are used to highlight the strengths and limitations of the DStM in typical system-level simulation scenarios.

Accuracy

Assessing and comparing the accuracy of different memristor models is inherently challenging, due to the lack of standardized benchmarks and the device-specific nature of most modeling approaches. The majority of published models are developed and parametrized for specific memristor types (such as OxRAM, PCM, or STT-MRAM) and must be individually fitted to measured data from actual devices in order to reflect their true behavior. As a result, direct comparisons between models are only meaningful when all are calibrated against identical measurement datasets.

Additionally, since memristor behavior is strongly influenced by intrinsic device variability and stochastic switching mechanisms, accuracy assessments should be based on statistical analysis across multiple devices and programming cycles, rather than on single-device measurements. For these reasons, the evaluation in this section emphasizes a comparison to measured data, highlighting how well the DStM can reproduce the observed statistical distributions of conductance states in practical devices.

While a Stanford-PKU-based model fitted to IHP 250 nm devices exists [A181], its parameters correspond to devices with a different structure and only partially specified operating conditions (such as programming strategy, temperature, or time after programming). Moreover, the Stanford-PKU model accounts for only selected sources of variability based on physical assumptions [A182], whereas the DStM uses statistical distributions derived directly from measurements, thereby capturing the full effective variability. Given these differences in methodology and modeling scope, the accuracy of both models cannot be directly compared. For this reason, the evaluation focuses on (a) direct reproduction of measured distributions, and (b) comparison to representative empirical models (such as the approaches used in NeuroSim, MemTorch or SySCIM), which are commonly used for system-level simulation.

The accuracy of both the idealized variability modeling (as used in, for example, NeuroSim [A156] or SySCIM [A37]) as well as the DStM approach depends on how well the assumed statistical distributions reflect the actual device behavior under specific operating conditions. Because device characteristics can vary considerably with those conditions, model accuracy is inherently use-case specific. To illustrate the differences between the modeling approaches, a representative example is presented in the following. It should be emphasized, however, that the goal of this comparison is not to claim superior accuracy for this particular dataset, but to demonstrate how the proposed DStM framework systematically captures realistic device statistics across use cases where an idealized model cannot be reliably calibrated.

Figure 3.4.1 presents histograms of measured conductance values for IHP 250 nm RRAM devices programmed to four target conductance states (50 μ S, 100 μ S, 150 μ S, and 200 μ S) using the IGVVA algorithm (for details about this programming algorithm see [A183]). All measurements were performed at 85 °C and 24 h after programming. In addition to the histograms, the figure shows probability density functions (PDFs) for both the DStM approach,

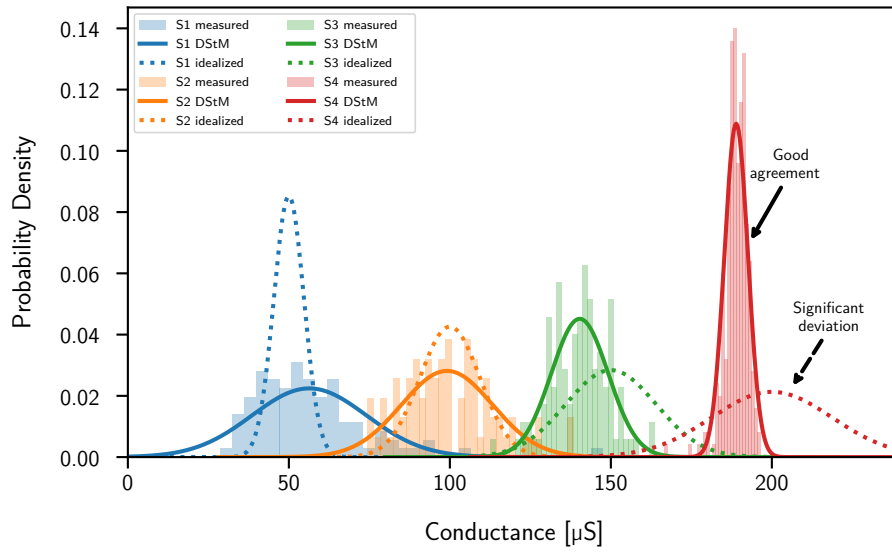


Figure 3.4.1: Comparison of measured and modeled conductance distributions for four conductance states in IHP 250 nm RRAM devices.

where a normal distribution is fitted individually to each state, and for the idealized variability model, where each distribution is centered at the target value and assigned a user-defined relative standard deviation.

In the DStM approach, the normal distributions closely match the measured data for each state, capturing effects such as retention-induced shifts in the mean and the generally lower variability observed for higher conductance states. In contrast, the idealized approach assumes identical relative variability for all states and ignores systematic deviations from the target values. In this example, the devices actually exhibit decreasing relative and absolute variability for higher conductance states, which the idealized model fails to capture. Furthermore, the idealized model requires the user to manually specify the relative standard deviation, yet provides no guidance on how to select a physically meaningful value under real operating conditions.

To ensure a fair and informative comparison, the relative standard deviation (the sole parameter of the idealized model) was tuned to achieve the best possible agreement with the measured data, by minimizing the average distance between the modeled and measured distributions. This distance can be quantified using the Wasserstein metric (also known as the Earth Mover’s Distance), which measures the average shift required to transform one distribution into another [A184] and is expressed in the same units as the values (in this case, μS). The Wasserstein metric was chosen over alternatives such as the Kolmogorov-Smirnov statistic because it reflects the overall distributional difference and provides a more physically meaningful, interpretable measure for engineering applications. Lower values in this metric indicate better agreement with the reference data. In this case, the optimal value for the user-defined relative standard deviation was found to be 0.083. This optimization leads to an upper bound on the accuracy that such idealized models can achieve when perfectly calibrated. In practical scenarios, however, users typically lack the required experimental data

for such tuning and must instead rely on heuristic or literature-based parameter choices. The DStM approach, in contrast, determines all parameters directly from measurement data and thus achieves this level of agreement by construction, without requiring manual calibration. Consequently, while the numerical comparison highlights the achievable accuracy difference in this specific example, the broader point is methodological: DStM provides a systematic framework that generalizes to arbitrary devices and operating conditions, whereas idealized approaches inherently depend on ad-hoc user inputs, and are severely limited by the a priori assumptions.

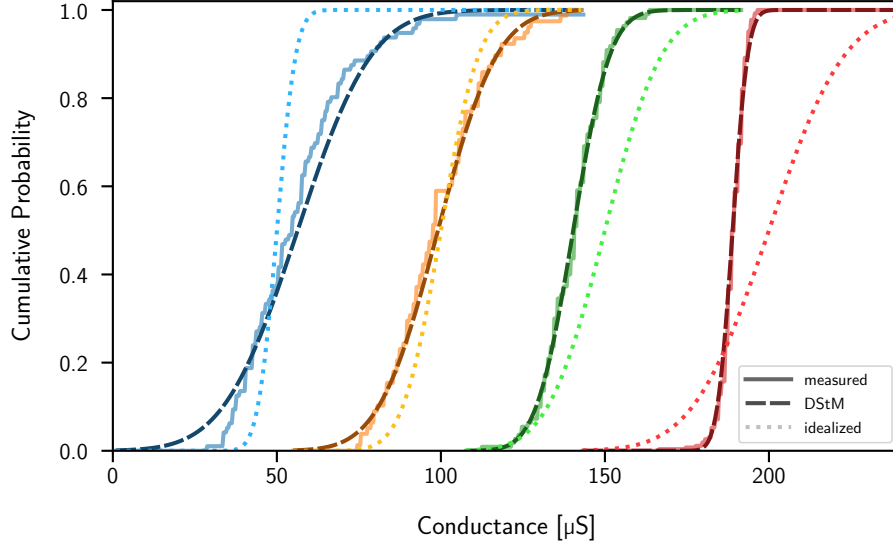


Figure 3.4.2: Comparison of measured and modeled cumulative distributions for IHP 250 nm RRAM devices.

For this analysis, empirical cumulative distribution functions (CDFs) were generated for each conductance state by repeatedly sampling from both the measured dataset and the modeled distributions (100000 samples). Figure 3.4.2 presents these CDFs and further illustrates the differences between the DStM and idealized modeling approaches. Both the measured and DStM CDFs exhibit gradual transitions and extended tails, capturing the full range of device variability. On the other hand the idealized CDFs rise more abruptly, reflecting an underestimation of the likelihood of outlier values. This effect is especially pronounced in the lower and upper tails of the distributions. Here the cumulative probability of the idealized model rapidly approaches 0 or 1. Quantitatively, the DStM model achieves a mean Wasserstein distance to the CDF of the measurements, that is about six times lower than that of the idealized model ($1.6 \mu\text{S}$ vs. $9.7 \mu\text{S}$, respectively). This corresponds to an 84% improvement in accuracy for these devices and operating conditions. Such differences can have significant implications for the robustness of multi-level applications, where accurate modeling of rare events and state overlap is critical.

To evaluate the impact of statistical modeling accuracy on classification performance, confusion matrices were constructed for the four RRAM conductance states. These matrices represent the probability of correctly or incorrectly assigning a state to a read device value,

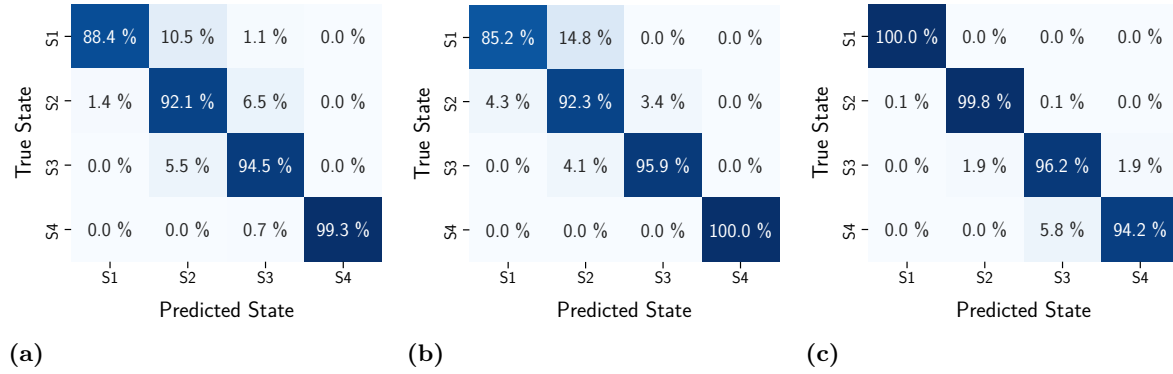


Figure 3.4.3: Confusion matrices for the classification of four RRAM conductance states: (a) sampled from measurement data, (b) predicted using the DStM model, and (c) predicted using an idealized variability model.

and thus directly reflect the likelihood of state transitions or classification errors in practical applications.

The matrices were generated by sampling 2,000 conductance values from (a) the measured data, (b) the DStM model, and (c) the idealized model. Each sampled value was assigned to a predicted state based on thresholds set at the midpoints between the four target conductance values, mirroring the typical readout strategy in actual circuits.

As shown in Figure 3.4.3, the measured data (a) and DStM model (b) both capture the fact that most misclassifications occur in the lowest conductance states, where device variability is highest. In contrast, the idealized model (c) strongly underestimates the probability of state overlap and misclassification for these states, while overestimating errors for the highest conductance state. This demonstrates that the idealized approach, by imposing a uniform relative variability, does not faithfully reproduce the actual error characteristics of the device. The DStM approach, in contrast, offers a more accurate reflection of state-dependent behavior and provides a more reliable foundation for system-level simulation.

Simulation Speed

Efficient simulation speed is a crucial requirement for the practical application of memristor models in large-scale system-level studies. In this section, the computational performance of the proposed DStM is evaluated and benchmarked against several established memristor models that are commonly used in the literature and in system-level simulation frameworks.

Selection of Reference Models

Comparing simulation speed across memristor models is challenging due to differences in intended use and integration within various frameworks. Therefore, the focus here was on reference models based on their relevance for system-level studies and their implementation in widely-used simulation toolflow. The following models and tools were considered:

- **MemTorch:** Provides a collection of key literature models implemented in Python, including the Joglekar model [A141], the HP/Linear Ion Drift (LID) model [A14], the

Stanford-PKU model [A147–A149], the VTEAM model [A136] and the Data-Driven model [A151, A152]. These models are representative examples and their MemTorch implementations support direct benchmarking outside the MemTorch framework. Therefore, these implementations were selected for comparison here.

- **NeuroSim:** Contains an application-specific model tightly integrated with the inference engine, making isolated benchmarking of the memristor model infeasible.
- **JART-VCM Model:** This model is a compact model for RRAM VCM devices, fitted to real measurements. Its code is openly available. And while it is originally implemented in MATLAB it was included here as an example for a more device specific model and ported to C++ for fair comparison.
- **Linear Ion Drift Model:** While a version of this model is already included in this comparison, its implementation from the MemTorch framework is in Python. Because of its low complexity it is expected that this model will result in a high simulation speed. Therefore, to isolate the effect of implementation language, the LID was also ported to C++ and benchmarked alongside the DStM model.

It should be noted that all models are benchmarked as is in isolation to provide a baseline for simulation speed. In a full system-level simulation, additional computational overhead from the simulation of peripheral circuitry and system components will further increase the total simulation time.

It should also be noted, that none of the compared models natively support device variability effects. Therefore, for a fair evaluation, we benchmark the DStM model both with variability effects enabled and disabled.

Measurement Methodology

For DStM, JART-VCM (C++), and LinearIonDrift (C++), the Google Benchmark library was used [A185], which provides reliable micro-benchmarking with automatic iteration, averaging and precise tracking of simulated accesses. For MemTorch models, custom measurement code was implemented based on Python’s *time.perf_counter()*, averaging over multiple thousand runs. All models are implemented as single-core applications, so no parallelization artifacts are expected. For the write benchmark, the measured time corresponds to simulating the process of setting or programming the memristor devices into a specific conductance state. For the read benchmark, the devices are first set to a defined conductance state, after which the time required to read out the internal state (or to simulate the application of a voltage pulse and return the resulting current) is measured. This approach ensures a consistent and comparable evaluation across models, while accounting for differences in the way each model implements state-setting and readout procedures.

Benchmark Metric and Platform

Simulation speed is measured as the number of device operations (reads and writes) performed per second, averaged over multiple iterations. All benchmarks were conducted on a desktop PC equipped with an AMD Ryzen 5 3600 CPU (6 cores, 12 threads; L1 cache: 32kB data + 32kB instruction per core, L2 cache: 512kB per core, L3 cache: 32MB shared), running

Ubuntu 24.10. To minimize external influences, the CPU governor was fixed to *performance* mode, all turbo and boost features were disabled, and no user tasks were running in the background.

It should be noted that the absolute simulation speed depends on the test system (especially CPU architecture and cache sizes), but the relative performance across models is representative for typical research workflows.

Simulation Speed of the DStM Model

The raw simulation performance of the proposed DStM model was evaluated by measuring the number of simulated access operations per second as a function of the number of simulated devices. Three scenarios were considered: pure read access, pure write access, and combined access (one write followed by one read). Each scenario was benchmarked with and without the variability effects enabled in the DStM, in order to quantify the computational overhead introduced by the stochastic variability modeling. Results are presented in Figure 3.4.4, which shows the simulation throughput versus the number of simulated devices, with separate subplots for the two configurations (variability disabled and enabled). The curves show the average speed for write, read, and a combined one-write-one-read operations.

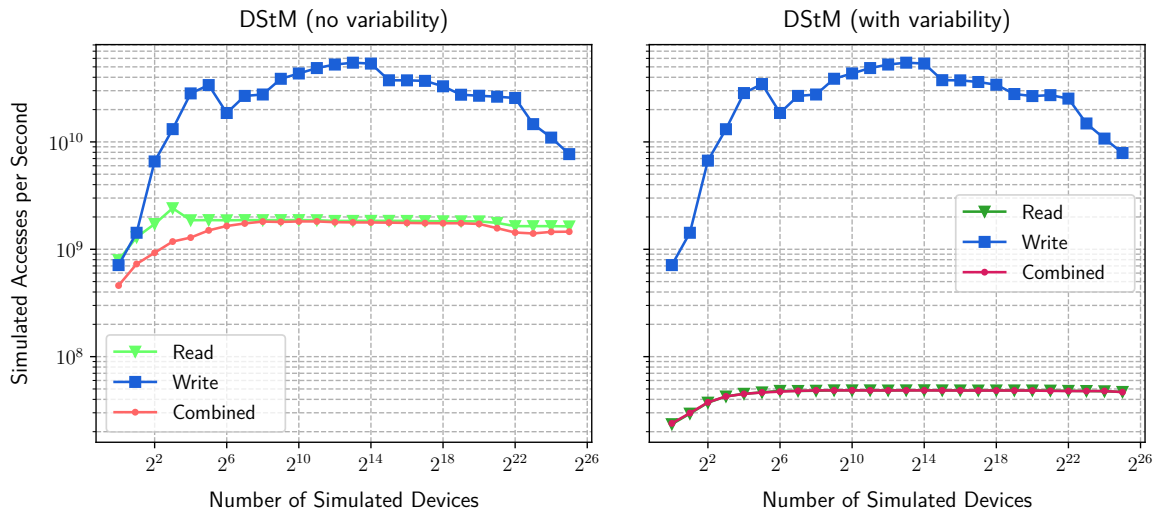


Figure 3.4.4: Comparison of per-operation simulation speed for the DStM memristor model. Left: Baseline implementation simulating ideal target conductances (no effects active). Right: Full implementation including realistic device variability (*variability* effect active).

Write operations demonstrate extremely high performance in all configurations, ranging from $7.1 \cdot 10^8$ accesses per second for a single simulated device up to $5.4 \cdot 10^{10}$ accesses per second for 2^{14} devices. This performance remains unchanged by the activation of variability, as the variability effect in the DStM is only applied during read operations.

In contrast, read operations are significantly slower, particularly when variability is enabled. Without variability, read speeds range from $7.9 \cdot 10^8$ to $2.4 \cdot 10^9$ accesses per second (from a single device up to 2^3 devices). Enabling variability reduces throughput to $2.3 \cdot 10^7$ to $4.8 \cdot 10^7$ accesses per second (from a single device up to 2^6 devices), corresponding to a reduction to

approximately 2 % of the original speed and highlighting the computational cost of simulating realistic variability.

For the combined access benchmark (one write plus one read per device), the read operation is the dominant factor. The results closely follow those of the read-only scenario, as the write operation contributes negligible overhead compared to the computational demands of the read operation, particularly when variability is enabled.

For both read and combined access scenarios, throughput initially increases with the number of simultaneously simulated devices (due to more efficient vectorized access operations), but plateaus after approximately 2^6 devices, indicating an optimal amount of simulated devices for this setup. Nevertheless, simulation of a single device remains highly efficient in all tested scenarios.

Comparison of Simulation Speed with Reference Models

To contextualize the simulation speed of the DStM model, the throughput of write and read access simulations was benchmarked against all selected reference models. Figures 3.4.5 and 3.4.6 present the results for write and read accesses per second, respectively, as a function of the number of simulated devices.

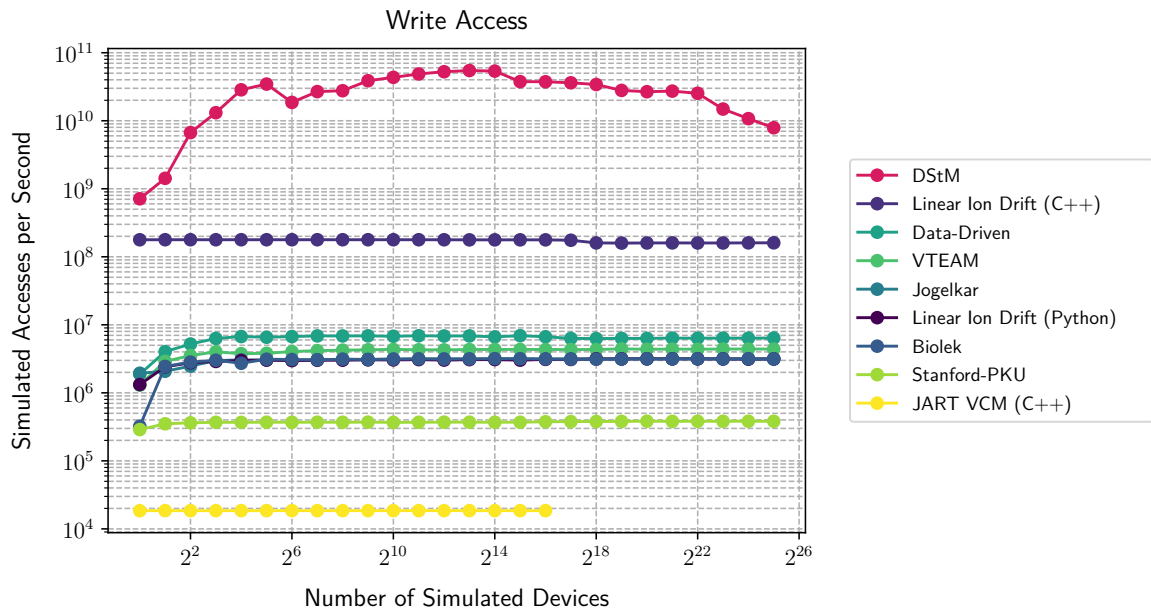


Figure 3.4.5: Write access simulation speed of all evaluated memristor models as a function of the number of simulated devices.

It is important to note that the implementation details of each reference model influence the simulation procedure and, consequently, the measured speed. The MemTorch models provide a direct interface to set the internal state of a device to a target conductance, which allows for rapid state initialization and effectively abstracts away the details of the programming process. As a result, the simulation of write operations is highly simplified, and only the readout process is modeled in more detail.

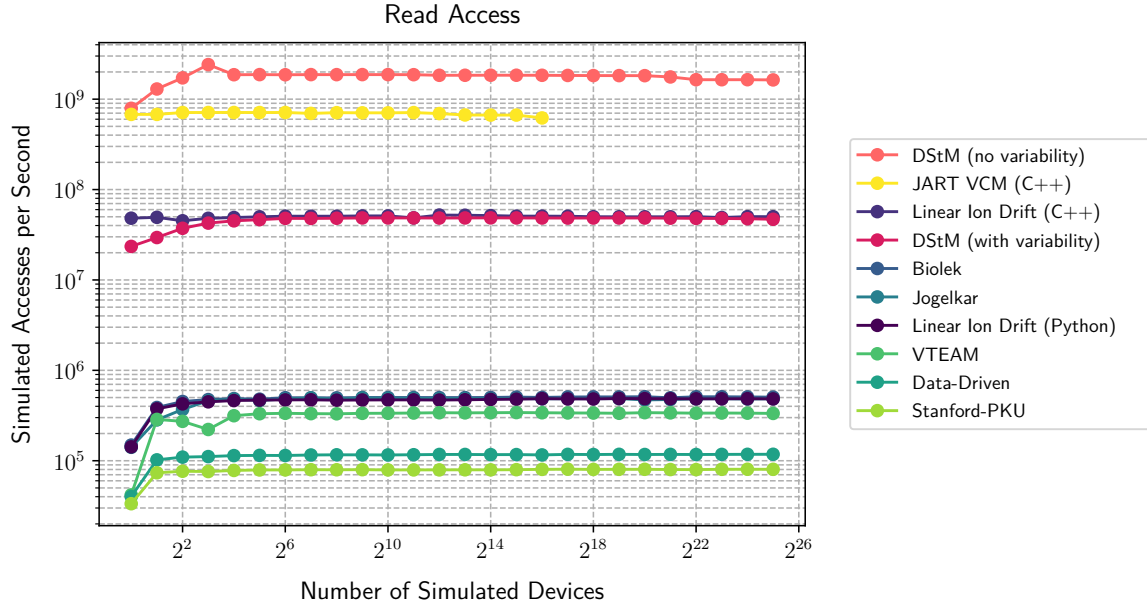


Figure 3.4.6: Read access simulation speed of all evaluated memristor models as a function of the number of simulated devices.

In contrast, the JART-VCM model does not support direct state-setting. Instead, the device state must be programmed by explicitly simulating the effect of voltage pulses, requiring the numerical solution of physical equations over many time steps. This leads to significantly slower write simulation times, as the computational overhead for each programming event is much higher compared to models with direct state manipulation.

These distinctions are critical for interpreting the benchmark results, as they explain the observed differences in simulation speed between the various models and highlight the trade-off between simulation fidelity and computational efficiency.

Additionally, it can be seen, that for most reference models, the simulation speed is largely independent of the number of simultaneously simulated devices, with only a minor increase in throughput observed from single-device simulation up to 2^4 devices. Beyond this point, performance remains almost constant for both write and read accesses.

Write Access

The DStM model demonstrates the highest write access simulation speed among all evaluated models. As previously discussed, the write access speed of the DStM model remains consistent regardless of whether device variability is considered or not. Across the tested device counts, DStM achieves between 4 times ($7.13 \cdot 10^8$ vs. $1.78 \cdot 10^8$ accesses per second for a single device) and up to 300 times higher throughput ($5.37 \cdot 10^{10}$ vs. $1.77 \cdot 10^8$ accesses per second for 2^{14} devices) compared to the second-fastest model, which is the Linear Ion Drift (LID) model implemented in C++. When compared to the fastest model available from MemTorch (Joglekar model), the DStM is up to four orders of magnitude faster. The JART-VCM model implemented in C++ exhibits the lowest write speed, as its write operation explicitly

simulates the physical programming process in fine time steps, involving complex equations and computationally intensive mathematical functions.

Read Access

For read access, the DStM model without variability enabled achieves the highest simulation speed, with up to 3.4 times faster access simulations ($2.41 \cdot 10^9$ vs. $7.13 \cdot 10^8$ accesses per second) compared to the second-fastest model (JART-VCM). The high read speed of the JART-VCM model can be attributed to its approach of updating device parameters during the write operation, leaving the read operation to compute a simple equation based on stored internal parameters. Enabling variability in the DStM model reduces its speed to a level comparable to the LID model in C++, with both operating at approximately 2% of the speed of the DStM without variability (corresponding to the DStM without variability being about 39 times faster). Nevertheless, even with variability enabled, the DStM remains roughly two orders of magnitude faster than the fastest MemTorch model for read accesses ($4.85 \cdot 10^7$ vs. $5.00 \cdot 10^5$ accesses per second for the Biolek model).

It should be noted that the JART-VCM model was only evaluated up to 2^{16} devices for both read and write operations, due to prohibitive simulation times caused by its slow write operation. Moreover, since for the read access simulations of all models a preceding write simulation phase was used to set the internal state to a defined value, the read benchmarking was affected by this problem as well.

These results shown lead to two key conclusions:

- The DStM model is generally among the fastest memristor models evaluated for both read and write access simulation. Even with variability enabled, which other models do not include.
- The evaluated models differ substantially in simulation speed for read and write operations, primarily depending on when and how internal parameters are updated within the simulation workflow.

Therefore, a further analysis of combined access patterns is conducted in the following.

Simulation Speed for Combined Accesses

A dedicated benchmark was conducted to evaluate the simulation speed for combined access patterns, consisting of one write followed by one read operation per device, across all selected models. Speed measurements were performed for both single-device simulation and for an array of 1024 devices (2^{10}), which is representative of practical in-memory computing arrays (e.g., 32×32 crossbars). Results are shown in Figures 3.4.7 (single device) and 3.4.8 (1024 devices), with horizontal bar charts displaying simulated accesses per second.

For single-device simulation, the DStM model without variability achieves the highest throughput, simulating $4.60 \cdot 10^8$ combined accesses per second, outperforming the second-fastest model (Linear Ion Drift in C++, $3.38 \cdot 10^7$ accesses per second) by more than one order of magnitude. The DStM with variability enabled achieves $2.39 \cdot 10^7$ accesses per second, closely matching the performance of the LID model in C++. Among the MemTorch

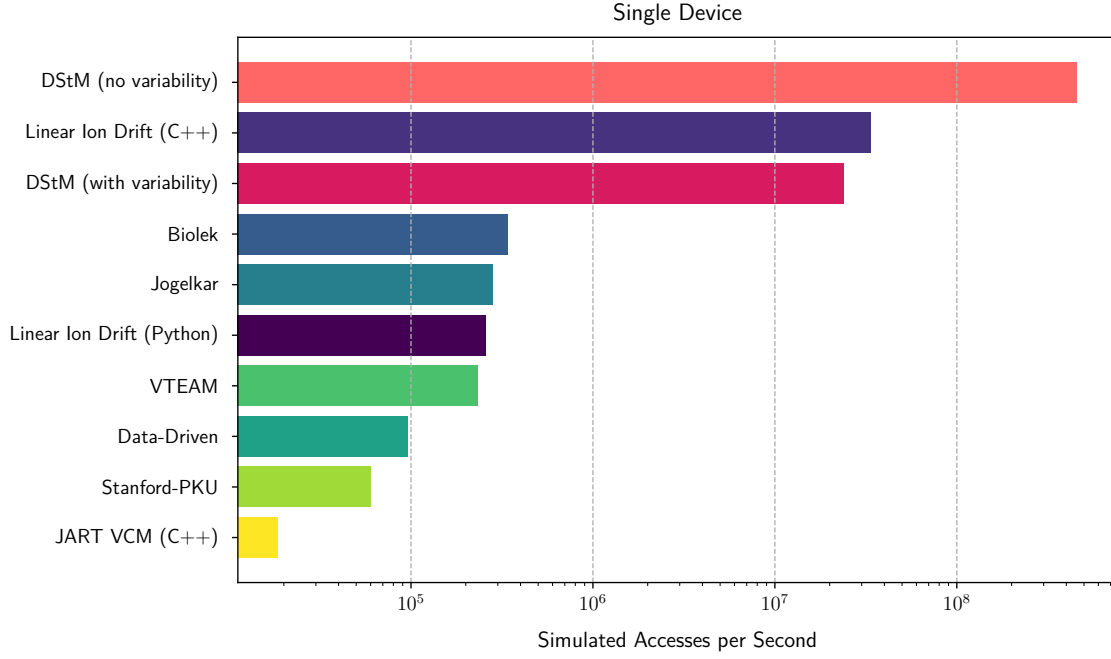


Figure 3.4.7: Simulation speed of a combined access for a single-device simulation.

models, Biolek is the fastest with $3.42 \cdot 10^5$ accesses per second. This is almost two orders of magnitude slower than the DStM with variability. Other MemTorch models perform at similar levels, with the Stanford-PKU model being the slowest in this group. The JART-VCM model in C++ is the slowest overall, achieving only $1.85 \cdot 10^4$ combined accesses per second.

For the simulation of 1024 devices, the DStM model without variability remains the fastest, reaching $1.82 \cdot 10^9$ combined accesses per second. With variability enabled, the DStM achieves $4.85 \cdot 10^7$ accesses per second (about 2.7 % slower), but still remains the second-fastest model overall. In this scenario the DStM can benefit from the parallelization of access simulations. The LID model in C++ closely matches the DStM with variability, with an average difference of simulated accesses below 1 %. The rest of the ranking mirrors the single-device case: Biolek is the fastest of the MemTorch models but still two orders of magnitude slower than the DStM with variability enabled. The Stanford-PKU model (MemTorch) and JART-VCM (C++) remain the slowest.

These results demonstrate that, for both realistic simulation cases, the DStM model offers optimal simulation speed. Furthermore, combined with the evaluation of the read access simulation it can be seen, that even in use cases with a predominance of read accesses, the DStM remains among the three fastest models. Notably, it archives this while being the only one to simulate realistic device variability, which adds additional cost.

3.5 Discussion of the DStM

In summary, the DStM model presented in this chapter offers a practical and highly efficient solution for realistic memristor variability modeling at the system level. It achieves **up to**

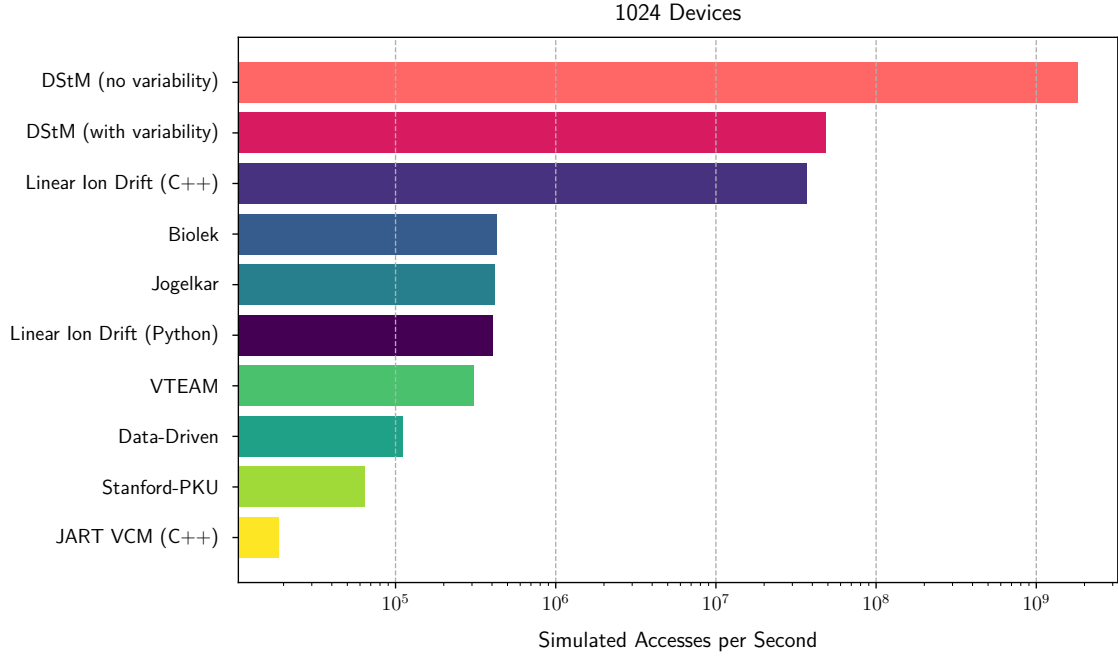


Figure 3.4.8: Simulation speed of a combined access for a simulation of 1024 devices.

84% **higher accuracy** compared to established system-level approaches, while enabling simulation speeds **up to two orders of magnitude faster** than the fastest prior models that do not even model variability. The DStM is technology-agnostic, requires no a priori knowledge, and can be easily adapted to new measurement data, making it highly flexible for a range of devices and applications.

The simulation performance is high for single device simulations as well as large-scale array scenarios and across a variety of read-write ratios. This positions the DStM as an ideal choice for system-level studies. However, model fidelity is inherently limited by the quality of the underlying measurement data. The applicability of the model is restricted to discrete operating points and continuous device behavior is not represented. For later stages of a computer system development, such as circuit design and prototyping, more physically detailed models may be necessary. Finally, the DStM is focused on modeling device behavior and does not capture extra-functional properties such as energy, latency, or area. These aspects are addressed in the subsequent chapter through integration into a system simulation framework.

Chapter 4

Memristor-Based System Simulation Framework

Designing and evaluating computer architectures that leverage memristor-based devices presents significant challenges, especially when it comes to accurately capturing their complex behavior and integrating them with conventional system components. As memristor technologies mature and enable a wider range of applications, from hybrid memory and storage to in-memory computing accelerators, the need for flexible, system-level modeling and simulation frameworks becomes increasingly urgent.

System-level virtual prototyping is a well-established approach for exploring design spaces, validating concepts, and quantifying extra-functional properties such as energy consumption and latency prior to fabrication [A102]. Among the various tools available, SystemC stands out as a widely adopted system-level modeling language and simulation environment, offering high flexibility through its C++ foundation and enabling seamless integration of both hardware and software components [A186].

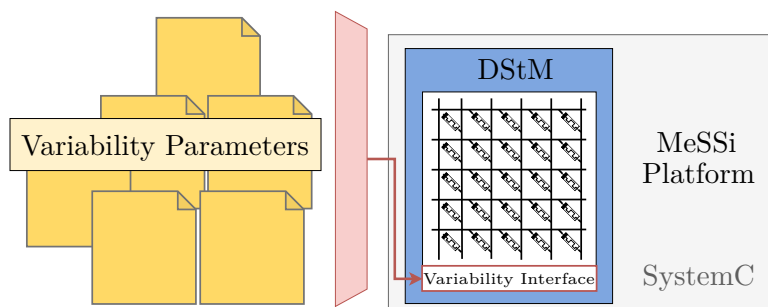


Figure 4.0.1: Overview of the SystemC-based MeSSi platform including the DStM.

While several simulation environments have been proposed for specific memristor-based applications, they are often limited in scope, supporting only fixed architectures or a narrow set of components (see Section 2.5). In contrast, this thesis introduces the Memristor System Simulation (MeSSi) platform: a generic, extensible virtual prototyping framework implemented

in SystemC. MeSSi is designed to facilitate the modeling and analysis of memristor-based subsystems, whether used as memory, in-memory computing units, or for other roles within larger digital system contexts. A schematic overview of MeSSi is depicted in Figure 4.0.1. The framework serves not only as a tool for simulation but also as a methodological testbed to evaluate how device-level memristor nonidealities propagate to system-level performance metrics.

In the following sections, the architecture and implementation of MeSSi are described in detail. Special attention is given to its modular structure, the integration of the DStM memristor model and peripheral components, and the ability of the platform to estimate key system-level properties such as energy consumption and latency. Its design is validated through case studies that assess scalability, modeling accuracy, and energy-latency estimation capabilities, which will be presented in the subsequent chapter.

4.1 Background: SystemC for Virtual Prototyping

System-level simulation has become an indispensable tool for the design, exploration, and validation of modern computing systems, particularly as hardware and software complexity increases. A virtual prototype in this context is a software-based representation of a hardware system or subsystem, enabling fast functional simulation, early software development, and the analysis of extra-functional properties such as energy and timing. Whereas digital twins aim for real-time synchronization with physical systems, virtual prototypes emphasize design exploration and predictive evaluation prior to implementation.

SystemC is a widely adopted modeling language and simulation environment standardized by IEEE [A187], designed to support hardware/software co-design at multiple abstraction levels. Implemented as a C++ library, SystemC enables the seamless integration of high-level functional models, cycle-accurate modules, and transaction-level communication. This flexibility makes it particularly well suited for constructing virtual prototyping platforms that can combine detailed hardware models, behavioral descriptions, and the direct incorporation of algorithmic C++ code. This modeling paradigm supports a top-down exploration methodology, where abstract prototypes can be incrementally refined as device or architectural details become available.

A key feature of SystemC is its support for extra-functional property modeling. Metrics such as energy, power, and latency can be annotated and tracked in parallel with simulated activities [A102, A186]. This capability is critical for evaluating novel memory and compute architectures, where the impact of device- and system-level choices on energy efficiency and performance must be evaluated early in the design process.

In the context of memristor-based architectures, the modular abstraction hierarchy of SystemC enables the consistent integration of device-level models, such as DStM, with architectural simulators. This makes it an ideal foundation for evaluating how device nonidealities affect system-level behavior.

4.2 Architecture of the MeSSi Platform

MeSSi provides a flexible, modular, and extensible environment for the system-level simulation of memristor-based subsystems within realistic system contexts. The platform natively supports both conventional memory operation and analog in-memory computing, allowing researchers to evaluate a broad spectrum of computing paradigms. In addition, other operational modes such as in-memory logic or physical unclonable function (PUF) generation can be easily integrated as required. This architectural flexibility is crucial for investigating how device-level characteristics propagate through different abstraction layers, and for rapidly testing alternative integration strategies without reimplementing core simulation components.

At its core, MeSSi integrates the DStM model to simulate memristor behavior. Therefore, the platform is highly extensible, allowing the simulation of various types of memristors and operating conditions. The platform can further incorporate peripheral circuits and system-level components such as CPUs and accelerators. This modularity enables systematic exploration of diverse architectures and application scenarios within a unified simulation framework. The architecture follows a hierarchical structure, distinguishing between device, circuit, and system abstraction layers. This separation supports top-down system exploration and bottom-up model validation. By encapsulating device models and interfaces in standardized modules, MeSSi enables reproducible comparisons across different memristor technologies and architectures.

Design Principles and Goals

The design of MeSSi follows principles intended not only to ensure software extensibility but also to support a systematic research methodology for exploring the architectural implications of memristive technologies. The key goals guiding this design are:

- **Flexibility:** To accommodate the diversity of memristor devices and circuit topologies, MeSSi allows researchers to interchange models and configurations dynamically, facilitating comparative studies across architectures and operating conditions.
- **Modularity:** By clearly separating components, MeSSi supports incremental development and independent validation of device, circuit, and system layers. This is a prerequisite for reproducible, multi-level design-space exploration.
- **System-Level Integration:** Through standardized transaction-level modeling (TLM) interfaces, MeSSi can be embedded into larger virtual prototypes together with other system components, enabling cross-layer studies that link device behavior to system performance metrics such as latency and energy.

Together, these principles establish MeSSi as a research platform that bridges device-level modeling and system-level evaluation within a coherent, reproducible simulation methodology.

Platform Overview

Figure 4.2.1 provides a block diagram of the MeSSi platform, illustrating the main modules and their interconnections within the SystemC simulation environment.

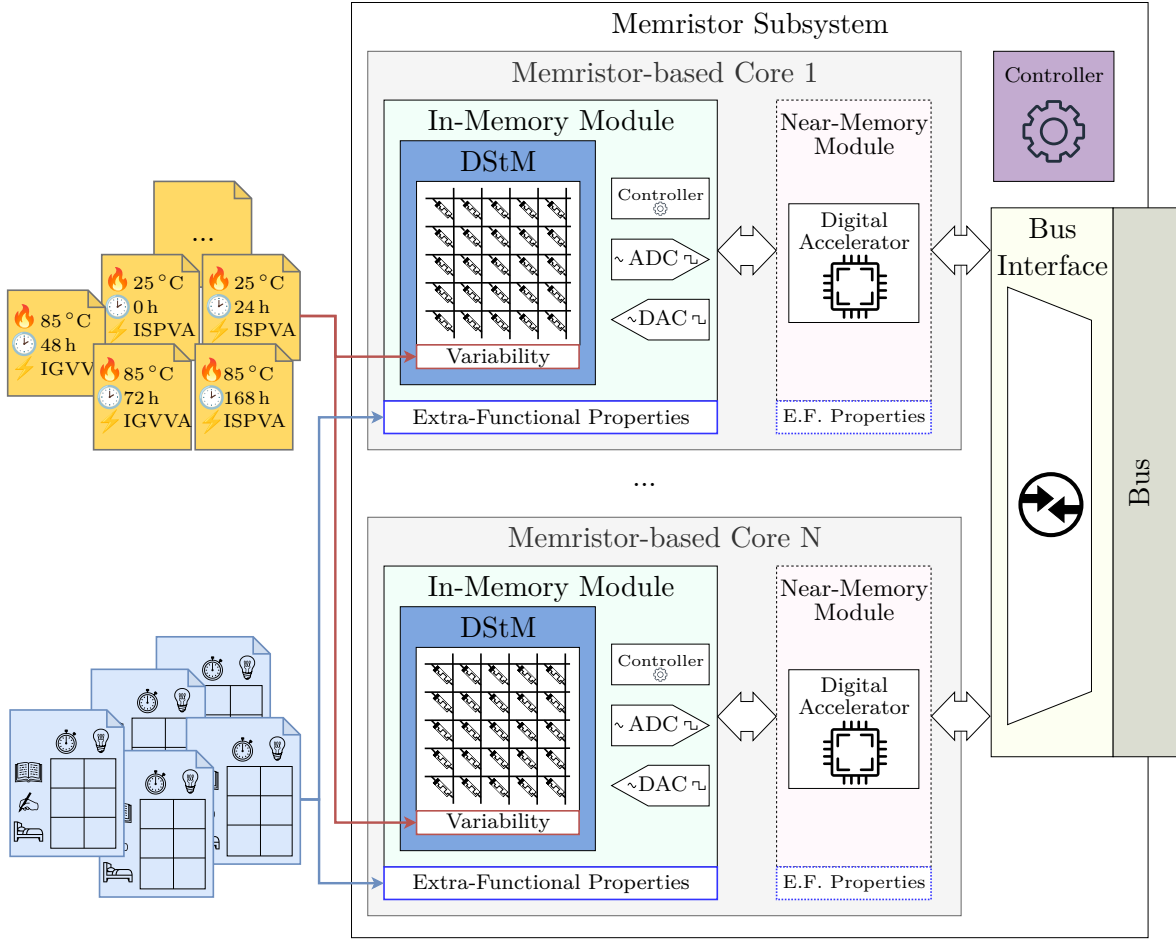


Figure 4.2.1: Block diagram of the MeSSi virtual prototyping platform.

To support systematic exploration of memristor-based architectures across multiple abstraction layers, MeSSi follows a hierarchical modular organization. This structure aligns with the principle of separation of concerns, allowing independent evaluation of device, circuit, and subsystem behavior while maintaining a coherent system-level integration. MeSSi's architecture is organized into three principal modules:

- **In-Memory Module:** This module provides the core abstraction for device-level exploration, enabling consistent comparison of different memristor technologies and peripheral circuit configurations under unified simulation conditions. It forms the foundation of MeSSi and is implemented as a base class that encapsulates the memristor array together with models of peripheral components (such as ADCs, DACs, and device controllers). It defines a unified abstraction through getter methods for extra-functional properties and virtual read/write functions, ensuring interface consistency across application domains. Derived classes implement the concrete behavior for specific application domains introduced in Section 2.2. At the current stage, MeSSi provides implementations for discrete storage elements and analog in-memory computing, while the categories of memristor-based logic and physical unclonable functions remain open

for future extensions. The design is highly parametrizable and extensible, supporting integration of different memristor types via the DStM model introduced in Chapter 3, as well as customized peripheral/control logic and simulation of extra-functional properties for those components.

- **Near-Memory Module (Optional):** The near-memory layer supports architectural-level studies of hybrid analog-digital computing concepts, allowing analysis of trade-offs between computation locality and data movement. This module enables the simulation of additional digital accelerators or support logic (such as error correction or activation functions) located close to the memristor array. It is intended to model near-memory computing concepts, where digital processing is placed adjacent to the memory core for efficiency. Here also extra-functional properties can be included.
- **Memristor Subsystem (Top-Level) Module:** At the subsystem level, MeSSi enables full-system co-simulation and cross-layer analysis, linking device characteristics to system performance metrics such as latency and energy. Multiple in-memory (and optionally near-memory) modules can be grouped into a *core*, and several cores can be combined in a top-level subsystem. The top module provides a TLM interface for communication with the rest of the system simulation, such as CPUs, bus systems, or other accelerators. It also hosts controllers responsible for inter-module coordination and system-level data handling.

Internal and External Communication

- **Internal Communication:** In the current mainline implementation of MeSSi, communication between modules is realized directly with TLM-2.0 interfaces. This design choice simplifies latency modeling and aligns naturally with transaction-level system simulations, which is central to the case studies presented in this thesis. The choice of TLM-2.0 for intra-platform communication also reflects a methodological focus on timing-aware yet computationally efficient simulation, ensuring that system-level studies remain scalable while preserving latency accuracy.

An alternative variant of MeSSi also exists, in which internal communication is implemented using standard SystemC ports and channels. This design is motivated by the possibility of co-simulating with hardware description language (HDL) components, thereby enabling integration of RTL accelerators with memristor models. Although not utilized in the case studies, this functionality highlights the architectural flexibility and forward compatibility of MeSSi.

- **External Communication:** At the subsystem boundaries, MeSSi consistently employs a loosely-timed TLM-2.0 interface, supporting high-level transaction-based interactions with other simulated components. This abstraction facilitates efficient system-level simulation while preserving sufficient detail for accurate architectural exploration.

Customizability and Extensibility

MeSSi is intentionally designed for extensibility. This extensibility is central to MeSSi's role as a research platform, enabling reproducible exploration of new device models and

integration strategies without restructuring the simulation backbone. Depending on the use case, peripheral modules (e.g., custom ADC types) can be added or modified, and specialized controllers for other operation modes such as in-memory logic or physical unclonable function (PUF) generation can be implemented. A dedicated Mapper Module enables user-defined mappings between data values and conductance states, supporting both simple and reliability-aware coding schemes.

4.3 Implementation

The MeSSi platform is implemented as a set of modular SystemC components, each designed to capture the architectural structure and functional behavior of memristor-based subsystems. This modular design enables both reuse and customization, supporting a wide range of application scenarios and facilitating future extensions. The implementation of MeSSi emphasizes reproducibility and controlled abstraction. Each module is implemented as an independently testable SystemC component, reflecting a hierarchical modeling methodology where behavioral, architectural, and device-level phenomena can be varied systematically. This structure enables consistent experimental conditions when comparing alternative device models or system configurations.

InMemoryModule Base Class

At the heart of the platform is the *InMemoryModule* base class, which provides a generic, reusable template for modeling memristor arrays together with their peripheral circuitry. It establishes a methodological bridge between device-level behavior and system-level evaluation. By encapsulating memristor arrays and peripheral circuitry behind a uniform TLM interface, it enables cross-comparison of different memristor models and control strategies within architectural contexts. The class is parametrized by bus width, address width, and memory size, allowing easy adaptation to different system configurations. The array dimensions are derived from the specified bus width, memory size, and operational mode.

In the **mainline implementation**, the *InMemoryModule* exposes a TLM-2.0 interface for read and write transactions. This design enables accurate modeling of memory latency, seamless integration into system-level simulations, and efficient communication with other TLM components such as CPUs, buses, and accelerators. Each TLM transaction triggers the corresponding internal behavior: device programming, data mapping, and access counting, while abstracting the low-level device physics through the DStM model. Extra-functional properties such as access latency, read/write energy, and static power are parametrized through configuration files and are accumulated during simulation. This parameter-driven configuration allows systematic exploration of technology parameters and ensures reproducibility of simulation experiments, as all key assumptions are externalized and version-controlled. Getter methods allow reporting of these values at the end of the simulation.

In addition to this TLM-based implementation, an **alternative variant** of the *InMemoryModule* exists that uses explicit SystemC ports (clock, reset, address, data lines, and handshake signals). This variant is motivated by the possibility of co-simulating MeSSi with RTL accelerators written in VHDL or Verilog, providing cycle-accurate signal-level interfaces. Although not employed in the case studies presented in this thesis, it remains a potential ex-

tension path for future research. The coexistence of both TLM-2.0 and signal-level variants underlines the methodological flexibility of the platform: high-speed, loosely-timed simulations for system-level exploration, and cycle-accurate co-simulation for validation against RTL implementations.

The simplified code structure of the base class is illustrated below, showing its parametrization, encapsulation of peripheral models, bookkeeping of extra-functional properties, and the abstract read/write methods that are specialized by derived classes:

```

1  template<size_t bus_width, size_t addr_width, size_t mem_size>
2  struct InMemoryModule : public sc_core::sc_module
3  {
4  private:
5      ...
6      // Simulation components
7      std::unique_ptr<memrstor<_array_height, _array_width>> mem_array_;
8      adc adc_;
9      dac dac_;
10
11     // Access counter
12     size_t read_count_ = 0u;
13     size_t write_count_ = 0u;
14
15     // Extra-Functional property parameters loaded from parameter file
16     double cfg_read_energy_per_op_;
17     double cfg_write_energy_per_op_;
18     double cfg_static_power_;
19     sc_core::sc_time cfg_read_latency_;
20     sc_core::sc_time cfg_write_latency_;
21
22     // TLM transport
23     void b_transport(tlm_generic_payload& trans, sc_time& delay) {
24         ...
25 public:
26     // TLM target socket
27     tlm_utils::simple_target_socket<InMemoryModule> target_socket;
28     ...
29     // Virtual methods to be implemented by derived classes
30     virtual void read(std::uint64_t addr, std::uint8_t* dst, size_t len) = 0;
31     virtual void write(std::uint64_t addr, const std::uint8_t* src, size_t len
32 ) = 0;
33     ...
34
35     // Extra-Functional properties
36     getStaticEnergy(sc_core::sc_time runTime) const;
37     getWriteEnergy() const;
38     getReadEnergy() const;
39     getReadLatency() const;
40     getWriteLatency() const;
41     ...
42 }

```

Listing 4.1: Definition of the abstract *InMemoryModule* base class.

Peripheral Component Models

Peripheral components such as ADCs (analog-to-digital converters), DACs (digital-to-analog converters), and device controllers are modeled as independent modules within the *InMemoryModule*. These peripheral models are fully parametrizable, enabling researchers to study the impact of different resolutions, quantization schemes, or circuit architectures on system behavior. Controllers can be adapted or replaced depending on the application, whether conventional memory access, analog computation, or custom logic is required.

The Mapper Module

To enable flexible mapping between digital data values and physical device states, MeSSi introduces a dedicated Mapper module. This component allows the user to specify custom mappings (e.g., for Gray coding, reliability-aware, or application-specific encoding) by defining a permutation between data values and conductance states. The Mapper class can be included during read and write operations, thus enhancing reliability or supporting specialized coding strategies without modifying the overall simulation architecture. Methodologically, the Mapper acts as a controlled abstraction layer that isolates the effects of data encoding from those of device physics, enabling systematic comparison of coding schemes under identical hardware and workload conditions.

Table 4.3.1 shows an exemplary mapping of two-bit input values to device conductance states. In this example, the most common bit pattern (00), which frequently appears in quantized neural network weights, is assigned to the highest conductance state ($x = 3$). Since the highest conductance state is often the most stable in many RRAM devices, this mapping minimizes the overall error probability by aligning frequent input patterns with the most reliable device states. Such mappings can be systematically varied within MeSSi to study trade-offs between reliability, coding complexity, and performance, providing a methodological basis for cross-technology evaluation.

Table 4.3.1: Example mapping from 2-bit input to conductance state index x .

Input bits	State index x
00	3
01	0
10	1
11	2

Application-Specific Derived Classes

Derived classes specialize the *InMemoryModule* base class for the different application domains introduced in Section 2.2. Each derived class implements the concrete behavior of the TLM interface including the read and write methods, and the tracking of the extra-functional properties, thereby mapping the abstract in-memory module to a particular use case. At the current stage of MeSSi, two application domains are supported: discrete storage elements and in-memory computing. The domains of memristor-based logic and physical unclonable functions remain open for future extensions and can be realized by deriving additional modules from the same base class. Methodologically, this inheritance structure formalizes a modular

experimentation framework, ensuring that each application domain can be isolated, extended, and benchmarked under consistent simulation and measurement conditions.

Discrete Storage Operation Module This module models conventional memory read and write functionality, device selection, and the mapping between digital data and memristor conductance states. In the simulated array, the width corresponds to the number of devices per row (which can be accessed in parallel to satisfy a bus request), while its height represents the number of rows. All devices are assumed to be organized in a 1T1R configuration, where each memristor is paired with an access transistor. This reflects typical physical implementations, suppresses sneak-path currents and enables selective access through wordline (WL) control.

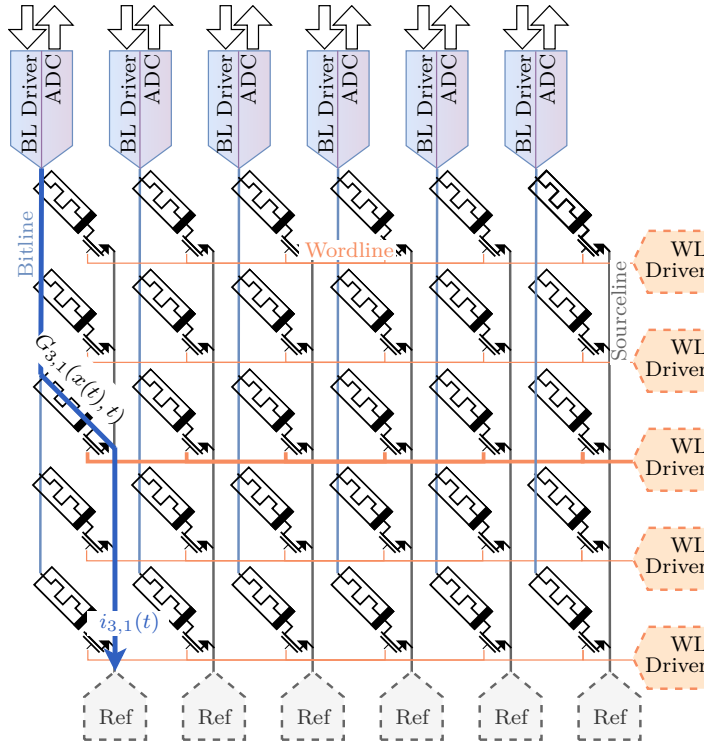


Figure 4.3.1: The principle of discrete storage element operation implemented in MeSSi.

Figure 4.3.1 illustrates the principle of operation as captured in the MeSSi platform. Solid lines denote explicitly simulated components and signal paths, while dashed lines indicate elements represented only at an abstract level. In particular, column ADCs are modeled in detail, since they define the quantization of the readout current and thereby strongly impact functionality. By contrast, peripheral elements such as wordline and bitline (BL) drivers, sourceline biasing, and programming pulse generation are abstracted: device selection is handled directly in the array model, and conductance states are updated by assigning target conductance indices rather than simulating detailed switching waveforms. Additional circuit blocks often present in hardware, such as decoders, sense amplifiers or transimpedance amplifiers (TIAs) preceding the ADCs, and higher-level mode-control logic, are not part of the explicit simulation. However, their impact on latency and energy is included parametrically

through the DStM model and the extra-functional properties models. This selective abstraction constitutes a methodological trade-off: it preserves architectural fidelity where it most affects functionality (e.g., quantization), while reducing complexity in peripheral circuitry that primarily influences performance metrics.

In practice, column ADCs may be dedicated per bitline or shared (e.g., time-multiplexed) across multiple columns to trade off read parallelism against area and energy efficiency. For binary operation, sense amplifiers might replace full ADCs. Both configurations are supported within the abstraction of MeSSi, allowing controlled experiments on the trade-off between read-parallelism and peripheral overhead. For clarity, the example in Fig. 4.3.1 assumes one ADC per column.

The readout behavior of a selected device at position (r, c) is captured by the DStM model as:

$$i_{r,c}(t) = G_{r,c}(x(t), t) \cdot v_{read} \quad (4.1)$$

$$x_{r,c}(t) = \left\lfloor \frac{i_{r,c}(t) - i_{\min}}{\Delta} \right\rfloor, \quad \Delta = \frac{i_{\max} - i_{\min}}{Q_{\text{levels}}} \quad (4.2)$$

where $i_{r,c}(t)$ is the read current of the device, $x_{r,c}(t)$ the corresponding quantization index, and Δ the quantization step size determined by the ADC resolution. The number of quantization levels Q_{levels} is set according to the number of distinct states that must be distinguished: for discrete storage, this matches the number of targeted device conductance states, whereas in in-memory computing scenarios it is typically larger, since accumulated currents from multiple devices must be digitized with higher precision. This compact formulation links the continuous device behavior to the discrete digital values observed at the memory interface, thereby bridging physical operation and system-level abstraction. From a methodological perspective, this formalization provides a reproducible translation between analog device physics and digital system metrics, which is central to the validation of mixed-domain models in MeSSi.

Analog In-Memory Computing Module This module models matrix vector multiplication (MVM) in a memristor-based crossbar array. Each column performs one multiply accumulate (MAC) operation in parallel. Input voltages v_r are applied to the rows and are weighted by the device conductances $G_{r,c}$ according to Ohm's Law. The resulting currents are summed along the sourcelines in accordance with Kirchhoff's Law, producing a column current I_c . In this way, the conductance matrix acts as the weight matrix, the applied voltages act as the input vector, and the accumulated column currents provide the analog dot products. The array dimensions then define the computational structure: the number of columns corresponds to the number of MAC operations carried out in parallel, while the number of rows corresponds to the number of multiplicative terms contributing to each MAC. From a methodological standpoint, this formulation provides a controlled and scalable representation of analog computation within the framework, enabling quantitative comparison of array-level behavior under different device, circuit, and numerical precision assumptions.

As in the discrete-storage operation module, all devices are assumed to be organized in a 1T1R configuration, which suppresses unwanted sneak-path currents and allows selective

activation of columns through their wordlines. Passive arrays are not modeled by default, since their additional sneak paths would require explicit treatment.

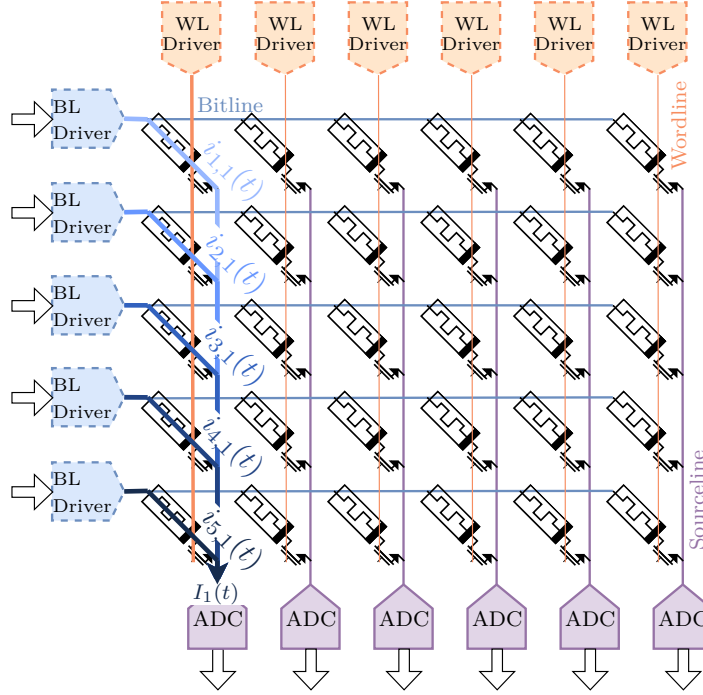


Figure 4.3.2: The principle of in-memory matrix-vector multiplication using analog MAC operations implemented in MeSSI.

Figure 4.3.2 follows the same convention as the discrete storage module, where solid lines denote explicitly simulated paths and blocks, while dashed lines indicate abstracted periphery. Column-wise accumulation and column ADCs are modeled explicitly, as they set the effective numerical precision of the MVM. Peripheral circuits such as wordline and bitline drivers, current-to-voltage converters (e.g., TIAs), decoders, and mode-control logic are abstracted, with their latency and energy contributions incorporated parametrically through the DStM and extra-functional models. ADCs may be dedicated per column or time-multiplexed across multiple columns, providing an experimental control knob for exploring trade-offs between parallelism, area, and energy efficiency. The abstraction also supports differential (two-column) encoding for signed weights, implemented via either analog or digital subtraction.

The DStM-based compute step for an input vector $v_r(t)$ applied to the rows and a conductance matrix $G_{r,c}(\cdot)$ is

$$i_{r,c}(t) = G_{r,c}(x(t), t) \cdot v_r(t) \quad (4.3)$$

$$I_c(t) = \sum_r i_{r,c}(t) \quad (4.4)$$

$$k_c(t) = \left\lceil \frac{I_c(t) - I_{\min}}{\Delta} \right\rceil, \quad \Delta = \frac{I_{\max} - I_{\min}}{Q_{\text{levels}}} \quad (4.5)$$

Here, $i_{r,c}(t)$ is the current through the memristor device at row r and column c . $I_c(t)$ is the accumulated column current corresponding to the result of one MAC operation, and

$k_c(t)$ is its quantized digital output. Since many device currents are summed per column, the required ADC resolution Q_{levels} is typically higher than in discrete storage mode. This compact formulation connects the continuous device behavior with the discrete digital values at the memory interface. Methodologically, this formulation defines a reproducible mapping between analog computation and digital observables, enabling systematic evaluation of accuracy-energy trade-offs and the effects of device nonidealities on algorithmic performance.

4.4 Modeling Extra-Functional Properties

Beyond functional correctness, evaluating the energy and latency characteristics of novel memory and compute subsystems is essential for system-level design and architectural exploration. MeSSi is designed to support the systematic estimation of these extra-functional properties in a modular and configurable way, enabling detailed analysis and fair comparison of diverse architectures. Methodologically, this framework operationalizes extra-functional evaluation as a parametrized layer orthogonal to functional modeling, allowing systematic variation of technology, architecture, and workload parameters under consistent experimental conditions.

Energy Estimation

Energy consumption within MeSSi is estimated using an activity-based methodology, which separates static and dynamic energy components. This abstraction aligns with established system-level modeling practice, enabling scalable estimation without requiring detailed electrical simulation at every timestep. For each simulation, pre-characterized energy parameters, such as energy per read (or MAC) or write operation and static power values, are loaded from parameter files at startup. These parameters can originate from physical measurements, SPICE simulations of circuit implementations, or analytical tools such as NVSim [A40], depending on the desired balance between accuracy and abstraction.

During simulation, dynamic activity statistics (e.g., number of read and write events) are recorded by each module. At the end of the simulation, total dynamic and static energy consumption is calculated as:

$$E_{\text{dyn}} = N_{\text{read}} \cdot E_{\text{read}} + N_{\text{write}} \cdot E_{\text{write}} \quad (4.6)$$

$$E_{\text{static}} = P_{\text{static}} \cdot T_{\text{active}} \quad (4.7)$$

Where N_{read} and N_{write} denote the number of read and write operations, E_{read} and E_{write} the respective energy per access, P_{static} the static power, and T_{active} the total active simulation time of the module. This formulation ensures that all energy results are reproducible and comparable across simulation runs, as they depend solely on recorded activity metrics and externally defined parameters rather than hidden simulator states.

The level of detail for energy estimation is configurable:

- For rapid design space exploration, a coarse-grained mode can track and accumulate only the total dynamic and static energy on sub-system level.
- For fine-grained analysis, energy can be grouped by access type and array, depending on simulation requirements and available characterization data.

This configurability allows researchers to tailor the estimation fidelity to the purpose of the study: from early architectural exploration to detailed calibration against physical measurements.

Static energy is determined by multiplying the static power of each module by its active runtime, which is tracked globally and provided to each module at simulation end. This approach is robust even in loosely-timed TLM simulations, where local time advances may not always reflect global simulation time due to temporal decoupling. This approach therefore maintains methodological consistency even under different TLM timing modes, ensuring that energy accounting remains independent of the simulation’s temporal abstraction level.

It should be noted that the internal energy required for memristor access can be expressed as:

$$E_{mem} = \int_0^{t_{acc}} v(\tau)^2 \cdot G(x(\tau), \tau) d\tau \quad (4.8)$$

where $v(\tau)$ represents the voltage pulse train for either read or write access, $G(x(\tau), \tau)$ is the device conductance, and t_{acc} is the duration of the voltage pulse train. While MeSSi could in principle evaluate this integral directly, in practice the intrinsic memristor energy is typically much smaller than the energy consumed by the peripheral circuitry. In the present work, the energy per access parameter values already include both the intrinsic device energy and the dominant peripheral circuit overheads. This choice reflects a methodological trade-off between physical fidelity and model usability, ensuring that energy estimation remains both interpretable and efficient at system scale.

Latency Estimation

In the main *TLM-based implementation* of MeSSi, latency is modeled at the transaction level. Following the standard loosely-timed TLM style, transactions are implemented as blocking transports. This means that, for each transaction, the `b_transport` method is invoked with a payload containing the address, command, and data fields, as well as a delay parameter. The delay parameter is used to accumulate the estimated latency of the corresponding operation. After the target has processed the transaction, the initiator can advance the simulation time by the final value of this delay. This approach provides a consistent, transaction-level abstraction of latency, ensuring that timing effects can be modeled and varied systematically across devices and architectures without requiring signal-level detail.

For each memory or compute transaction (read, write, or matrix-vector multiplication), the latency contribution is computed and added to the delay parameter. The total delay depends on several factors:

- the type of operation being simulated (read, write, or in-memory MVM)
- the number of devices affected by the operation (e.g., single rows for read transactions or multiple columns for MVMs)
- the degree of parallelism in the architecture (e.g., the number of ADCs available in parallel)
- the base latency of a single memristor access (read or write)

By expressing these dependencies parametrically, MeSSi decouples latency modeling from specific device implementations, allowing researchers to study architectural scaling effects and technology sensitivity within a unified simulation framework.

The base latency parameters are read from an external configuration file at the beginning of the simulation. This enables easy adaptation to new device technologies or programming schemes without changes to the simulation code. This parameter-driven configuration embodies a methodological principle of MeSSi: simulation experiments remain fully reproducible and adaptable by altering configuration data rather than source code.

The calculation of the latency per transaction is handled within the respective operation modules (e.g., discrete storage or analog in-memory computing). Within each operation module, latency estimation follows an analytical formulation that scales with data size and architectural resources. For example, the read latency in a discrete storage array can be expressed as:

$$L_{\text{read}} = \left\lceil \frac{D_{\text{read}}}{D_{\text{row}}} \right\rceil \cdot \left\lceil \frac{W_{\text{array}}}{N_{\text{ADC}}} \right\rceil \cdot L_{\text{cfg,read}} \quad (4.9)$$

where

$$D_{\text{row}} = b_{\text{cell}} \cdot W_{\text{array}}$$

Here, D_{read} denotes the total amount of data to be read (in bits), b_{cell} is the number of bits stored per memristor cell, W_{array} is the array width (number of columns), and N_{ADC} is the number of available ADCs. The term $L_{\text{cfg,read}}$ is the configured base latency for a single memristor read access of b_{cell} bits.

The first ceiling term, $\left\lceil \frac{D_{\text{read}}}{D_{\text{row}}} \right\rceil$, accounts for the number of row activations required to read the requested data. The second term, $\left\lceil \frac{W_{\text{array}}}{N_{\text{ADC}}} \right\rceil$, captures the number of sequential column read groups that arise due to limited ADC parallelism. Consequently, the overall read latency increases when fewer ADCs are available and approaches the single-row access latency when full column parallelism ($N_{\text{ADC}} \geq W_{\text{array}}$) is achieved. This formulation provides a scalable analytical estimate that reflects both data volume and architectural parallelism within the array.

For the in-memory matrix-vector multiplication (MVM), the latency is estimated as:

$$L_{\text{MVM}} = \left\lceil \frac{W_{\text{array}}}{N_{\text{ADC}}} \right\rceil \cdot L_{\text{cfg,read}} \quad (4.10)$$

where W_{array} is the number of columns in the array and N_{ADC} is the number of available ADCs. In the MVM module, the term $L_{\text{cfg,read}}$ represents the effective latency of one full MAC operation for a single ADC time slot (i.e., one column-group read). It includes the application of the input vector as voltage pulses across all rows, the transient and current accumulation along the selected columns, and the conversion of the resulting column currents into the digital domain by the ADCs.

The factor $\left\lceil \frac{W_{\text{array}}}{N_{\text{ADC}}} \right\rceil$ captures the required number of sequential column groups due to limited ADC parallelism. When $N_{\text{ADC}} \geq W_{\text{array}}$, the MVM completes in a single time slot. This formulation abstracts device-level timing details into a consistent operation-level latency metric, enabling fair comparison of analog and digital execution paths within the platform.

For the *alternative implementation* that exposes signal-level SystemC ports, latency is realized with `wait()` calls and cycle counters inside SystemC processes (e.g., per access: `wait(Lread)` or `wait(Lwrite)`). Calling `wait()` suspends the process for the specified simulated time, thereby advancing the time inside the SystemC kernel. This mechanism enables cycle-accurate simulation and allows for HDL co-simulation, but was not employed in the case studies presented here. Both variants share the same parameter sources, but they differ in how latency is applied: the TLM-based implementation accumulates delays in the delay parameter of the transaction, whereas the signal-level implementation advances time explicitly. Together, the two timing models provide complementary methodological perspectives: the TLM-based approach supports high-speed architectural exploration, while the signal-level model enables validation against cycle-accurate reference implementations.

Notes on Area and Other Metrics

While energy and latency estimation are supported natively within MeSSi, area modeling is not currently implemented due to its strong dependence on the underlying technology and peripheral circuitry. However, area parameters could be incorporated in future work using the same annotation and accumulation techniques used for energy and latency. This planned extension would complete the methodological triad of key system metrics allowing unified trade-off analysis within the same simulation environment.

4.5 Summary of the MeSSi Platform

In this chapter, the MeSSi virtual prototyping platform for memristor-based systems was presented. In contrast to many existing system-level frameworks, which are often tailored to specific application domains or fixed architectural templates (see Section 2.5), MeSSi is designed as a *generic, extensible, and modular* environment that supports a wide range of operational modes, device technologies, and architectural configurations. Its architecture enables the simulation of both conventional memory operation and analog in-memory computing, while also enabling the straightforward integration of alternative operation modes, custom controllers, and new peripheral models. Methodologically, this design realizes a unified experimental framework in which architectural and device-level effects can be explored independently or jointly under consistent simulation assumptions.

A distinguishing strength of MeSSi lies in its ability to abstract the modeling of memristor devices and their peripheral circuitry in a way that remains interoperable with other simulated system components, from CPUs to accelerators. This interoperability is central to the methodological goal of bridging device-level modeling with system-level performance analysis. The platform supports realistic device modeling through multi-level storage capability, variability derived from measured data, and flexible exchange of device types and operating conditions for thorough design space exploration. Its modular structure ensures that components can be replaced, extended, or scaled, from single devices to complex multi-array subsystems,

without affecting the overall integration. This modularity provides a methodological guarantee that changes at one abstraction level (e.g., device or circuit) can be isolated, and their impact can be quantified at higher architectural levels. Furthermore, MeSSi integrates configurable data-to-conductance mapping strategies and built-in estimation of extra-functional properties, such as latency and energy, thus supporting comprehensive, reproducible system-level evaluation of both functional and performance aspects.

Collectively, these features, summarized in Table 4.5.1, establish MeSSi as a versatile and extensible platform for research on memristor-based architectures. The following chapter demonstrates its capabilities across diverse case studies.

Table 4.5.1: Key features of the MeSSi platform.

	Feature	Description
	Multi-Level Storage	Models multiple conductance states per device, including peripheral circuitry, enabling realistic multi-bit-per-cell evaluations.
	Variability	Captures both time-invariant and time-dependent variability, allowing selective assessment of variability sources based on measurement data.
	Devices & Operating Conditions	Supports rapid substitution of device models and operating conditions for broad design-space exploration.
	Modularity	Comprises clearly separated, interchangeable modules with configurable parameters for architecture-specific adaptation.
	Scalability	Scales seamlessly from single devices to large arrays and multi-array subsystems.
	Interoperability	Integrates with other simulated components via standardized SystemC/TLM interfaces.
	Mapping Strategies	Provides flexible, easily swappable data-to-conductance mapping schemes.
	Extra-Functional Properties	Estimates latency and energy consumption for individual components and the overall subsystem.
	Co-Simulation	Enables joint simulation with C++ algorithmic or HDL-based models for mixed-abstraction system studies.

Chapter 5

Case Studies using the Virtual Prototyping Framework

To illustrate the practical applicability of the MeSSi framework, this chapter presents five case studies, each corresponding to published work by the author of the thesis at hand. The case studies span a broad spectrum of application domains, ranging from memristor-based analog in-memory computing to processor architectures incorporating non-volatile memristor-based memory, and address diverse research questions such as energy efficiency, computational accuracy, and resilience to environmental stress.

The central purpose of these case studies is not only to highlight the potential of memristive devices in different contexts, but more importantly to demonstrate how the proposed modeling and simulation framework enables systematic exploration of these questions.

The DStM and the MeSSi framework were developed iteratively over the duration of these works. While earlier studies relied on focused, problem-specific virtual prototypes of individual components, each contributed critical insights and building blocks that were later unified into the versatile MeSSi platform in its final form. Thus, the case studies can be understood as milestones along the trajectory from isolated models to an integrated, reusable framework.

For each case study, the problem context, research objective, and the role of memristor modeling are outlined. Special attention is given to the aspects of memristor behavior that were explicitly modeled and proved critical for addressing the respective research question. Each section concludes with a summary table that highlights the relevant modeled features, such as variability, multi-level operation, operating condition effects, extra-functional property estimation, enabling a quick comparison across all use cases.

5.1 Ultra-Low-Power ECG Classification Architecture for Wearable Devices

Motivation and Problem Statement

Continuous monitoring of cardiac activity in daily life enables the early detection of cardiac anomalies, which can support timely diagnosis and intervention. Wearable devices equipped with appropriate sensing and processing capabilities can identify transient or intermittent conditions that might otherwise remain undetected during sporadic clinical examinations. The ability to process physiological signals directly on-device also reduces reliance on wireless transmission, thereby preserving privacy and reducing energy consumption.

However, continuous monitoring of cardiac rhythm in wearable devices is constrained by stringent power budgets to enable long battery lifetimes. Atrial fibrillation (AFib), a common arrhythmia characterized by irregular atrial activation and ventricular response, is typically detected in electrocardiogram (ECG) time series through analysis of rhythm and waveform morphology [A188]. Neural networks (NNs) are particularly well suited for this classification task, as they can learn complex, non-linear relationships in the ECG signal that may be difficult to capture with handcrafted features or simple statistical methods [A189]. When trained on sufficiently large and diverse datasets, they have the potential to achieve high diagnostic accuracy and generalize well to varying patient populations and recording conditions.

In embedded, low-duty-cycle scenarios, ECG acquisition is typically performed at comparatively low sampling rates, as the fundamental frequency content of the cardiac signal is limited by physiological constraints [A190]. Heart rates remain well below a few hundred hertz, and according to the sampling theorem, a rate in the order of a few hundred samples per second is sufficient to capture the relevant waveform morphology with high fidelity [A191]. In such systems, evaluation only needs to be triggered intermittently when a complete analysis window is available. Under these conditions, static and leakage power can dominate if processing elements remain active continuously, and data movement to and from external memory may exceed the energy cost of the computations themselves. Control and buffering overheads further increase the total energy per classification window.

The problem addressed in this case study can therefore be formulated as follows: Given an ECG time-series classification task for AFib detection in a wearable device using a neural network, how can the overall embedded system be organized so that the total energy per classification window is minimized without compromising classification accuracy?

A promising direction to address this challenges lies in a digital dataflow driven computer architecture that integrates on-chip, non-volatile memristive storage for NN parameters. Such an approach offers the potential to eliminate costly external memory accesses, introduce aggressive duty cycling enabled by the state retention of memristors, and align discrete conductance states with quantized NN parameters, thereby reducing both array size and energy needed for read-out. The subsequent subsections detail how these principles were applied with the help of the modeling approaches introduced in this thesis.

This use case was first presented in [B4], which describes the framework used to design an energy-efficient AI accelerator architecture that earned first prize in the German Federal Ministry of Education and Research (BMBF) *Energy-efficient AI Systems* innovation contest

in 2021 [A192]. The proposed architecture and design methodology targeted ultra-low-power ECG classification in wearable devices. The main architecture concept was later generalized in [C1] to a broader application domain. At the time of the initial work, many of the capabilities later integrated into the MeSSi framework were not yet present. However, key elements were already established: multi-level storage simulation using concrete IHP RRAM devices, scalability exploration via an abstract model of inference with RRAM-based weight storage, and the extraction of extra-functional properties from a concrete analog memory block implementation [B3]. These early evaluations provided the parameter values for latency and energy estimation that subsequently formed the basis of the extra-functional property models in MeSSi.

Application and Data Characteristics

The ECG signal is a low-frequency biomedical waveform whose fundamental frequency components correspond to the heart rate and its harmonics. Even in pathological conditions, heart rates remain well below a few hundred hertz. According to the sampling theorem, accurate reconstruction and feature preservation are possible with sampling rates in the order of a few hundred samples per second. In the present application, a sampling rate of 512 Hz was selected based on established clinical practice [A191], representing the lowest rate that avoids distortion of diagnostically relevant waveform components. Using the lowest feasible sampling frequency reduces the amount of acquired data and lowers the energy consumed in both acquisition and downstream processing.

To perform classification, a sufficient temporal context of the ECG signal must be available to capture rhythm irregularities and waveform features indicative of atrial fibrillation. Isolated samples do not provide enough information for a reliable decision. Consequently, data is buffered into fixed-length segments (analysis windows), each containing the required duration of signal for classification. This approach enables inference to be executed intermittently, only after a complete window has been acquired, thereby reducing the active time of the processing core.

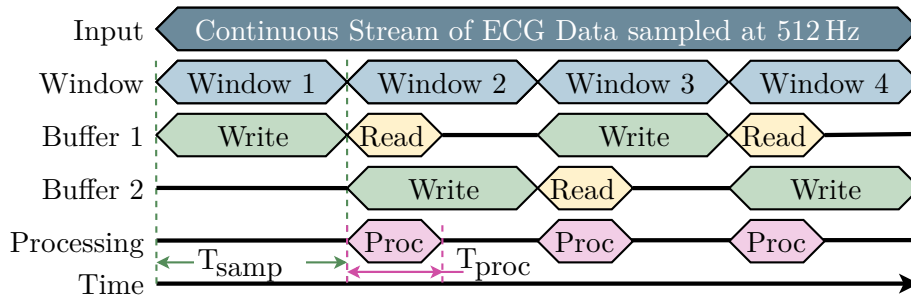


Figure 5.1.1: Sampling and processing scheme of the continuous ECG signal. Adapted from [B4].

The resulting sampling and processing scheme is illustrated in Figure 5.1.1. Data acquisition proceeds continuously at the selected sampling rate (with a sampling time of T_{samp}), while classification is triggered periodically upon the completion of an analysis window. During window acquisition, incoming samples are written into a buffer. Once a window is filled, its content is read and processed in a short burst (with a processing duration of T_{proc}), while

the buffer alternates to capture the next segment. This double-buffering arrangement ensures continuous data capture without gaps, while limiting high-frequency activity in the processing core to discrete, infrequent intervals aligned with the classification schedule.

Processing Architecture and Dataflow

The classification workload is implemented as a *1D convolutional neural network* (1D-CNN), which processes the ECG signal in the time domain. Convolutional layers are well suited for this task, as they can extract temporal features such as waveform morphology and rhythm patterns that are relevant for atrial fibrillation detection. The specific network topology was determined through a *neural architecture search* (NAS), which optimized the number of layers, filter sizes, and stride configurations for high classification accuracy under strict energy constraints. All weights were constrained to ternary states ($-1, 0, +1$), which matches the amount of states, stored in a single RRAM device. The resulting NN architecture comprises multiple convolutional layers followed by fully connected layers, with a low amount of parameters to enable efficient on-chip storage.

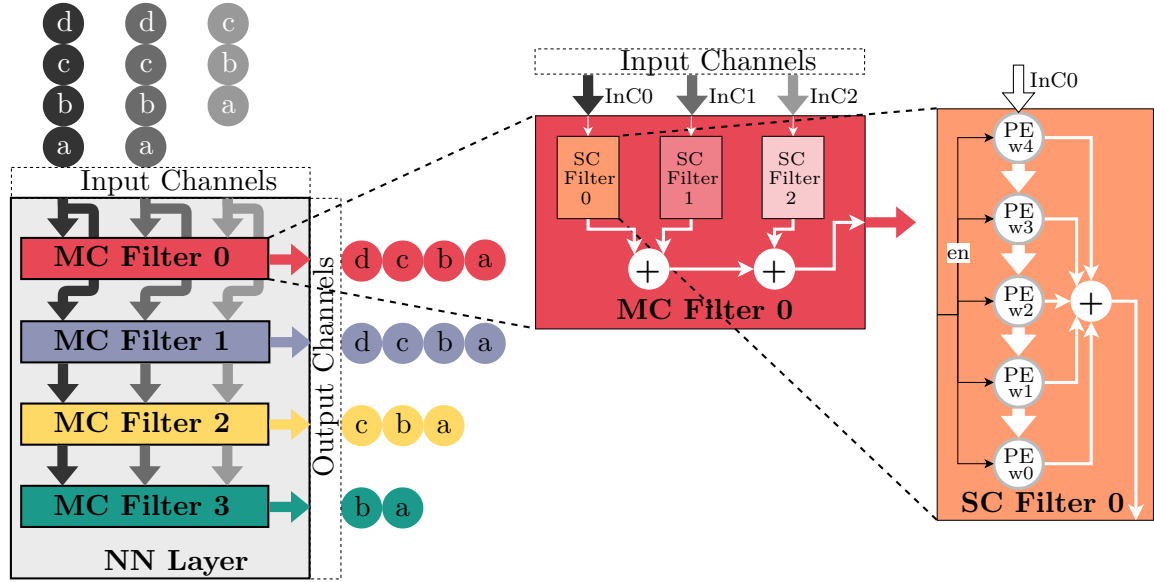


Figure 5.1.2: Hierarchical structure of the digital accelerator core for convolutional layer inference. Adapted from [B4].

A dedicated digital accelerator was designed to execute the convolutional layers with minimal energy and latency overhead. The compute fabric employs a *systolic dataflow* that is highly specialized to the fixed NN topology. As illustrated in Figure 5.1.2, three hierarchical processing modules are used:

- **Shift-based filter processing units (SC Filters):** Each unit implements a convolution filter kernel, including potential strides in the kernel movement. The design shifts input data through processing elements (PEs) with local weights, enabling efficient reuse of inputs without additional buffering.

- **Multi-channel filter modules (MC Filters):** Each module processes multiple input channels by instantiating and combining several SC Filters. The partial sums from each SC Filter are accumulated within the MC Filter to produce a single output channel.
- **Neural network layer modules:** Multiple MC Filters are instantiated in parallel to produce all output feature channels of a layer.

In the implemented systolic dataflow, input data is passed directly from one processing element to the next across all modules, thereby minimizing data movement and avoiding repeated memory accesses. This dataflow is depicted as bubbles, where each bubble corresponds to one value. Within a layer, data from the input channels flows sequentially through the MC Filters, except the first two MC Filters, which receive the data simultaneously. This arrangement produces the first two output channels in the same clock cycle, while each subsequent output channel is generated with a delay of one cycle. The scheduling inherently accounts for the internal accumulation delay in the MC Filters, eliminating the need for additional control logic or intermediate buffering. The resulting fixed and predictable timing reduces both design complexity and the energy overhead associated with programmable control paths.

This convolutional accelerator possesses a **weight-stationary dataflow**, in which neural network parameters remain resident in local storage within each PE throughout computation [A193]. Only input feature data and intermediate results are propagated between PEs. This strategy minimizes memory traffic for parameter access, a major contributor to energy consumption in compact networks with high weight reuse. By co-locating parameters with arithmetic units, large-scale buffer structures and complex control logic are avoided, reducing both static and dynamic power.

The accelerator is tightly coupled with embedded non-volatile RRAM blocks that store the NN parameters. Parameters are loaded in parallel into latch banks upon wake-up, reducing effective access latency to a single cycle. The combination of the specialized systolic dataflow, weight-stationary parameter placement, and parallel RRAM read-out significantly shortens the active processing time, enabling aggressive duty cycling of the Processing Core and substantial energy savings.

System Design

The designed system employs a dual power-domain architecture to minimize energy consumption in low-duty-cycle operation. This concept is illustrated generically in Figure 5.1.3. A small always-on domain handles continuous sensor acquisition, initial pre-processing, and buffering of the input data. The main processing domain, containing the non-volatile neural network accelerator (NV processing), remains completely powered down during idle phases and is only activated when a complete analysis window is available. This duty-cycling strategy eliminates static power consumption in the high-performance processing blocks for the majority of the operating time.

The specific system implementation for the ultra-low-power ECG classification use case is shown in Figure 5.1.4. The always-on **Data Control Core** integrates the pre-processing pipeline, a double-buffered SRAM for temporary storage of analysis windows, and an RRAM manager for interfacing with the non-volatile parameter storage. The **Processing Core** contains the RRAM blocks and the digital accelerator for the CNN implementing the fixed

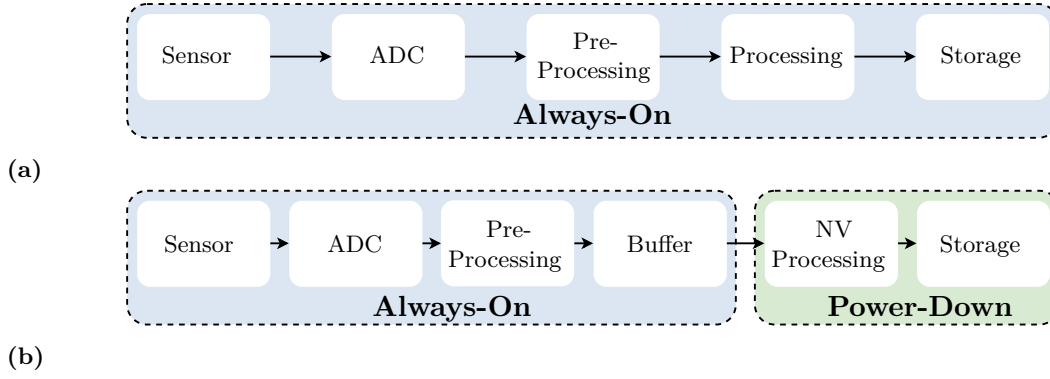


Figure 5.1.3: Comparison of different power-domain designs for low-duty-cycle signal classification in embedded systems. a) Block diagram of an architecture with a single power-domain. b) Block diagram of the dual power-domain architecture employed for this case study. Adapted from [C1].

systolic dataflow. Power gating between the two domains is implemented with dedicated switches, allowing the Processing Core to be fully isolated during idle periods. The double-buffering in the Data Control Core enables overlap between acquisition and processing, ensuring continuous operation without data loss.

Embedded RRAM Memory Blocks

The neural network parameters are stored in embedded resistive random-access memory (RRAM) blocks integrated directly within the Processing Core. Each block follows a 1T1R cell organization in the IHP SG13S 130 nm process [A194], contains an array of 32 cells per word line and supports multi-level storage for ternary parameter encoding. The peripheral circuitry comprises a voltage pulse generation unit for programming, precise reference generation for sensing, and a multi-way analog sense amplifier array. This enables parallel read-out of all parameters associated with a given processing element, allowing complete parameter loading into latch banks. This parallelism significantly reduces effective memory access latency and contributes to the short active time of the Processing Core. The design of the RRAM blocks, including the analog sense circuitry and the programming path, is described in detail in [B3], which also provides the basis for the extra-functional property parameters (latency, energy per operation) used in the present system-level evaluation.

The integration of these RRAM blocks with the fixed systolic accelerator enables a fully weight-stationary execution model. Once parameters are loaded after wake-up, all convolution and fully connected operations are executed without further accesses to the non-volatile memory. This minimizes data movement, reduces dynamic energy, and eliminates the need for large on-chip buffers or complex memory hierarchies.

Results

The proposed architecture for ultra-low-power ECG classification was evaluated using post-layout simulations of the complete system.

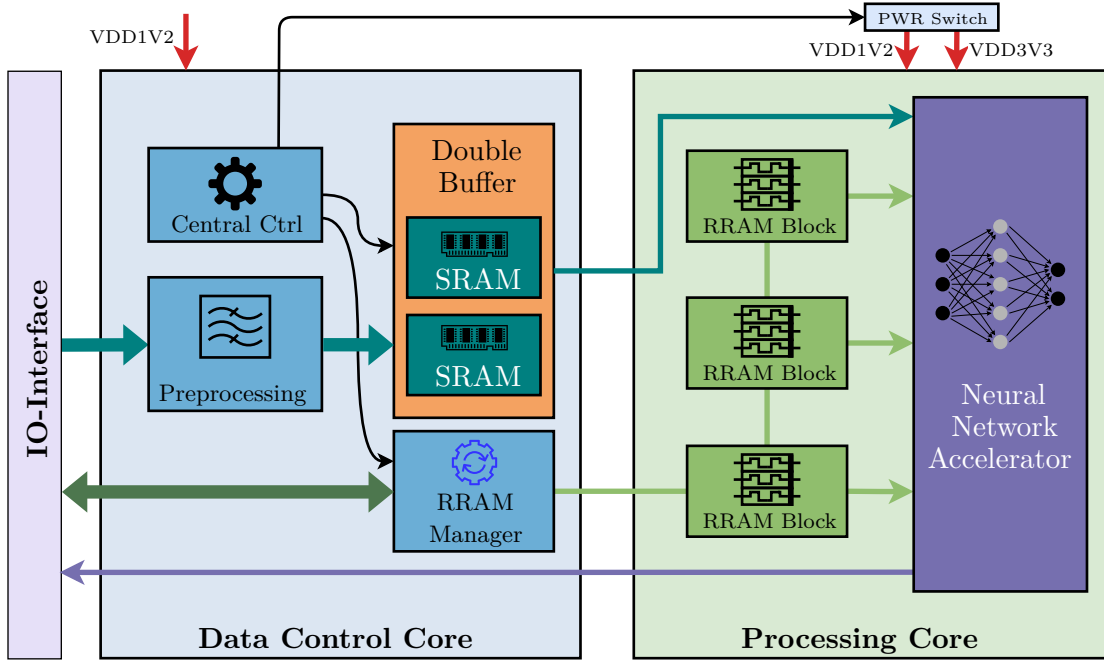


Figure 5.1.4: System-level block diagram of the ECG classification computer architecture. Adapted from [B4].

The classification accuracy was evaluated using a system-level testbench of the complete architecture, operating with recorded ECG data as input to the NN. The same quantized network parameters used in hardware were applied, ensuring that the measured performance reflects the implemented design. Sensitivity, defined as the proportion of correctly identified atrial fibrillation episodes among all actual AFib cases, reached 97 %. Specificity, defined as the proportion of correctly identified normal sinus rhythm segments among all non-AFib cases, was 86 %. These values meet the predefined thresholds of $\geq 95\%$ sensitivity and $\geq 85\%$ specificity for the target application and show that a clinically relevant level is achieved.

Energy and power consumption were analyzed for the complete system, including the Processing Core, Data Control Core, RRAM blocks with analog front-ends, and associated control logic. For a classification window length of 12.65 s, the following results were obtained:

Table 5.1.1: Energy consumption per classification window with and without power-down. Adapted from [B4].

Component	Energy [mJ]		Reduction [%]
	Always-On	Power-Down	
Processing Core	220.50	0.55	99.8
RRAM blocks (analog)	1.24	0.055	95.6
Data Control Core	11.75	11.75	—
Total System	233.49	12.36	94.7

These results correspond to an **average power** of approximately 0.98 mW over the 12.65 s classification cycle when power-down is enabled. The substantial reduction in total energy

is achieved by exploiting the non-volatility of the on-chip RRAM: after each wake-up, all network parameters are restored in parallel from the local memory blocks into integrated latches with negligible dynamic energy and latency overhead.

The breakdown in Table 5.1.1 illustrates that the *Processing Core* is the dominant contributor to energy savings under power-down, while the *Data Control Core* remains the primary static load in idle periods. The analog RRAM front-end also benefits from power gating, as the read and sense circuits can be fully shut down when latched data is available to the digital PEs.

From an architectural perspective, the combination of weight-stationary dataflow and embedded non-volatile memory addresses all three of the primary energy bottlenecks identified during this case study:

1. **Static power during idle periods** is minimized through complete power-down of the Processing Core.
2. **Parameter reload energy** is reduced to $<0.5\%$ of the total cycle energy by parallel RRAM read-out into latches.
3. **Control/buffering overhead** is removed by fixed dataflow specialization to the target NN topology.

The results confirm that device-aware modeling of extra-functional properties, together with co-design of dataflow and memory organization, enables order-of-magnitude improvements in total system energy for low-duty-cycle biomedical signal classification in wearables.

Role of the Virtual Prototyping Platform

The case study implemented key principles later formalized in the MeSSi framework, particularly for simulating memristor-based digital accelerators with locally integrated non-volatile memory. Neural network inference with ternary weights stored in RRAM devices was modelled using an abstract, Python-based simulation environment. This allowed bit-accurate evaluation of the digital intermediate results and supported the neural architecture search in selecting a network configuration with an optimal trade-off between accuracy and energy. The abstraction enabled the efficient simulation of hundreds of thousands of inference operations, providing a rapid hardware-software co-design loop without the overhead of detailed device-level simulation.

Variability was not explicitly included in the main simulation. However, a subsequent evaluation [B2] applied deterministic operating margins on top of the abstract model to account for known device-level variations and cycle-to-cycle fluctuations observed in measurements of the fabricated RRAM blocks. This study demonstrated that, when ensuring sufficient sensing margin and selecting appropriate comparator reference voltages and pulse amplitudes, the probability of bit errors caused by incorrect state classification in the presented RRAM memory was negligible. Although no variability-aware data-to-conductance mapping strategies were explored at the time, since all states being assumed equally stable, this evaluation directly inspired the later development of the DStM for precise system-level variability modeling.

A key contribution of this case study to the later MeSSi framework was the extraction of *extra-functional property parameters* from a concrete hardware implementation. Read and write access latencies, per-operation energy, and static power in different operating modes were derived directly from the circuit-level design and SPICE simulations of the RRAM memory block presented in [B3]. This block comprised 1T1R RRAM cells in the IHP SG13S 130 nm process, together with peripheral circuitry for multi-level programming, read-out, and reference generation. The SPICE-level evaluation included transient simulations of the sensing and programming phases and post-layout parasitic extraction to capture realistic signal propagation delays and power dissipation. The resulting parameter set covered the analog block power consumption in both active and idle states, the energy of the digital latch banks, and the overhead of control logic. Table 5.1.2 summarizes the features of the simulation platform relevant for this case study.

These parameters enabled the creation of an architectural energy model that was used to explore the impact of *duty cycling* in a dual power-domain system. Static and dynamic energy contributions were quantified for different idle interval lengths, allowing the optimal power-down duration to be identified. The results confirmed that the energy required to reload all parameters from RRAM after each wake-up was negligible compared to the savings from powering down the Processing Core between classifications. The methodologies for extracting and integrating these extra-functional property parameters were later incorporated into the MeSSi framework as part of its comprehensive extra-functional property evaluation capabilities.

Table 5.1.2: Relevant MeSSi features for this case study.

	Feature	Focus in this case study
	Multi-Level Storage	Mapping of quantized ternary NN weights to three-level RRAM states.
	Variability	Deterministic read/write margins only. No stochastic modeling.
	Devices & Operating Conditions	IHP SG13S 130 nm 1T1R RRAM configuration.
	Modularity	-
	Scalability	Abstract Python-based model enabling large-scale inference simulation.
	Interoperability	-
	Mapping Strategies	-
	Extra-Functional Properties	Read/write latency, per-access energy, and static power derived from SPICE-level characterization.
	Co-Simulation	-

5.2 Analog In-Memory Computation of Neural Networks

Motivation and Problem Statement

Analog in-memory computing with RRAM crossbar arrays offers the potential to overcome key bottlenecks of digital accelerators for deep neural networks (DNNs) [A195]. By performing vector-matrix multiplications directly within the memory array, such architectures can reduce data movement, increase effective memory bandwidth, and exploit massive inherent parallelism, thereby achieving high energy efficiency and computational density.

However, the properties of real RRAM devices deviate from the behavior of an ideal, noise-free memristor. For inference applications, each device must reliably represent multiple discrete conductance states to store quantized neural network weights (multi-level operation mode). Achieving this requires low variability, high state retention, and accurate programming to the target conductance. In practice, several non-idealities degrade inference accuracy compared to software baselines. Of particular concern is the *conductance drift* that occurs immediately after programming, even when iterative program-and-verify algorithms are used. Drift alters the stored weight values over time, introducing additional errors during inference.

At the time of this study, available computation-in-memory simulators (such as MemTorch [A38] or NeuroSim [A156]) offered only rudimentary drift modeling, often neglecting device- and array-level parameters that strongly influence drift resilience. These include the choice of programming algorithm, the specific target conductance states, and the mapping from neural network weights to device conductances. Without the ability to model these factors accurately, it is difficult to explore architectural and algorithmic design choices that mitigate drift-induced accuracy loss.

The objective of this case study, first published in [B6], was to develop a modeling and simulation approach that exposes these parameters explicitly at the system level. This enables the quantitative evaluation of drift effects and supports the co-design of programming algorithms, mapping schemes, and network quantization strategies to maximize inference accuracy under realistic device constraints. Early forms of what later became the MeSSi framework were therefore developed and applied in this study.

Application and System Context

In analog in-memory computing, the achievable inference accuracy depends not only on the intrinsic device variability but also on the precision and stability of the programming process used to store neural network weights. For multi-level operation, where each RRAM device represents one of several discrete conductance states, the program-and-verify algorithm determines both the placement of the conductance distributions and their susceptibility to drift over time. Environmental and operating conditions, particularly temperature, can further influence drift rates, read-out stability, and sensing margins, making their consideration essential for realistic evaluation.

A widely used baseline for multi-level programming is the incremental step pulse with verify algorithm (ISPVA), in which the top electrode voltage V_{TE} is gradually increased in discrete steps until the desired current threshold is reached. While effective for coarse state placement,

ISPVA exhibits abrupt conductance changes near the switching threshold, making it less suitable for fine conductance tuning across multiple low-resistance levels.

Programming Approach To improve placement precision, the *Incremental Gate Voltage with Verify Algorithm* (IGVVA) was introduced [A183]. IGVVA differs from ISPVA in that the programming pulses increment the transistor gate voltage V_G while keeping V_{TE} fixed above the set threshold, enabling more linear and gradual control of the cell current and thus the conductance. This tighter correlation between V_G and the programmed current leads to smoother conductance evolution and higher placement accuracy. In the present study, all measurement data used for parameter extraction originated from devices programmed using IGVVA under controlled temperature conditions to ensure consistency and to capture temperature-dependent behavior relevant to system operation.

Two operational variants of IGVVA exist [A196]:

- **IGVVA Set:** The device is programmed from the high-resistance state upwards to the desired low-resistance state by gradually increasing V_G in 10 mV steps from 0.5 V to 1.7 V at a fixed V_{TE} of 1.2 V, using 1 μ s pulses with 100 ns rise/fall times. This approach can achieve up to eight evenly spaced conductance levels between 50 μ S and 225 μ S, but levels in the lower half of the range are more susceptible to drift in the minutes following programming.
- **IGVVA Mixed:** Designed to mitigate drift in the lower conductance levels, this approach starts from a high-conductance state and reaches lower target levels via a controlled reset operation. Reset is performed using IGVVA-style V_G stepping (from 1.5 V to 2.9 V) at $V_{TE} = 1.2$ V. This 'mixed' use of set and reset operations results in smoother approach to the target conductance and can reduce drift for the lower conductance states, although it may increase variability for higher levels.

Mapping Approach In the simulated architecture, synaptic weights are represented using a differential mapping scheme, where each weight corresponds to the conductance difference between two RRAM devices. Since multiple conductance assignments can produce the same target difference, several mapping strategies are possible. For instance, a target difference of +50 μ S can be realized by pairing a 200 μ S device with a 150 μ S device, a 150 μ S device with a 100 μ S device, or alternatively, a 100 μ S device with a 50 μ S device.

Figure 5.2.1 compares the resulting conductance distributions for three mapping approaches, each achieving a differential state of +50 μ S. The distributions were obtained at 25 °C under the *Set* programming method, immediately after programming. It can be observed that the combination of +200 μ S and -150 μ S produces the narrowest distribution. In contrast, the combination of +150 μ S and -100 μ S results in a considerably wider spread, while the combination of +100 μ S and -50 μ S yields not only an even broader spread but also a pronounced shift in the mean.

To systematically explore such alternatives, two differential mapping schemes were developed in this work. The *High-Conductance-Reference Mapping* (HCRM) prioritizes assignments that maximize the use of high conductance values for one device in the pair, whereas the *Low-Conductance-Reference Mapping* (LCRM) favors low conductance values as the reference. If

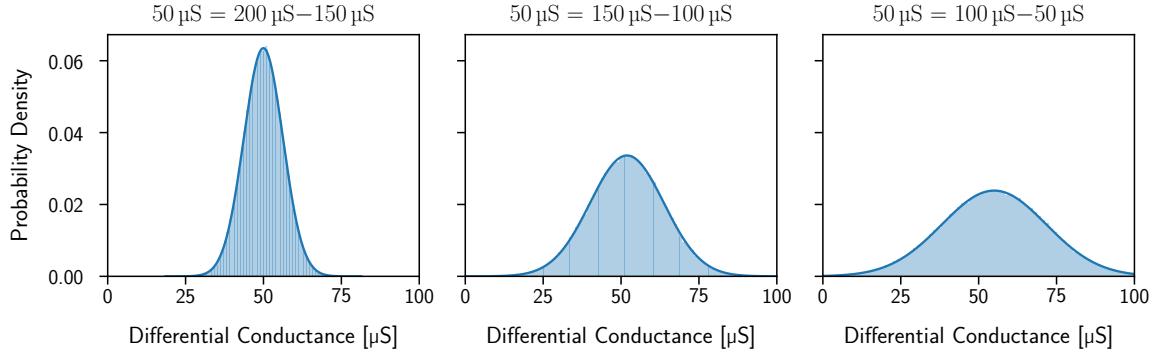


Figure 5.2.1: Conductance distributions of three mapping approaches used to obtain a differential state of $+50\ \mu\text{S}$.

high-conductance states exhibit greater stability, HCRM produces more stable differential states; conversely, if low-conductance states are more stable, LCRM is advantageous. Consequently, the choice of mapping scheme directly impacts the robustness of synaptic weight representations against conductance drift.

Modeling and Simulation Approach

In this case study, the modeling approach that would later be formalized as the DStM was introduced, although not yet under that name. The framework was developed to evaluate the inference accuracy of analog in-memory DNN accelerators under realistic device behavior, with particular emphasis on conductance drift after programming. The core principles of the DStM: discrete representation of device conductance states, statistical parameterization from measurements, and explicit time-dependence to capture post-programming drift, were already established. In combination with the modelled system components, this approach enabled a system-level representation that explicitly linked the programming algorithm, operating temperature, targeted conductance states, and post-programming conductance evolution. This capability provided a quantitative basis for assessing the impact of device-level behavior on application-level performance and for guiding the design of more robust analog in-memory DNN accelerators.

Device-level parameters were obtained from measurements of an IHP 250 nm 1T1R RRAM array programmed using the IGVVA algorithm under controlled environment conditions [A183]. For each programming variant, target state, temperature and time after programming the conductance distributions were extracted, enabling the definition of probability density functions for conductance values as a function of elapsed time after programming. These distributions were then embedded into the system-level model, allowing the inference process to be evaluated with realistic device behavior.

To integrate the device model into a system context, a set of flexible hardware component models was implemented in C++. These included parametrized crossbar array models with configurable size, data-to-conductance mapping, analog-to-digital and digital-to-analog conversion. The design supported mapping arbitrary weight matrices to RRAM-based crossbar structures, applying drift to the programmed conductances over time, and performing vector-matrix multiplications using the drift-affected values. The same components could be reused

for multiple neural network layers, enabling rapid evaluation of different architectures and mapping strategies. In later work, these C++ models were encapsulated in SystemC wrappers and incorporated into the MeSSi framework as building blocks for hybrid analog-digital system simulation. Therefore, the simulation of the in-memory matrix-vector multiplication matches the one of the MeSSi framework described in Section 4.3.

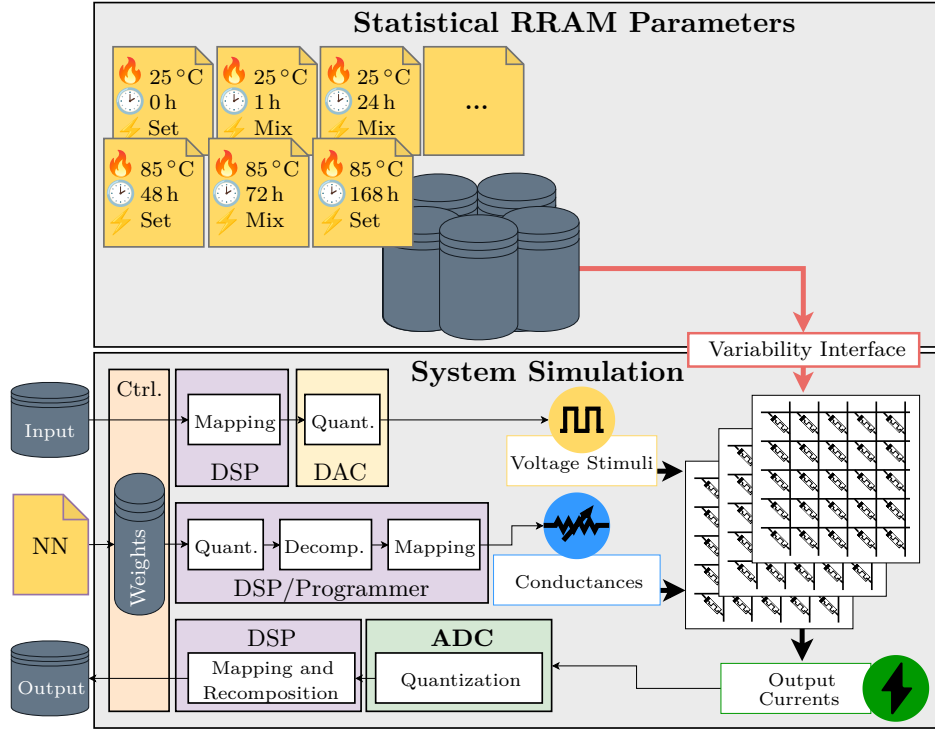


Figure 5.2.2: Block diagram of the in-memory computing platform configuration used in this case study. Adapted from [B6].

The specific simulator configuration used in this study is illustrated in Figure 5.2.2. The upper part shows the *Statistical RRAM parameters* extracted from measurements, performed for multiple programming algorithms (Set and Mixed variants of IGVVA), time points after programming, and operating temperatures (25 °C and 85 °C). These datasets were generated by an extraction script to fit statistical distributions describing the conductance states, which were then passed to the simulator through a dedicated parameter interface. The lower part of the figure depicts the *System Simulation* flow: neural network weights are quantized, decomposed if necessary, and mapped to the crossbar array, while input data is preprocessed and converted to voltage stimuli. The resulting conductances in the simulated RRAM array produce output currents, which are digitized and recomposed for the subsequent layers.

The simulation flow was designed to perform complete neural network inference using the drift-aware crossbar model. Quantized weights were mapped to conductance states according to the target programming algorithm and state definition. The programmed conductance distributions were then modified over simulated time to reflect drift, with the affected values used in all subsequent MAC operations. This allowed the impact of drift on classification

accuracy to be quantified directly at the application level, closing the loop between device behavior, circuit modeling, and system performance.

This combination of device-derived statistical modeling and configurable system components provided a flexible platform for exploring countermeasures to drift-induced accuracy loss. Strategies such as adjusting target states, modifying mapping schemes, or adapting network architectures could be tested rapidly, forming a foundation for later work on variability modeling and design-space exploration within MeSSi.

Results

The drift-aware system simulation enabled a detailed evaluation of the impact of programming algorithm, target state selection, and operating temperature on the inference accuracy of analog in-memory DNN accelerators. In the used IHP 250 nm RRAM devices, eight discrete conductance states can be reliably programmed. Throughout this section, these states are denoted as L1 (50 μ S), L2 (75 μ S), L3 (100 μ S), L4 (125 μ S), L5 (150 μ S), L6 (175 μ S), and L7 (200 μ S). The neural network under test was mapped to the simulated crossbar arrays using various state configurations, and the accuracy was monitored over extended time intervals after programming, reflecting realistic drift behavior derived from the measured devices.

Figure 5.2.3 shows the inference accuracy for four representative mapping strategies at 25 °C, using the Set variant of the IGVVA algorithm and the measured conductance drift characteristics. Each bar represents the accuracy at a given time after programming, from the after-switching (AS) point to 168 h, with the horizontal line indicating the reference accuracy without drift. A clear trend emerges: mappings that rely on closely spaced low-conductance states (e.g., L1-L4) exhibit rapid accuracy degradation due to their higher susceptibility to drift, particularly within the first 24 h. In contrast, mappings that maximize the conductance spacing (e.g., L1, L3, L5, L7) show significantly improved robustness, maintaining accuracy close to the reference over the entire observation period. This effect is consistent across both LCRM and HCRM mappings, though HCRM generally performs better due to reduced variation in the higher conductive states.

Using the most robust mapping strategy (L1, L3, L5, L7 with HCRM), the impact of both programming algorithm variant and operating temperature was investigated, as shown in Figure 5.2.4. The results demonstrate that the Mixed variant of IGVVA, which uses controlled Reset operations for certain levels, provides comparable or slightly better drift resilience compared to the Set-only variant, particularly at elevated temperatures. Nevertheless, the influence of operating temperature is pronounced: at 85 °C, a gradual but measurable decrease in accuracy is observed over time for both algorithms, whereas at 25 °C accuracy remains almost constant. From a system design perspective, these results highlight the strong interdependence between device-level behavior and application-level performance in analog in-memory computing.

Role of the Virtual Prototyping Platform

At the time of this case study, the MeSSi framework was not yet established in its final form. Instead, a dedicated C++-based simulation environment was developed to support hardware-software co-design of analog in-memory computing accelerators. This environment

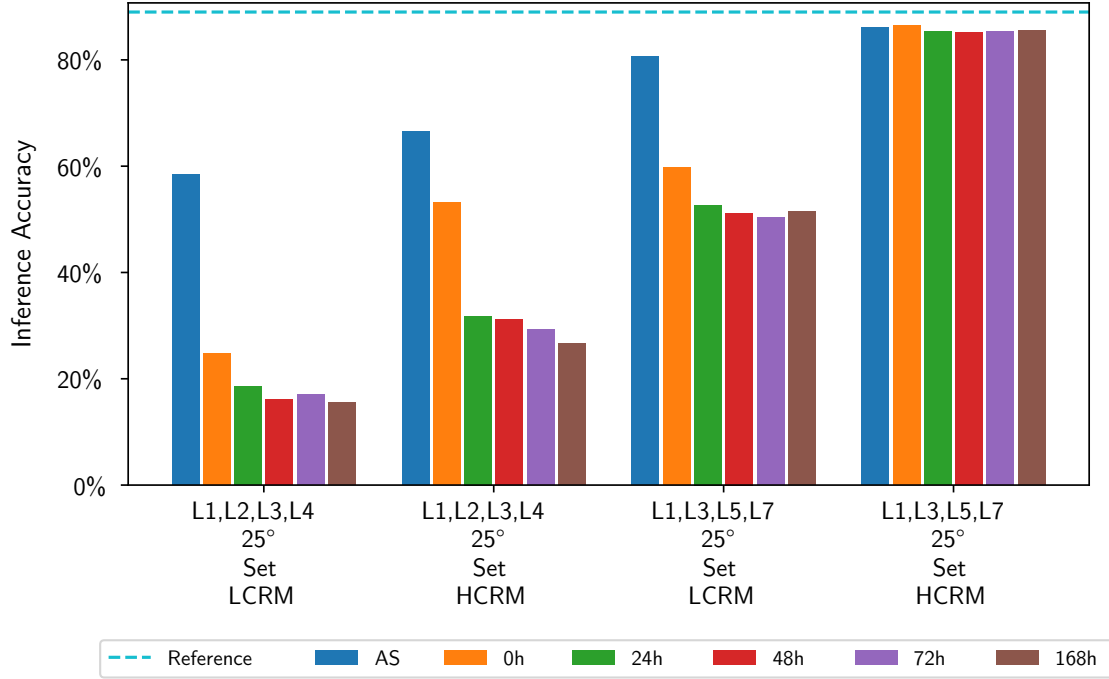


Figure 5.2.3: DNN inference accuracy for different mapping strategies. Adapted from [B6].

already contained several elements that would later become integral parts of MeSSi, including the DStM modeling approach with an extensible device parameter interface, modular hardware component models for matrix-vector multiplication, and a co-simulation interface to algorithmic neural network models.

The simulator enabled bit-accurate inference co-simulation of RRAM-based crossbar arrays, integrating measured device characteristics directly into the system model. Device parameters, including state distributions, drift behavior, and temperature dependence, were extracted from a fabricated IHP 4kbit RRAM array programmed with the IGVVA algorithm, and later reused in other case studies. For the first time, a variability-aware system-level model was realized that linked programming algorithm, operating temperature, targeted conductance states, and post-programming conductance evolution. This allowed systematic exploration of environmental and algorithmic design corners, such as *Set* vs. *Mixed* programming, 25 °C vs. 85 °C, and different post-programming retention times.

The platform also implemented variability-aware data mapping, enabling evaluation of strategies such as LCRM and HCRM under realistic device behavior, including both time-invariant variability and time-dependent drift. This functionality made it possible to quantify the robustness of weight encoding schemes and to quickly identify configurations that maximize inference accuracy over the intended system lifetime and therefore enabled the development of the optimized mapping scheme HCRM and the validation of its improved performance. Interoperability with digital control and preprocessing components further allowed the analog arrays to be placed in a realistic system context, capturing both computational accuracy and broader system-level effects.

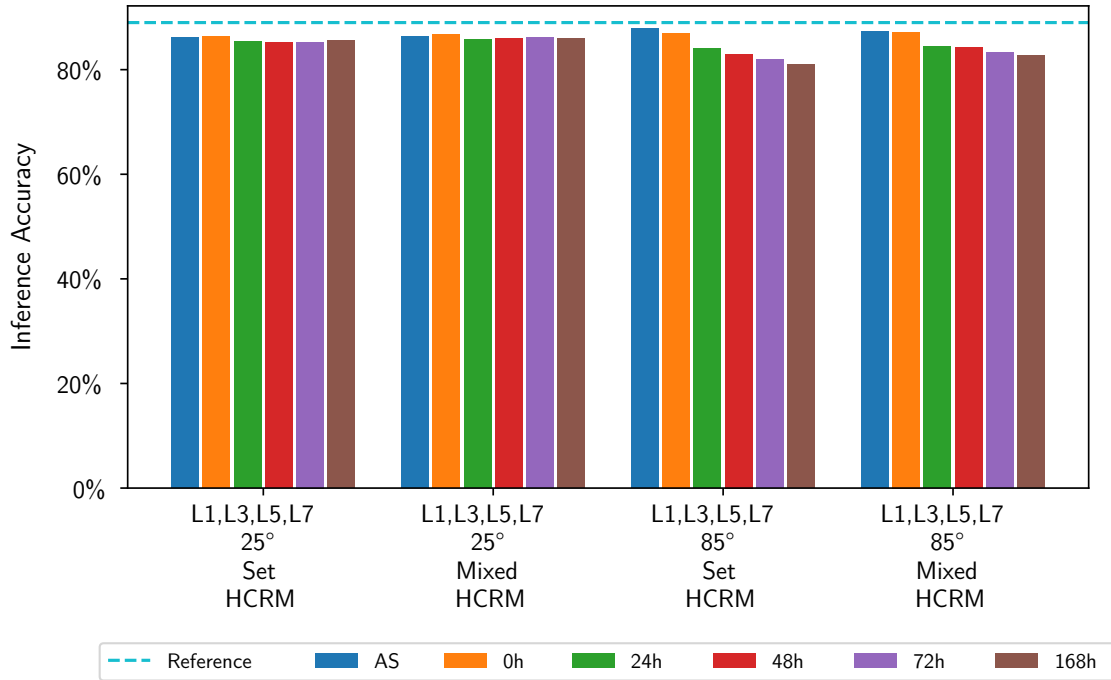


Figure 5.2.4: DNN inference accuracy over time for the most robust mapping strategy under different programming algorithm variants and operating temperatures. Adapted from [B6].

The methodological foundations developed in this case study, such as parametrized device modeling, drift-aware system simulation, mapping strategy evaluation, and environmental design corner analysis, formed a direct precursor to MeSSi. In MeSSi, these concepts were formalized and generalized, incorporating the full DStM model, extended variability parameterization, and SystemC-based interoperability with other simulated subsystems. Table 5.2.1 summarizes the platform features relevant to this case study.

Table 5.2.1: Relevant MeSSi features for this case study.

	Feature	Focus in this case study
	Multi-Level Storage	Analog computation with up to eight discrete conductance states.
	Variability	Variability-aware modeling of device-to-device variations and time-dependent drift across operating temperatures.
	Devices & Operating Conditions	IHP 250 nm 4-kbit RRAM array. Design-corner exploration of programming algorithm, temperature, and post-programming time.
	Modularity	Modular, parametrizable system component models.
	Scalability	Array-level simulation integrated into a full analog in-memory DNN accelerator model.
	Interoperability	-
	Mapping Strategies	Evaluation of LCRM and HCRM strategies to identify the most robust configuration.
	Extra-Functional Properties	-
	Co-Simulation	HW-SW co-simulation of NN workloads and RRAM-based crossbar arrays.

5.3 Radiation-Resilient Memory Subsystem

Motivation and Problem Statement

Satellites operate in a harsh and highly dynamic radiation environment, where extreme thermal variations and high-energy particles, especially during solar particle events (SPEs), can cause transient and permanent faults in electronic systems. While fault-tolerant processing platforms for space have seen significant advances, the memory subsystem has received comparatively little attention. Most existing designs rely on SRAM or DRAM [A197], which are inherently more vulnerable to radiation and require static protection measures such as error correction codes or redundancy applied continuously, leading to unnecessary power and performance penalties during nominal conditions.

RRAM offers intrinsic advantages for space use: it is non-volatile, highly energy- and area-efficient, and inherently more resilient to radiation due to ion-based data storage [A172]. However, multi-level RRAM devices exhibit variability in conductance states, and radiation-induced shifts can lead to bit errors, especially when storing multiple bits per cell. Moreover, most protection approaches for RRAM are static, keeping hardening active at all times rather than adapting to the actual space environment.

The central challenge addressed in this case study, which was first published in [C2] and [B7], is therefore the design, modeling, and evaluation of a radiation-aware, dynamically reconfigurable RRAM-based memory subsystem for a satellite SoC platform. The system should adapt its protection level based on real-time and predicted radiation conditions, balancing fault tolerance, power efficiency, and performance. This requires linking device-level radiation effects to system-level error rates and, ultimately, to the correct execution of application workloads, something that existing models and evaluation frameworks have not enabled before.

TETRISC Architecture and RRAM Extension

The baseline for this case study is the TETRISC (TETra Core System based on RISC-V) platform, a RISC-V-based quad-core SoC designed for adaptive reliability in harsh environments such as space [A198]. It builds on the PULPissimo platform, an open-source microcontroller SoC developed by the PULP project [A199], and integrates four RI5CY cores, a set of on-chip monitor for single-event upsets (SEUs), aging, and temperature, as well as the *HiRel Controller* for dynamic fault-tolerance management. SEUs are transient faults in digital circuits caused by high-energy particles, which can lead to bit flips in memory cells or logic registers. The *HiRel Controller* enables switching between high-performance, low-power, and fault-tolerant modes at runtime based on real-time environmental conditions.

Figure 5.3.1 shows a block diagram of the platform, with the changes introduced in this case study highlighted in green. Four CPU cores are used and can be switched between three operation modes. In the *power-saving* mode, only a single core is active to execute tasks. This consumes the least power but is also the most susceptible to radiation-induced errors. In the *high-performance* mode, all cores are active simultaneously, allowing for faster processing of multiple applications in parallel. However, this comes at the cost of higher power consumption while maintaining the same susceptibility to radiation-induced errors as

in normal operation. In the *fault-tolerant* mode, all four CPU cores execute the same task. Their results are then compared, and a Quad-Modular Redundancy (QMR) voter determines the correct output before it is written to memory. Switching between the operation modes depends on predictions of solar events by the SEU monitor. This component detects bit errors introduced in a small SRAM block and uses them to predict the space radiation environment [A200]. A similar approach is evaluated for the memory banks in this case study.

The CPU cores are connected to four memory banks via a tightly coupled memory interconnect. In the original design, each bank consists of 40 kB SRAM. For this work, they were replaced with non-volatile RRAM-based memory banks to exploit their inherent radiation tolerance, non-volatility, and potential for higher storage density. These banks can also be switched between *normal* and *fault-tolerant* modes. This change required modifications to the memory access infrastructure, including the addition of a programmable majority voter to support redundant storage across multiple banks and majority voting during reads.

The right half of Figure 5.3.1 details the *HiRel Controller*, which coordinates adaptive fault-tolerance measures for both CPU cores and memory banks. It contains programmable majority voters for each domain (CPU and memory), input and output multiplexing logic, and control logic for synchronization, interrupt generation, and clock gating. In the extended design, it also manages an *RRAM Power Control* unit, allowing unused memory banks to be power-gated to reduce energy consumption without data loss. This enables dynamic reconfiguration of the memory subsystem, for example, powering on additional banks for redundancy during predicted radiation events or powering them down during nominal conditions.

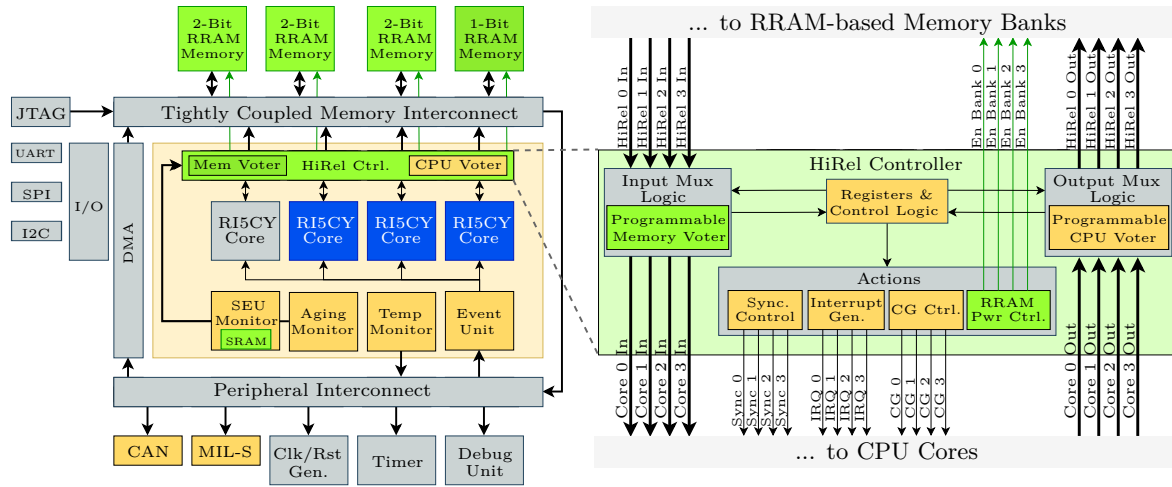


Figure 5.3.1: TETRISC SoC architecture with the RRAM-based memory bank extension and HiRel Controller structure. Adapted from [B7].

Research Questions and Approach

This case study set out to investigate how a radiation-resilient multi-level RRAM subsystem could be integrated into the TETRISC satellite SoC platform, and how such a subsystem could adapt dynamically to varying radiation conditions. Four overarching research questions were defined:

1. **Memory subsystem architecture:** How should the memory subsystem be structured to allow switching between high-performance, low-power, and fault-tolerant operation without compromising interface compatibility?
2. **Protection strategy trade-offs:** Which error-mitigation schemes (such as triple- or quad-modular redundancy (TMR/QMR), extended bit-to-state mapping, or single-bit-per-cell operation) provide the best balance between reliability, storage density, and power consumption under different radiation scenarios?
3. **Adaptive operation:** How can real-time radiation monitoring and prediction be leveraged to adjust the protection level dynamically, enabling energy-efficient operation during nominal conditions while maintaining robustness during high-radiation events?
4. **System-level impact:** How do radiation-induced device-level errors propagate through the memory subsystem to affect application-level behavior, and what configurations minimize the impact on realistic satellite workloads?

To address these questions, the simulation platform, already introduced in the previous case study, was extended with:

- An abstract radiation model parametrized from nominal-condition measurements of IHP RRAM devices and published radiation data.
- Power and energy simulation for multi-level RRAM banks, accounting for static dissipation, read/write energy, and mode-specific bank usage.
- An application-level fault injection environment based on a modified QEMU RISC-V emulator, enabling controlled study of fault effects on diverse benchmark applications.

This setup enabled a multi-level evaluation: from radiation environment, through device physics and memory architecture, to full application execution. This formed the basis for the design of the proposed adaptive, radiation-resilient TETRISC+RRAM architecture.

Simulation Setup and Co-Simulation Strategy

To address the research questions outlined above, the evaluation of the TETRISC architecture with RRAM-based memory extensions was performed using a co-simulation environment that combines detailed memory-domain fault modeling with application-domain reliability analysis. The memory subsystem was realized as a dedicated configuration of the MeSSi virtual prototyping platform, enabling accurate modeling of access latencies, energy consumption, and fault behavior under radiation. This builds directly on the modular architecture of the MeSSi framework and the DStM-based device model, ensuring that the radiation-aware RRAM modeling used here can be reused or extended in other application contexts.

As illustrated in Fig. 5.3.2, the simulation is divided into two domains. In the *memory domain*, the RRAM-based subsystem is modelled within MeSSi using DStM devices parametrized with a radiation-specific configuration. These configurations consist of error probability distributions for different radiation intensities, derived from nominal-condition measurements of IHP devices combined with published data for similar technologies under radiation. The parameters are stored as DStM parameter configuration files and loaded at simulation start, enabling seamless integration into the overall evaluation flow. From this

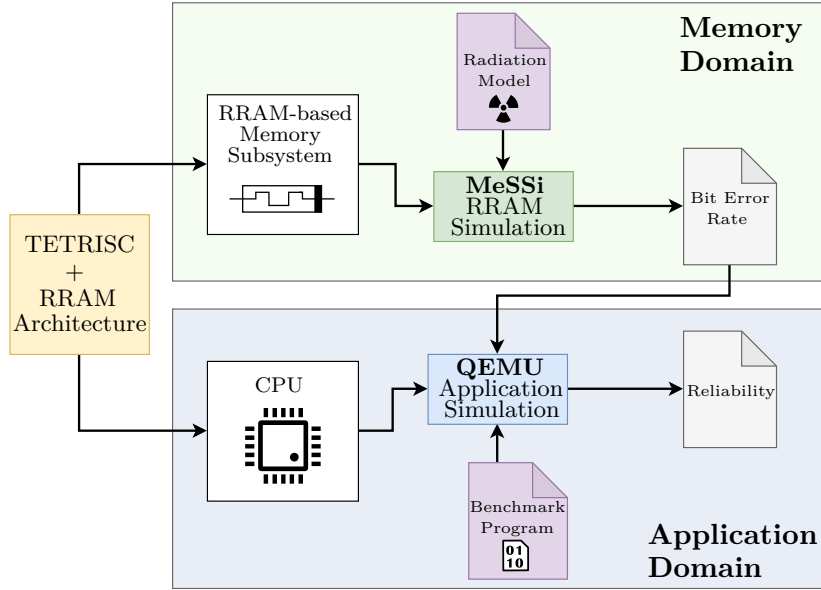


Figure 5.3.2: Overview of the co-simulation approach, combining memory-domain fault modeling with application-domain reliability analysis. Adapted from [B7].

simulation bit error rate (BER) profiles for each radiation condition can be exported during operation of the RRAM-based memory banks.

In the *application domain*, the RISC-V CPU cores of the TETRISC platform and their workloads are simulated using QEMU, which executes target application code on a functional CPU model. The BER profiles from the memory-domain simulation are used to inject bit flips and transient faults into memory transactions, directly affecting the running application. This makes it possible to correlate device-level fault behavior with system-level reliability metrics and to assess the effectiveness of different fault-tolerance strategies.

The specific simulation configuration for the RRAM subsystem of this study is shown in Fig. 5.3.3. It consists of multiple in-memory RRAM banks, each with dedicated ADCs for read-out and controlled by a parametrizable peripheral logic layer. This layer handles operation sequencing and bus interfacing to the tightly coupled memory interconnect of the TETRISC system. Radiation profiles for simulating single-event upsets and multi-bit errors are injected as DStM configuration parameters. This modular setup allows rapid switching between design corners, such as bits-per-cell configuration and radiation levels, and facilitates HW/SW co-simulation where the timing and fault behavior of the memory directly influence application execution in QEMU. The following section presents the results obtained from this multi-domain simulation setup, structured to answer each of the research questions defined earlier.

Results

The simulation results address the research questions defined for this case study, linking radiation environment, device-level behavior, memory subsystem configuration, and application-level reliability. Radiation intensities are expressed as R10, R20, R30, R40, and R50, rep-

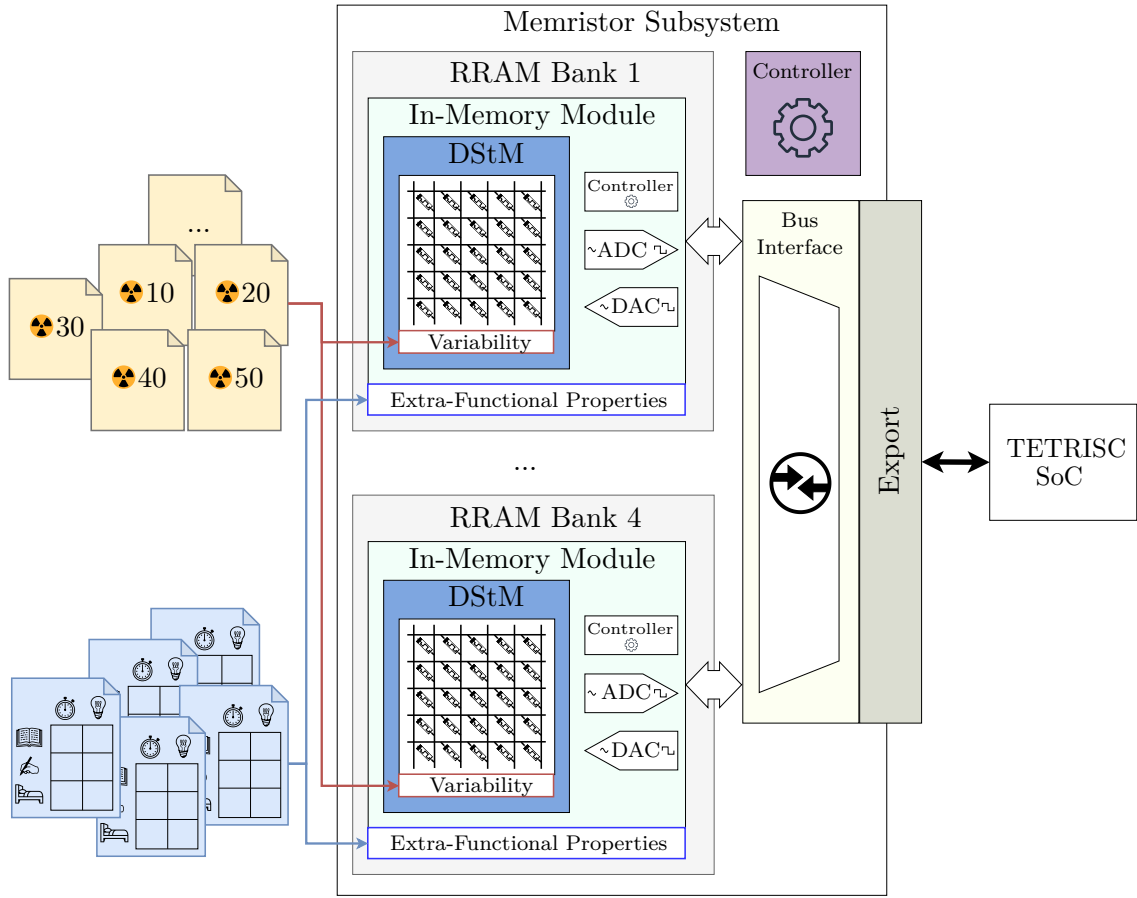


Figure 5.3.3: Detailed simulation flow of the radiated RRAM-based memory subsystem within the MeSSi platform.

representing progressively harsher space radiation conditions, from nominal low-radiation operation (R10) to severe solar particle events (R50). In the DStM radiation model, these conditions are emulated by shifting the mean conductance of each discrete RRAM state towards higher values, with the shift magnitude increasing with radiation intensity. Higher-conductance states are less affected, which is modelled as an exponentially decreasing shift for the upper three states. The shift model was parametrized by fitting the exponential function

$$G = a \cdot b^n$$

to published radiation measurement data [A201–A203] and applying it to the unirradiated IHP device measurements. The resulting parameters are given in Table 5.3.1.

Device- and subsystem-level results. Six different memory subsystem configurations were evaluated: the unprotected configuration, the TMR configuration, the QMR configuration, and their extended variants, each with 2 bit of data stored per RRAM device, as well as a configuration with only 1 bit of data stored per RRAM device. In the unprotected configuration, a single memory bank is used and data is stored without redundancy. In the

Table 5.3.1: Parameters of the exponential radiation model $G = a \cdot b^n$, where n is the discrete radiation intensity level index.

Conductance Level	a (μS)	b	Radiation Intensity	n
L1	43.20	1.022565	R10	1
L2	92.47	1.010592	R20	2
L3	141.70	1.002535	R30	3
L4	202.30	1.000009	R40	4
			R50	5

TMR and QMR configurations, three or four RRAM memory banks are used, respectively, to store the same data redundantly. A TMR or QMR majority voter then determines the correct values before they are transferred to the processing cores. In the extended TMR and QMR configurations, data is also stored redundantly across three or four banks. However, in this case, the mapping between data bits and conductance states is varied for each bank.

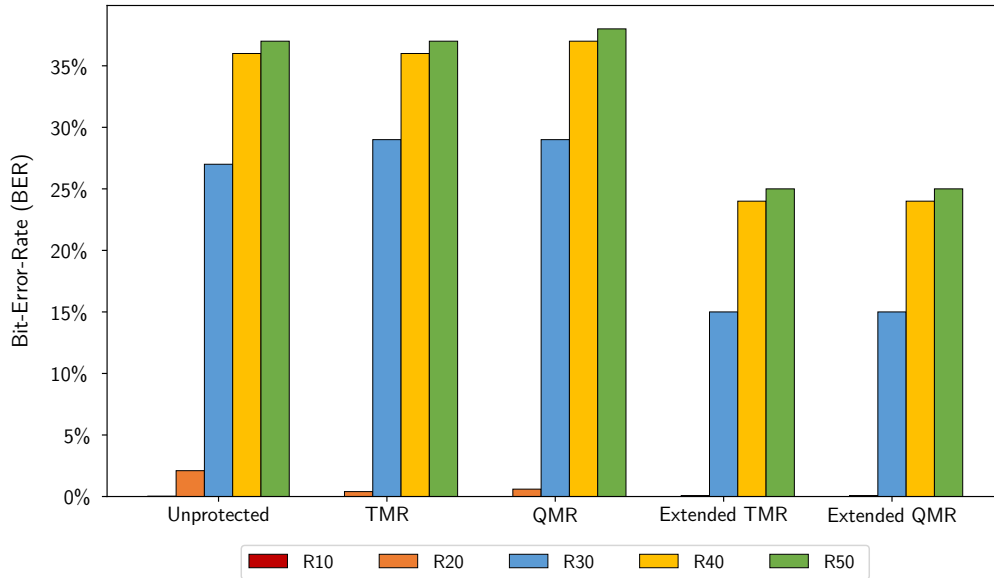
**Figure 5.3.4:** Simulated Bit Error Rate (BER) for different protection schemes and radiation intensity levels. Adapted from [B7].

Figure 5.3.4 summarizes the simulated BER values for different radiation levels. In the unprotected configuration, BER remained negligible at low radiation levels (R10) but rose sharply at moderate levels ($\approx 27\%$ at R30) and saturated at $\approx 37\%$ for R50. Standard TMR and QMR reduced BER to below 1% for radiation levels up to R20, but offered no improvement at higher intensities due to correlated state misclassifications across banks. Extended TMR and QMR, which remap bits to different RRAM state encodings in each bank, nearly eliminated BER up to R30 and reduced it by $\approx 33\%$ to 50% for R40 and R50 compared to the standard schemes, without introducing additional hardware overhead.

From a device configuration perspective, 1-bit-per-cell operation completely eliminated BER for all simulated radiation levels, but doubled the dynamic power consumption. Pure 2-bit-

per-cell operation provided higher density but reduced resilience. The optimal trade-off was found in a hybrid configuration: using three banks in 2 bit mode and one in 1 bit mode, combined with the extended TMR scheme, reduced the BER under high radiation to zero, while limiting the average power overhead to $1.5 \times$ the unprotected baseline configuration.

When the memory subsystem is switched to *normal* operation mode, unused RRAM banks can be fully power-gated, reducing their power draw to zero while still retaining all stored data. When switched to *fault-tolerant* mode, the additional banks can be powered on, and data can be stored in 1 bit-per-device mode.

Application-level results. Application-level fault injection using the *Embench* benchmark suite showed that BER improvements do not always yield proportional gains in correctness. Compute-bound workloads with low memory activity (e.g., *aha-mont64*) maintained high correctness rates even under elevated BER, whereas memory-intensive workloads (e.g., *statemate*, *nettle-aes*) suffered severe degradation, often dominated by crashes rather than silent data corruption.

Failure mode analysis. At high BER, crashes were the dominant failure mode, easily detectable but halting execution entirely. At moderate BER, incorrect results became more frequent, posing a risk of undetected errors without explicit result validation. Timeouts were rare but remain relevant for safety-critical scenarios.

Overall assessment. The findings indicate that a flexible memory subsystem, combining extended TMR for general workloads with selective 1 bit mode for critical code and data during predicted high-radiation intervals, offers the best balance between performance, energy efficiency, and resilience for space applications. Leveraging real-time radiation monitoring (e.g., from the TETRISC SEU sensors) allows activating these high-protection modes only during critical periods ($\sim 2.5\%$ of mission time), significantly reducing the long-term energy cost of fault tolerance.

Role of the Virtual Prototyping Platform

For this case study, the Virtual Prototyping Platform was central to linking device-level RRAM characteristics, radiation effects, and the full system-level behavior of the TETRISC+RRAM architecture. The memory subsystem, implemented as a dedicated configuration of the MeSSi platform, enabled hardware/software co-simulation of the SoC in which statistical device variability and radiation sensitivity were described by distributions extracted from IHP RRAM measurements and extended with radiation-induced shift models derived from literature. This made it possible to capture not only intrinsic device variability, but also the influence of environmental conditions such as radiation intensity, duration, and occurrence probability.

By integrating these effects directly into the simulation flow, the platform supported exploration of a wide design space under consistent conditions. Extra-Functional properties, such as energy consumption, and error rates, could be evaluated alongside functional correctness, enabling trade-off analysis across performance, energy efficiency, and reliability. Different operation modes (high-performance, low-power, fault-tolerant) could be simulated without

modifying the architecture model, and switching between design corners, such as alternative programming algorithms, bits-per-cell configurations, or redundancy schemes, was straightforward. The framework also supported variability-aware data mapping, allowing bit-to-state assignments to be optimized for state-specific radiation resilience, as in the extended TMR/QMR configurations evaluated here.

A transaction-level modeling interface connected the RRAM subsystem to the digital components of the TETRISC SoC, enabling end-to-end evaluation of system behavior under fault conditions. This allowed direct linking of bit error rates from the memory-domain simulation to application-level fault injection experiments in a QEMU-based framework, correlating physical effects at the device level with the success rates and failure modes of complete algorithms.

The co-simulation workflow thus enabled true hardware-software co-design for a radiation-resilient memristor-based system, combining device-level modeling, memory subsystem architecture, and system-level execution within one unified platform. This marked the first time that the key capabilities of MeSSi (parametrized device modeling, environmental fault modeling, variability-aware mapping, and TLM-based subsystem integration) were present together in a single use case. Table 5.2.1 summarizes the platform features relevant to this case study.

Table 5.3.2: Relevant MeSSi features for this case study.

	Feature	Focus in this case study
	Multi-Level Storage	Evaluation of 2-bit and 1-bit-per-cell configurations, including adaptive switching between modes for radiation resilience.
	Variability	Modeling of state-specific sensitivity to radiation-induced conductance shifts.
	Devices & Operating Conditions	IHP RRAM cells characterized under nominal conditions. Radiation behavior extrapolated from similar devices across multiple intensity levels.
	Modularity	Parametrizable architecture model with interchangeable fault-mitigation strategies (TMR, QMR, extended mapping, bit-per-cell change).
	Scalability	Simulation of multiple memristor-based memory banks with 40 kB each.
	Interoperability	TLM-based interface between the SystemC memory model and digital SoC simulation for end-to-end fault analysis.
	Mapping Strategies	State-permutation mappings for improved majority-voting resilience.
	Extra-Functional Properties	Joint evaluation of BER, performance, and power consumption under realistic radiation event duty cycles.
	Co-Simulation	HW-SW co-simulation linking memory-domain radiation modeling to QEMU-based application-level fault injection.

5.4 RISC-V CPU with Integrated RRAM

Motivation and Problem Statement

Designers of digital hardware are increasingly considering architectures that make use of memristive devices, for example in the form of RRAM-based flip-flops, register files, or caches. A key challenge in this context arises before fabrication: system components must be validated against realistic device behavior to ensure correct operation and to assess the impact of device-level variability on overall performance and reliability. Setting up such an evaluation with real memristor devices is both expensive and time consuming, since it requires a dedicated test infrastructure [A204]. In addition, exploring different operating conditions, such as alternative programming algorithms, retention times after programming, or varying temperatures, further increases the effort [A170] and makes large design-space studies impractical.

The MeSSi virtual prototyping framework provides a way forward by abstracting device-level characteristics into stochastic models, enabling system-level simulation of memristive subsystems. Yet, when these simulations are performed purely in software, they cannot be directly connected to real digital hardware designs, which limits their usefulness for hardware verification and pre-silicon validation.

The research question of this case study, first published in [B8], therefore is:

Can the MeSSi virtual prototyping framework, including its stochastic memristor models, be transferred to FPGAs in order to accelerate system simulation and to act as a hardware-in-the-loop (HIL) simulator?

By answering this question, the study explores whether an FPGA-based realization of MeSSi can both (i) achieve higher than real time performance compared to device-level experiments, and (ii) provide a practical bridge between virtual modeling and real hardware designs, enabling pre-silicon evaluation of digital systems that are intended to operate with memristive memory elements.

FPGA-Based Realization of MeSSi

To address the research question, the modeling concepts of the MeSSi platform were transferred from a software-based SystemC environment to FPGA. The core idea was to preserve the same device-level abstractions through confusion matrices, in particular the Discrete Stochastic Memristor (DStM) model and the representation of discrete memory operations, while at the same time exploiting the parallelism and digital interfaces of a real hardware implementation.

The confusion matrix provides a compact way to describe the probabilistic behavior of memristive devices under read and write operations. Each entry in the matrix captures the probability of transitioning from one conductance state to another, which results from a combination of the underlying memristor device model (parameterized by measurement data) and the behavior of the ADC. The flow for generating the transition table is visualized in Figure 5.4.1. Raw data is characterized into statistical conductance distributions, mapped according to the ADC configuration to derive confusion matrices, and finally transformed

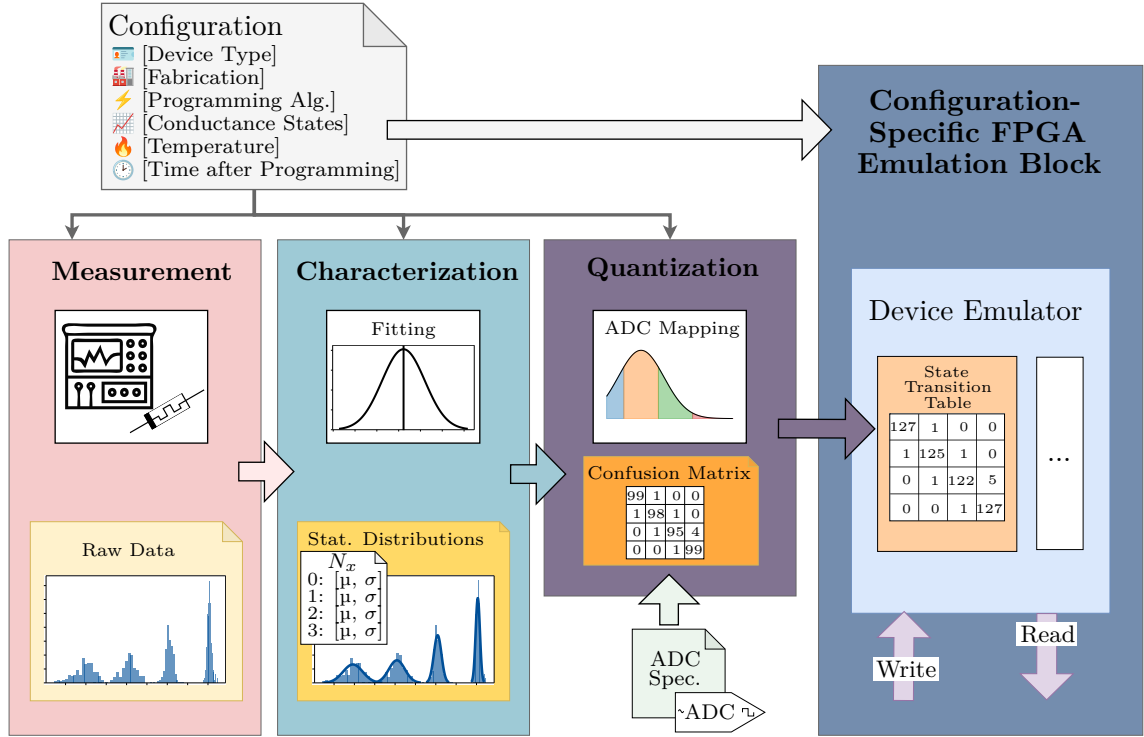


Figure 5.4.1: Modeling flow from device measurements to FPGA emulation.

into configuration-specific state transition tables for FPGA-based device emulation. In the specific setup used for this case study, an ideal ADC was assumed with thresholds placed midway between two neighboring conductance states. However, the approach is general: if a different ADC characteristic is chosen, or if conductance distributions change due to alternative device measurements, programming algorithms, or operating conditions, the transition table must be regenerated accordingly.

In the FPGA implementation, the confusion matrix is stored as a transition table that can be accessed in constant time. Random sampling of transitions is realized using a linear-feedback shift register (LFSR), which generates pseudo-random numbers to resolve the stochastic outcomes of state changes. This approach allows the stochastic device dynamics to be reproduced in hardware with minimal resource overhead. When incorporating the confusion matrix into the design the bit width of the generated random value is taken into account to scale the values of the transition table accordingly.

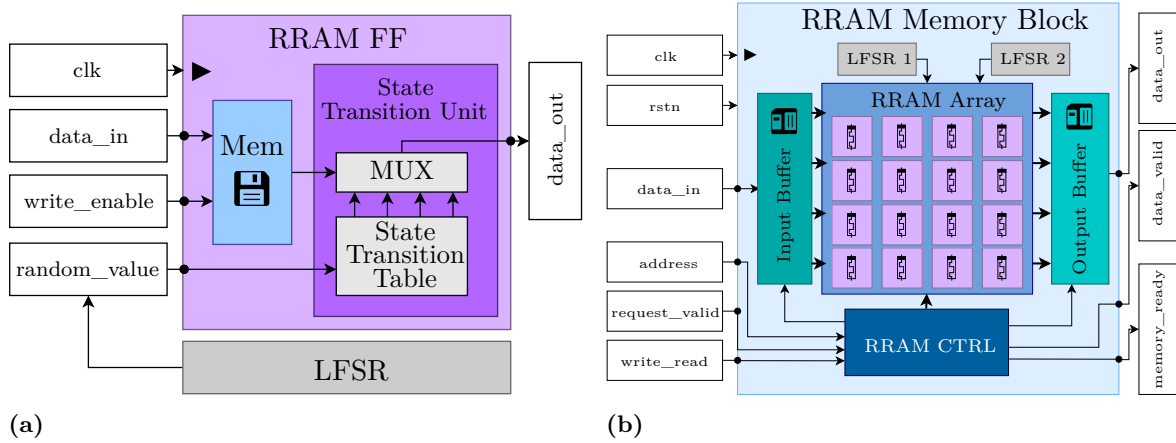
Two simulation blocks were developed: an RRAM-based flip-flop block for representing individual storage elements and an RRAM-based memory block for modeling larger arrays. Both blocks were integrated into the FPGA design as synthesizable modules, making them directly accessible to other hardware components. In this way, digital circuits such as processors or controllers can interact with the simulated memristor devices through standard interfaces, as if they were connected to physical arrays.

Table 5.4.1: Average properties of the implemented simulation blocks, based on multiple synthesis runs and simulated number of devices.

IP	$f_{\max}$	Simulation Duration		Resources per cell	
		Read	Write	FF	LUT
RRAM FF	470 MHz	2 ns	2 ns	2.1	4.9
RRAM Memory Block	200 MHz	10 ns	5 ns	2.4	9.8

The basic principle of these blocks is illustrated in Fig. 5.4.2. The flip-flop block (Fig. 5.4.2a) contains a memory element to store the target conductance state and a state transition unit, which applies the probabilistic state changes defined by the transition tables. A pseudo-random value generated by a LFSR determines the outcome of each read operation, ensuring that the stochastic behavior (including time dependent effects) of the memristive device is faithfully reproduced.

The memory block (Fig. 5.4.2b) extends this concept to an array of cells. Input and output buffers provide the data interface, while a controller manages read and write operations and status signaling. For each write access, the target conductance index of the addressed cell is updated. For each read access, the correct entry in the transition table is selected and sampled using random values generated by a LFSR. Since an LFSR produces pseudo-random values with a fixed period, sharing the same sequence across multiple devices may result in identical outcomes for cells that are in the same state. To reduce this correlation different bit masks per device are used and additionally, multiple LFSRs can be employed within the memory block, ensuring that devices with identical target states do not necessarily generate the same output values. This structure allows larger memory subsystems, such as caches, to be emulated with realistic device-level variability while remaining accessible through standard memory control signals.

**Figure 5.4.2:** Block diagrams of the implemented emulation blocks. Adapted from [B8].

The properties of the implemented simulation blocks synthesized for a Xilinx ARTIX-7 FPGA are shown in Table 5.4.1. The supplied transition tables were generated from IHP 250 nm devices and four different target conductance states were used.

This FPGA-based realization turns MeSSi into a hardware-in-the-loop simulator: the memristive behavior is not emulated by physical devices, but by a faithful, hardware-accelerated model. This enables digital designs to be tested and validated under realistic memory conditions before an actual tape-out with real memristors is performed.

Evaluation of a RISC-V CPU with RRAM-Based Components

The FPGA-based realization of MeSSi was evaluated in the context of a RISC-V processor with memristive storage elements. Two subsystems were studied: an RRAM-based register file and an RRAM-based cache memory.

An overview of the complete evaluation flow is shown in Figure 5.4.3. Application benchmarks from the *Embench* benchmark suite were executed in a QEMU-based simulation of the RISC-V architecture. During execution, representative register and cache access patterns were recorded through tracing extensions. These traces were then applied to the FPGA-based memristor emulation blocks, where each access was resolved according to transition tables derived from device measurements and ADC mapping. For every access, the stochastic outcome was determined in hardware, and error statistics were aggregated into bit error rates (BER). In addition, the total simulation duration of all memory accesses per benchmark was evaluated to quantify performance.

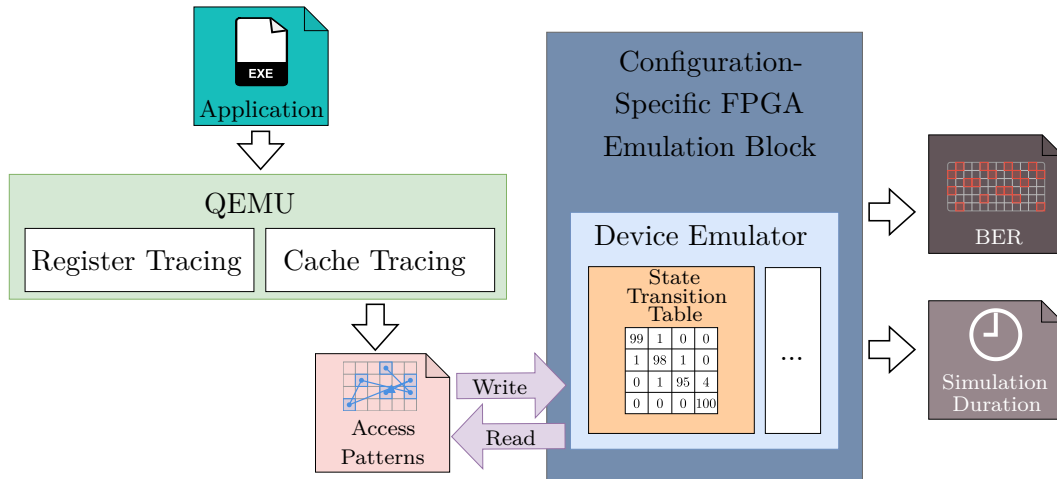


Figure 5.4.3: Evaluation flow from QEMU-based execution traces to FPGA-based memristor emulation and BER estimation.

Results

Reliability Evaluation

The BER was investigated for register files under different temperature conditions and benchmark workloads from the Embench suite. At 25 °C, the average BER across all registers was below 1 % for a 32-bit register, corresponding to fewer than one bit error per register on average. Such errors can be corrected with lightweight error correction schemes, for example simple Hamming codes. At elevated temperatures, error rates increased significantly. At 125 °C, the maximum BER observed for the Nettle AES benchmark was about 20 % in

a single register. Under these conditions, more advanced codes such as Bose-Chaudhuri-Hocquenghem-Codes (BCH) would be required to ensure reliable operation. The overall comparison of the BER evaluated with the FPGA implementation and the SystemC-based MeSSi implementation showed excellent agreement, with deviations below one percentage point on average and never exceeding four percentage points.

Performance Evaluation

The FPGA simulation blocks were clocked at 150 MHz and provided substantial acceleration compared to both software simulation and real device access. Relative to real RRAM devices, the FPGA-based approach achieved an average speedup of about $1200\times$ for write accesses and $7.5\times$ for read accesses, thereby consistently outperforming the actual device latency and enabling faster-than-real-time simulation. Compared to the SystemC implementation of MeSSi, a speedup between $8\times$ and $23\times$ for writes and between $140\times$ and $600\times$ for reads was observed, depending on the memristor array size and benchmark algorithm. These results demonstrate that FPGA-based MeSSi not only reproduces the functional behavior of memristive devices with high accuracy, but also enables efficient large-scale evaluation campaigns that would otherwise be prohibitively slow.

Role of the Virtual Prototyping Platform

In this case study, the role of the MeSSi virtual prototyping platform is extended from a purely software-based simulation framework to an FPGA-based hardware-in-the-loop simulator. By transferring the discrete stochastic modeling concepts to FPGA hardware, MeSSi provides the same variability-aware abstraction of memristor devices in a form that can interact directly with digital system designs. This ensures methodological consistency between software simulation and hardware-assisted prototyping.

The FPGA implementation allows MeSSi to serve as a surrogate for physical memristor arrays. Instead of requiring costly and time-consuming device test setups, realistic device behavior can be reproduced in hardware and connected to processor, accelerator designs, or other digital components prior to fabrication. This enables pre-silicon validation under realistic memory error conditions, bridging the gap between device-level characterization and system-level architecture evaluation.

Furthermore, the FPGA-based realization enables simulation speeds beyond real time. This makes it possible to rapidly explore the impact of operating conditions, device parameters, and architectural choices, providing early feedback in the design process when design decisions are most critical.

Overall, the virtual prototyping platform assumes the role of a unifying framework: it offers flexible software-based exploration, while at the same time enabling efficient hardware-assisted evaluation and hardware-in-the-loop validation. In this way, MeSSi significantly broadens the range of design studies that can be conducted, from device modeling to full system-level verification. Table 5.4.2 summarizes the platform features relevant to this case study.

Table 5.4.2: Relevant MeSSi features for this case study.

	Feature	Focus in this case study
	Multi-Level Storage	Discrete multi-level memory operation represented by confusion matrices.
	Variability	Variability-aware model including ADC quantization effects.
	Devices & Operating Conditions	IHP 250 nm 4-kbit RRAM array. Evaluation under varying temperatures and access patterns.
	Modularity	FPGA-based emulation blocks for RRAM-based flip-flops and memory arrays, directly connectable to digital system designs.
	Scalability	Hardware implementation achieving faster-than-real-time operation, enabling large-scale access-trace evaluation.
	Interoperability	Integration with digital hardware blocks implemented on FPGA.
	Mapping Strategies	-
	Extra-Functional Properties	Energy and latency estimation supported through shared models with the SystemC framework.
	Co-Simulation	Hardware-in-the-loop simulation with FPGA-based memristor models interacting with QEMU-based system execution.

5.5 Parameter Study of a Memristive Analog Neuron Circuit

Motivation and Problem Statement

In the previous case studies, variability models were directly parameterized using measurement data from IHP RRAM devices, which enabled accurate evaluation of device- and system-level effects. In practice, however, detailed statistical characterization of a given memristor technology may not always be available. This raises the question of how variability can be estimated realistically without resorting to simplified assumptions such as Gaussian distributions with a user-defined, fixed relative standard deviation across all conductance states, such as applied in other system-level memristor frameworks (see Section 2.5 and Section 3.4).

To address this question, this case study investigates the application of the measurement-driven modeling approach to an analog neuron circuit comprising memristive elements. The circuit features discrete operating points that are defined by the high- and low-resistance states of the memristors. This property renders it a suitable candidate for evaluating the methodology beyond purely digital memory applications. To enable such an evaluation, the DStM modeling approach developed within MeSSi was extended through the introduction of the Variability Intensity Model (VIM). By relating variability directly to the nominal conductance states rather than introducing it as a fixed global factor, VIM provides preliminary estimates of variability effects on circuit-level behavior while maintaining realistic state dependence, even in the absence of comprehensive measurement data.

Methodology

The Hindmarsh-Rose neuron model provides a mathematical description of the spiking and bursting behavior observed in biological neurons [A205]. Approximating this model by an electrical circuit offers a means to build evaluation environments for neuromorphic concepts. A realization with memristive devices is particularly attractive, as it reduces the number of required active circuit elements and enables a comparatively simple implementation [A97].

Up to now, this circuit approximation has been analyzed under the assumption of idealized component behavior. The research question addressed in this case study, which was first published in [B11], is therefore whether the memristive approximation of the Hindmarsh-Rose neuron model can be brought one step closer to practical implementation by evaluating its operation at critical operating points under the influence of realistic memristor variability.

Simulation Approach The circuit was simulated using a wave-digital model [A206], which ensures numerical stability and enables the efficient evaluation of nonlinear components. The memristive devices follow established wave-digital RRAM models [A207], with the high-resistance state simplified to a constant value in order to directly assess robustness against variations in this parameter.

Variability Introduction To reflect the stochastic nature of real RRAM behavior, variability was introduced on top of the wave-digital device models. The main focus was on the memristive parameters, in particular the conductance states, switching thresholds, and switching speeds, which were altered independently at every simulation step to emulate both time-dependent and time-invariant variations. In addition, absolute parameter offsets were

applied to capture fabrication-related deviations. Additionally, tolerance analyses were also performed for the passive circuit components in order to assess their relative influence on the overall robustness of the neuron dynamics. To define realistic stochastic variations of the memristor parameters, a dedicated model was required. This led to the development of the Variability Intensity Model (VIM), described in the following.

Variability Intensity Model (VIM) Traditional system-level memristor models typically represent device variability by assigning a fixed coefficient of variation (i.e., relative standard deviation) to all conductance states (see discussion in Section 2.5). Each simulated state is then modeled as a normal distribution with a mean equal to the target conductance and a standard deviation given by this coefficient of variation multiplied by the mean.

However, experimental observations (see Section 3.4) indicate that this assumption does not hold in practice: the variability of memristive devices depends strongly on the nominal conductance, with lower conductance states exhibiting significantly larger fluctuations. The DStM addresses this issue when sufficient measurement data is available by directly parameterizing statistical models from device characterization results. In cases where such detailed data is unavailable, an alternative approach is required to estimate realistic variability behavior.

The Variability Intensity Model (VIM) was developed for this purpose by the author of the thesis at hand. Instead of assuming a constant relative standard deviation, VIM establishes an empirical correlation between the nominal conductance state and its standard deviation for a given device type. This relationship, derived from analytical evaluation of measured conductance values of IHP 250 nm RRAM devices, enables extrapolation to unmeasured conductance levels while preserving realistic state dependence. Consequently, VIM provides a means to generate state-aware variability parameters for system-level simulations even in the absence of comprehensive measurement data.

To implement this concept, VIM defines Gaussian distributions for each programmed conductance state, where the standard deviation scales as a function of the nominal conductance according to the extracted correlation. The scaling relation was derived from measurement data obtained at different conductance levels and operating temperatures, and subsequently extrapolated beyond the measured range. In this way, VIM captures the empirically observed dependence of variability on conductance state, providing a more realistic alternative to the use of fixed relative standard deviations. The generated variability parameter files are processed by the DStM, which supplies time-varying conductance values to the wave-digital simulation, ensuring that each memristor exhibits state-dependent stochastic behavior.

To facilitate practical use of the model, an abstract variability parameter Vx was introduced. It represents the normalized standard deviation (i.e., the coefficient of variation) at a baseline conductance of 10 μS , thereby providing a compact descriptor for temperature-dependent variability levels. Each variability level Vx defines a family of state-dependent standard deviation curves, which scale according to the empirical correlation established by the VIM. This formulation enables straightforward adjustment between realistic and worst-case operating conditions. For example, $V80$ corresponds to conditions that yield a relative variability of 80 % at a conductance of 10 μS .

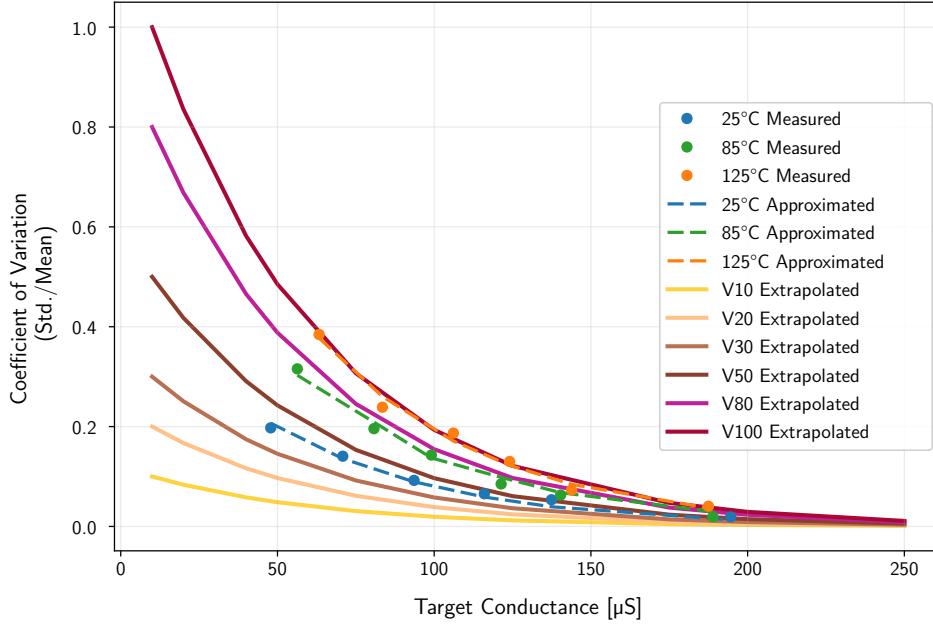


Figure 5.5.1: Coefficient of variation of RRAM conductance variability as a function of the target conductance. Adapted from [B11].

Figure 5.5.1 illustrates both the experimental analysis and the resulting variability model. The markers represent measured data points, showing the coefficient of variation (normalized standard deviation) of conductance across different target states and temperatures. The dashed curves indicate extrapolations of these measurements, while the solid lines correspond to the variability levels (V_{10} - V_{100}) derived from the Variability Intensity Model. These curves highlight the pronounced dependence of variability on conductance: lower conductance states exhibit substantially larger fluctuations compared to higher states.

Overall Simulation Flow The complete simulation flow for evaluating the variability robustness of the memristive analog neuron circuit is summarized in Figure 5.5.2. In the first step, state-dependent statistical distributions of the conductance variability are generated using the VIM, parameterized by the selected variability level Vx . These distributions are converted into parameter files for the DStM model, which provides time-dependent conductance values for each memristor during runtime. The resulting parameter sets are supplied to the wave-digital simulation of the Hindmarsh-Rose circuit, where they dynamically modulate the memristive device models.

After the simulation, the circuit dynamics are analyzed in the time domain to evaluate the neuronal behavior. The output signals are classified manually into three categories: spiking behavior, bursting behavior, or non-functional operation (no oscillatory dynamics). In addition to the variability tolerance simulations, deterministic tolerance analyses were performed by symmetrically varying component parameters around their nominal values to determine the parameter ranges that preserve the intended operating regime.

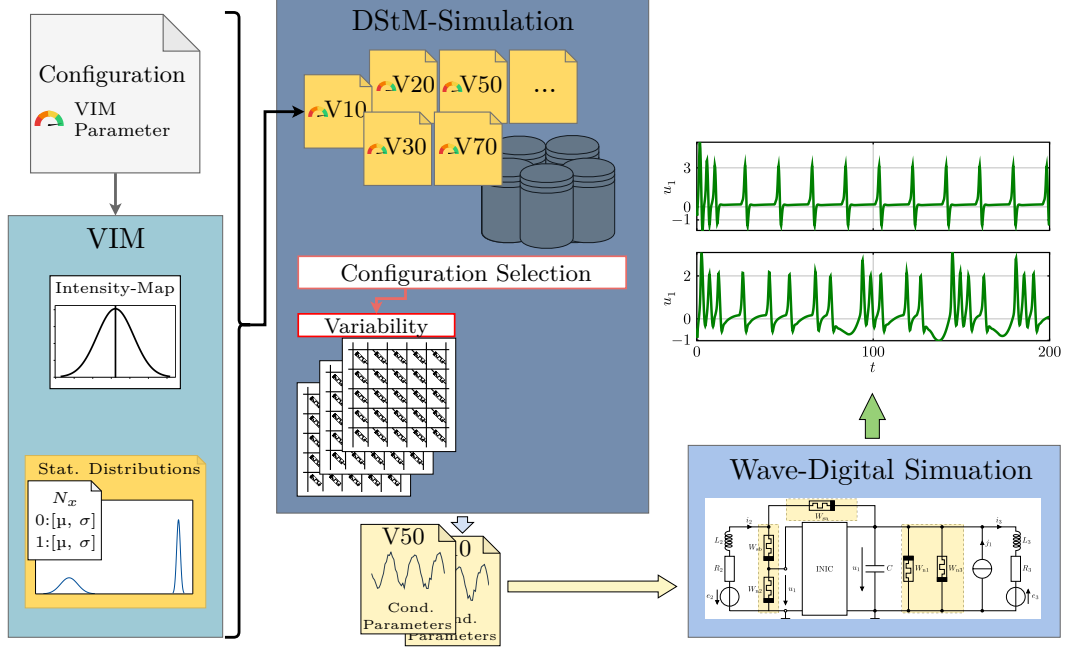


Figure 5.5.2: Simulation flow for the variability tolerance evaluation of the memristive analog neuron circuit.

Results

Two types of robustness analyses were performed: absolute parameter offsets and time-varying variability during operation.

Absolute parameter offsets

In the static tolerance analysis, component values were shifted symmetrically around their nominal points. The circuit was found to be largely insensitive to variations of the passive components, whereas the memristive parameters dominated the overall robustness. Spiking operation remained functional under conductance offsets of up to $\pm 80\%$ at low-conductance states, while bursting operation tolerated only about $\pm 40\%$ on average. At higher conductance states, robustness decreased, and correct operation was limited to offsets within $\pm 10\%$ to $\pm 20\%$.

Variability during operation

When state-dependent conductance fluctuations generated by the VIM method were introduced at every simulation step, the circuit continued to exhibit spiking and bursting dynamics within certain variability margins. Spiking was preserved under fluctuations of up to approximately $\pm 60\%$ at low-conductance states, whereas bursting became unstable already at around $\pm 30\%$. At higher conductance levels, the admissible variability window narrowed further, with functional operation restricted to $\pm 10\%$ to $\pm 15\%$.

These results highlight that spiking dynamics are substantially more robust than bursting, and that circuit tolerance strongly depends on the absolute conductance state. Moreover, they

demonstrate the importance of distinguishing between static parameter offsets and stochastic variability during operation, as both affect circuit robustness in complementary ways. Importantly, the observed higher tolerance at low conductance states and reduced tolerance at high conductance states aligns with the behavior of IHP 250 nm RRAM devices, in which low-conductance states exhibit stronger variability while high-conductance states are more stable.

Role of the Virtual Prototyping Platform

In contrast to the previous case studies, the complete MeSSi framework was not employed in this evaluation. Instead, the DStM modeling approach established within MeSSi, played a central role in developing the VIM for concrete RRAM devices. VIM leverages the DStM concept of state-dependent parameterization to generate realistic variability predictions for conductance states, thereby extending the scope of the modeling approach to scenarios where only limited measurement data are available.

This case study demonstrates how the measurement-driven variability modeling approach can be applied beyond memory and in-memory computing contexts to analog neuromorphic circuits. By relating device variability to the programmed conductance state rather than imposing a global statistical factor, VIM enabled a realistic representation of RRAM behavior in the memristive Hindmarsh-Rose neuron circuit.

The results show that spiking and bursting dynamics of the circuit are strongly shaped by memristive variability, with spiking proving more robust than bursting. Furthermore, the analysis also sets the baseline for device- and circuit designers to find matching devices that support the actual hardware implementation of the Hindmarsh-Rose model.

Overall, the study highlights that the proposed methodology can provide meaningful preliminary estimates of system behavior in the presence of device variability, even for circuits with complex nonlinear dynamics. This illustrates the generality of the modeling approach and underlines its relevance for evaluating memristive systems at different abstraction levels. Table 5.5.1 summarizes the platform features relevant to this case study.

Table 5.5.1: Relevant MeSSi features for this case study.

	Feature	Focus in this case study
	Multi-Level Storage	-
	Variability	Variability Intensity Model (VIM) for generating realistic, state-dependent variability distributions based on IHP 250 nm RRAM data.
	Devices & Operating Conditions	Evaluation of analog circuit design at discrete operating points.
	Modularity	-
	Scalability	-
	Interoperability	-
	Mapping Strategies	-
	Extra-Functional Properties	-
	Co-Simulation	-

Chapter 6

Conclusion and Outlook

This thesis addressed the central challenge of realistically and efficiently evaluating memristor-based computer architectures at the system level. While memristive technologies promise dense non-volatile memories, radiation-resilient storage, and in-memory computing accelerators, their adoption remains hindered by strong device variability, complex switching dynamics, and limited modeling support beyond circuit-level simulation. Bridging the gap between device-level behavior and architectural evaluation is therefore essential for technology-aware design. This work contributes a methodology and framework that enable such integration, emphasizing both accuracy and scalability.

6.1 Conclusion

The overarching aim of this thesis was to enable realistic and efficient system-level evaluation of memristor-based architectures by bridging the gap between device-level characterization and architectural exploration. Two main research goals were defined and pursued:

1. **Goal 1:** Develop and implement a computationally efficient, statistically accurate memristor model derived directly from empirical measurements, capturing device variability under diverse environmental and programming conditions.
2. **Goal 2:** Create a flexible, modular evaluation framework, combining a SystemC-based virtual prototyping platform with real-time FPGA IP blocks, that supports end-to-end co-simulation of memristor-based systems, from device behavior to complete architectures.

The following summarizes how these goals were addressed and what insights emerged from the conducted research.

Key Contributions and Insights

The first major contribution is the development of a variability-aware memristor model (DStM) that combines measurement-driven stochasticity with computational efficiency. By explicitly modeling time-dependent and time-invariant variability, the model achieves accuracy improvements of up to 84% compared to state-of-the-art system-level approaches, while

retaining a simulation speed two orders of magnitude faster. This directly addresses **Goal 1** by demonstrating that realistic device variability can be incorporated into large-scale simulations without prohibitive performance costs.

The second contribution is the design and implementation of the MeSSi virtual prototyping platform. Unlike monolithic simulators, MeSSi is structured as a modular, reusable framework that encapsulates memristor arrays, peripheral circuits, and controllers within a standardized SystemC interface. This design enabled flexible case studies across different architectural levels, from device-level registers and caches to in-memory computing modules. In doing so, it fulfills **Goal 2** of this thesis by creating a flexible, modular evaluation framework that links device modeling with complete digital architectures.

A third contribution lies in the evaluation of representative use cases. Through case studies such as radiation-induced reliability degradation, RRAM-based cache and register file analysis, and memory subsystem energy-performance trade-offs, this thesis demonstrated that variability-aware modeling fundamentally alters the conclusions drawn at the system level. Protection strategies that appeared effective under idealized models were shown to fail when realistic device distributions and correlations were considered, underscoring the necessity of variability integration in design space exploration. Among these use cases, the thesis presented a proof-of-concept for FPGA-based acceleration of memristor simulations. By off-loading the stochastic state transition dynamics to hardware, simulation bottlenecks were alleviated, and scalability for large workloads was demonstrated. This validates the efficiency aspect of the proposed methodology and highlights its extensibility to larger system scales.

Taken together, these contributions illustrate how system-level evaluation of memristor-based architectures can be made both realistic and efficient, providing new methodological foundations for emerging device research.

Broader Contextualization

Beyond its immediate technical results, this work contributes to a broader scientific and methodological context. In the field of neuromorphic and in-memory computing, where non-volatile memories are envisioned as enablers of energy-efficient architectures, accurate system-level assessment is a prerequisite for technology transfer. Conventional SPICE-level simulations offer accuracy but lack scalability, whereas purely behavioral models capture architectural trends but neglect physical limitations. The approach developed here establishes a middle ground: models grounded in measurement yet abstracted for architectural exploration.

Methodologically, this thesis advances the idea of *variability-aware virtual prototyping*. While demonstrated for memristors, the framework could be extended to other emerging device classes characterized by strong stochasticity, such as spintronic or ferroelectric memories. In this sense, the work points to an interdisciplinary methodology: integrating device physics, statistical modeling, and system-level design within a single evaluation flow.

Limitations

The results presented here are subject to several limitations. The quality of the variability-aware models depends on the availability and scope of measurement data. Rare failure modes,

long-term degradation, and correlated effects are difficult to capture through measurements. Certain continuous-time analog phenomena, such as pulse-shape dependence, sneak-path currents, or thermal coupling, were abstracted in favor of scalability. Finally, the FPGA-based acceleration was developed as a prototype and abstracted memory accesses broadly. These limitations do not diminish the validity of the presented contributions but highlight areas for refinement.

6.2 Outlook and Future Work

Building on these foundations, several directions for future research can be identified. In terms of **applications**, the MeSSi platform opens opportunities beyond conventional memory studies or analog in-memory computing. Promising directions include the evaluation of security primitives such as physically unclonable functions (PUFs), memristor-based logic, or computing platforms with low duty cycles. Safety-critical use cases, for instance in hospitals or military electronics, where information must persist even under sudden power loss, constitute another relevant research area. Moreover, memristor-based content-addressable memories represent a novel design space for efficient search and retrieval tasks.

With respect to the **MeSSi platform itself**, future work could include the incorporation of additional memristor models with diverse physical properties, which would enable cross-technology comparisons and strengthen the generality of the framework. Likewise, developing mixed-abstraction approaches that switch simulation granularity dynamically at runtime would allow balancing accuracy and performance according to experimental needs. Coupling MeSSi with hardware-in-the-loop setups or integrating it into widely used frameworks such as gem5 or NVMain would further broaden its applicability, while FPGA-based acceleration could be scaled towards full-system simulations to address performance bottlenecks.

Finally, in a **broader context**, variability-aware virtual prototyping can be generalized as a methodological paradigm for emerging technologies in which stochastic effects play a decisive role, extending the impact of this work beyond memristors.

Closing Statement

In conclusion, this thesis has developed and validated a methodology that bridges device-level stochasticity and system-level evaluation of memristive architectures. By uniting measurement-driven modeling, modular virtual prototyping, and hardware evaluation, it lays the foundation for a technology-aware design flow that is both accurate and efficient. While challenges remain, the trajectory set by this work points toward a future where variability is not an obstacle to memristor adoption, but a design parameter to be quantified, understood, and ultimately exploited. In this sense, the thesis has made concrete contributions to the efficient evaluation of memristor-based computer architectures and to bridging the gap between devices and systems.

Appendix A

Statement on Writing Assistance

Given that English is not my first language, AI-based language tools, including grammar and writing assistants as well as Large Language Models, were used to improve the clarity, readability, and overall quality of the English formulation throughout the manuscript. All content, ideas, analyses, and conclusions presented in this thesis are entirely my own (unless explicitly stated otherwise). Specifically, the tools DeepL Translator^{A1}, DeepL Write^{A2}, and ChatGPT^{A3}, in their respective online versions available between October 2024 and October 2025, were utilized.

A1 <https://www.deepl.com/en/translator>

A2 <https://www.deepl.com/en/write>

A3 <https://chatgpt.com/>

Appendix B

Bibliography

- [A1] O. Aouedi et al., “A Survey on Intelligent Internet of Things: Applications, Security, Privacy, and Future Directions,” *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024. DOI: [10.1109/comst.2024.3430368](https://doi.org/10.1109/comst.2024.3430368).
- [A2] “Boosting AI with neuromorphic computing,” *Nature Computational Science*, vol. 5, no. 1, pp. 1–2, Jan. 2025, ISSN: 2662-8457. DOI: [10.1038/s43588-025-00770-4](https://doi.org/10.1038/s43588-025-00770-4).
- [A3] T. Sipola, J. Alatalo, T. Kokkonen, and M. Rantonen, “Artificial Intelligence in the IoT Era: A Review of Edge AI Hardware and Software,” in *2022 31st Conference of Open Innovations Association (FRUCT)*, IEEE, Apr. 2022, pp. 320–331. DOI: [10.23919/fruct54823.2022.9770931](https://doi.org/10.23919/fruct54823.2022.9770931).
- [A4] Epoch AI, *Data on Machine Learning Hardware*, [Accessed: 28.04.2025], Oct. 2024. [Online]. Available: <https://epoch.ai/data/machine-learning-hardware>.
- [A5] M. Verhelst, L. Benini, and N. Verma, “How to Keep Pushing ML Accelerator Performance? Know Your Rooflines!” *IEEE Journal of Solid-State Circuits*, pp. 1–18, 2025, ISSN: 1558-173X. DOI: [10.1109/jssc.2025.3553765](https://doi.org/10.1109/jssc.2025.3553765).
- [A6] Y.-J. Feng et al., “Accelerating Data Transfer in Dataflow Architectures Through a Look-Ahead Acknowledgment Mechanism,” *Journal of Computer Science and Technology*, vol. 37, no. 4, pp. 942–959, Jul. 2022, ISSN: 1860-4749. DOI: [10.1007/s11390-020-0555-6](https://doi.org/10.1007/s11390-020-0555-6).
- [A7] A. A. Khan, J. P. C. De Lima, H. Farzaneh, and J. Castrillon, “The Landscape of Compute-near-memory and Compute-in-memory: A Research and Commercial Overview,” 2024. DOI: [10.48550/arxiv.2401.14428](https://doi.org/10.48550/arxiv.2401.14428).
- [A8] T. Nakada and H. Nakamura, “Normally-Off Computing,” in *Normally-Off Computing*. Springer Japan, 2017, pp. 57–63, ISBN: 9784431565055. DOI: [10.1007/978-4-431-56505-5_4](https://doi.org/10.1007/978-4-431-56505-5_4).
- [A9] I. H. Im, S. J. Kim, and H. W. Jang, “Memristive Devices for New Computing Paradigms,” *Advanced Intelligent Systems*, vol. 2, no. 11, Aug. 2020, ISSN: 2640-4567. DOI: [10.1002/aisy.202000105](https://doi.org/10.1002/aisy.202000105).
- [A10] L. Chua, “Memristor - The missing circuit element,” *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971, ISSN: 0018-9324. DOI: [10.1109/tct.1971.1083337](https://doi.org/10.1109/tct.1971.1083337).
- [A11] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, “‘Memristive’ switches enable ‘stateful’ logic operations via material implication,”

- Nature*, vol. 464, no. 7290, pp. 873–876, Apr. 2010, ISSN: 1476-4687. DOI: [10.1038/nature08940](https://doi.org/10.1038/nature08940).
- [A12] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, “Memory devices and applications for in-memory computing,” *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, Mar. 2020, ISSN: 1748-3395. DOI: [10.1038/s41565-020-0655-z](https://doi.org/10.1038/s41565-020-0655-z).
- [A13] Y. Huang, T. Ando, A. Sebastian, M.-F. Chang, J. J. Yang, and Q. Xia, “Memristor-based hardware accelerators for artificial intelligence,” *Nature Reviews Electrical Engineering*, vol. 1, no. 5, pp. 286–299, Apr. 2024, ISSN: 2948-1201. DOI: [10.1038/s44287-024-00037-6](https://doi.org/10.1038/s44287-024-00037-6).
- [A14] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008, ISSN: 1476-4687. DOI: [10.1038/nature06932](https://doi.org/10.1038/nature06932).
- [A15] S. Vongehr and X. Meng, “The Missing Memristor has Not been Found,” *Scientific Reports*, vol. 5, no. 1, Jun. 2015, ISSN: 2045-2322. DOI: [10.1038/srep11657](https://doi.org/10.1038/srep11657).
- [A16] I. Abraham, “The case for rejecting the memristor as a fundamental circuit element,” *Scientific Reports*, vol. 8, no. 1, Jul. 2018, ISSN: 2045-2322. DOI: [10.1038/s41598-018-29394-7](https://doi.org/10.1038/s41598-018-29394-7).
- [A17] L. Chua and S. M. Kang, “Memristive devices and systems,” *Proceedings of the IEEE*, vol. 64, no. 2, pp. 209–223, 1976, ISSN: 0018-9219. DOI: [10.1109/proc.1976.10092](https://doi.org/10.1109/proc.1976.10092).
- [A18] F. Marrone et al., “Experimental validation of state equations and dynamic route maps for phase change memristive devices,” *Scientific Reports*, vol. 12, no. 1, Apr. 2022, ISSN: 2045-2322. DOI: [10.1038/s41598-022-09948-6](https://doi.org/10.1038/s41598-022-09948-6).
- [A19] V. Garcia and M. Bibes, “Ferroelectric tunnel junctions for information storage and processing,” *Nature Communications*, vol. 5, no. 1, Jul. 2014, ISSN: 2041-1723. DOI: [10.1038/ncomms5289](https://doi.org/10.1038/ncomms5289).
- [A20] L. Chua, “Resistance switching memories are memristors,” *Applied Physics A*, vol. 102, no. 4, pp. 765–783, Jan. 2011, ISSN: 1432-0630. DOI: [10.1007/s00339-011-6264-9](https://doi.org/10.1007/s00339-011-6264-9).
- [A21] Z. Wang et al., “Resistive switching materials for information processing,” *Nature Reviews Materials*, vol. 5, no. 3, pp. 173–195, Jan. 2020, ISSN: 2058-8437. DOI: [10.1038/s41578-019-0159-3](https://doi.org/10.1038/s41578-019-0159-3).
- [A22] Y. Xiao et al., “A review of memristor: material and structure design, device performance, applications and prospects,” *Science and Technology of Advanced Materials*, vol. 24, no. 1, Feb. 2023, ISSN: 1878-5514. DOI: [10.1080/14686996.2022.2162323](https://doi.org/10.1080/14686996.2022.2162323).
- [A23] A. Mehonic, A. Sebastian, B. Rajendran, O. Simeone, E. Vasilaki, and A. J. Kenyon, “Memristors—From In-Memory Computing, Deep Learning Acceleration, and Spiking Neural Networks to the Future of Neuromorphic and Bio-Inspired Computing,” *Advanced Intelligent Systems*, vol. 2, no. 11, Aug. 2020, ISSN: 2640-4567. DOI: [10.1002/aisy.202000085](https://doi.org/10.1002/aisy.202000085).
- [A24] M. Lanza and H. Tian, “Materials for Memristors,” *Advanced Functional Materials*, vol. 34, no. 15, Dec. 2023, ISSN: 1616-3028. DOI: [10.1002/adfm.202314512](https://doi.org/10.1002/adfm.202314512).
- [A25] M.-K. Song et al., “Recent Advances and Future Prospects for Memristive Materials, Devices, and Systems,” *ACS Nano*, vol. 17, no. 13, pp. 11 994–12 039, Jun. 2023, ISSN: 1936-086X. DOI: [10.1021/acsnano.3c03505](https://doi.org/10.1021/acsnano.3c03505).
- [A26] F. Qin, Y. Zhang, H. W. Song, and S. Lee, “Enhancing memristor fundamentals through instrumental characterization and understanding reliability issues,” *Mate-*

- rials Advances*, vol. 4, no. 8, pp. 1850–1875, 2023, ISSN: 2633-5409. DOI: [10.1039/d3ma00069a](https://doi.org/10.1039/d3ma00069a).
- [A27] L. Gao, Q. Ren, J. Sun, S.-T. Han, and Y. Zhou, “Memristor modeling: challenges in theories, simulations, and device variability,” *Journal of Materials Chemistry C*, vol. 9, no. 47, pp. 16 859–16 884, 2021, ISSN: 2050-7534. DOI: [10.1039/d1tc04201g](https://doi.org/10.1039/d1tc04201g).
 - [A28] G. C. Adam, A. Khiat, and T. Prodromakis, “Challenges hindering memristive neuromorphic hardware from going mainstream,” *Nature Communications*, vol. 9, no. 1, Dec. 2018, ISSN: 2041-1723. DOI: [10.1038/s41467-018-07565-4](https://doi.org/10.1038/s41467-018-07565-4).
 - [A29] W. Sun et al., “Understanding memristive switching via in situ characterization and device modeling,” *Nature Communications*, vol. 10, no. 1, Aug. 2019, ISSN: 2041-1723. DOI: [10.1038/s41467-019-11411-6](https://doi.org/10.1038/s41467-019-11411-6).
 - [A30] M. Fritscher et al., “Mitigating the Effects of RRAM Process Variation on the Accuracy of Artificial Neural Networks,” in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer International Publishing, 2022, pp. 401–417, ISBN: 9783031045806. DOI: [10.1007/978-3-031-04580-6_27](https://doi.org/10.1007/978-3-031-04580-6_27).
 - [A31] M. Fritscher et al., “Prototyping Reconfigurable RRAM-Based AI Accelerators Using the RISC-V Ecosystem and Digital Twins,” in *High Performance Computing*. Springer Nature Switzerland, 2023, pp. 500–514, ISBN: 9783031408434. DOI: [10.1007/978-3-031-40843-4_37](https://doi.org/10.1007/978-3-031-40843-4_37).
 - [A32] M. Fritscher et al., “A flexible and fast digital twin for RRAM systems applied for training resilient neural networks,” *Scientific Reports*, vol. 14, no. 1, Oct. 2024, ISSN: 2045-2322. DOI: [10.1038/s41598-024-73439-z](https://doi.org/10.1038/s41598-024-73439-z).
 - [A33] C. Lammie, W. Xiang, and M. Rahimi Azghadi, “Modeling and simulating in-memory memristive deep learning systems: An overview of current efforts,” *Array*, vol. 13, p. 100 116, Mar. 2022, ISSN: 2590-0056. DOI: [10.1016/j.array.2021.100116](https://doi.org/10.1016/j.array.2021.100116).
 - [A34] P.-Y. Chen, X. Peng, and S. Yu, “NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067–3080, 2018. DOI: [10.1109/tcad.2018.2789723](https://doi.org/10.1109/tcad.2018.2789723).
 - [A35] A. Lu, X. Peng, W. Li, H. Jiang, and S. Yu, “NeuroSim Simulator for Compute-in-Memory Hardware Accelerator: Validation and Benchmark,” *Frontiers in Artificial Intelligence*, vol. 4, Jun. 2021, ISSN: 2624-8212. DOI: [10.3389/frai.2021.659060](https://doi.org/10.3389/frai.2021.659060).
 - [A36] Y. Zhao, R. Chen, P. Huang, and J. Kang, “Modeling-Based Design of Memristive Devices for Brain-Inspired Computing,” *Frontiers in Nanotechnology*, vol. 3, Apr. 2021, ISSN: 2673-3013. DOI: [10.3389/fnano.2021.654418](https://doi.org/10.3389/fnano.2021.654418).
 - [A37] S. H. H. Shadmehri et al., “SySCIM: SystemC-AMS Simulation of Memristive Computation In-Memory,” in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1467–1472. DOI: [10.23919/date54114.2022.9774749](https://doi.org/10.23919/date54114.2022.9774749).
 - [A38] C. Lammie, W. Xiang, B. Linares-Barranco, and M. Rahimi Azghadi, “MemTorch: An Open-source Simulation Framework for Memristive Deep Learning Systems,” *Neurocomputing*, vol. 485, pp. 124–133, May 2022, ISSN: 0925-2312. DOI: [10.1016/j.neucom.2022.02.043](https://doi.org/10.1016/j.neucom.2022.02.043).
 - [A39] J. Lee, A. Lu, W. Li, and S. Yu, “NeuroSim V1.4: Extending Technology Support for Digital Compute-in-Memory Toward 1nm Node,” *IEEE Transactions on Circuits*

- and Systems I: Regular Papers*, vol. 71, no. 4, pp. 1733–1744, Apr. 2024, ISSN: 1558-0806. DOI: [10.1109/tcsi.2024.3362822](https://doi.org/10.1109/tcsi.2024.3362822).
- [A40] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, “NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 994–1007, Jul. 2012, ISSN: 1937-4151. DOI: [10.1109/tcad.2012.2185930](https://doi.org/10.1109/tcad.2012.2185930).
- [A41] J. S. Meena, S. M. Sze, U. Chand, and T.-Y. Tseng, “Overview of emerging non-volatile memory technologies,” *Nanoscale Research Letters*, vol. 9, no. 1, Sep. 2014, ISSN: 1556-276X. DOI: [10.1186/1556-276x-9-526](https://doi.org/10.1186/1556-276x-9-526).
- [A42] U. Sharma, G. Kumar, S. Mishra, and R. Thomas, “Ferroelectric tunnel junctions: current status and future prospect as a universal memory,” *Frontiers in Materials*, vol. 10, Sep. 2023, ISSN: 2296-8016. DOI: [10.3389/fmats.2023.1148979](https://doi.org/10.3389/fmats.2023.1148979).
- [A43] T. Mikolajick et al., “Next generation ferroelectric materials for semiconductor process integration and their applications,” *Journal of Applied Physics*, vol. 129, no. 10, Mar. 2021, ISSN: 1089-7550. DOI: [10.1063/5.0037617](https://doi.org/10.1063/5.0037617).
- [A44] N. Zagni, F. M. Puglisi, P. Pavan, and M. A. Alam, “Reliability of HfO₂-Based Ferroelectric FETs: A Critical Review of Current and Future Challenges,” *Proceedings of the IEEE*, vol. 111, no. 2, pp. 158–184, Feb. 2023, ISSN: 1558-2256. DOI: [10.1109/jproc.2023.3234607](https://doi.org/10.1109/jproc.2023.3234607).
- [A45] S. S. P. Parkin, M. Hayashi, and L. Thomas, “Magnetic Domain-Wall Racetrack Memory,” *Science*, vol. 320, no. 5873, pp. 190–194, Apr. 2008, ISSN: 1095-9203. DOI: [10.1126/science.1145799](https://doi.org/10.1126/science.1145799).
- [A46] B. Li, S. Zhang, L. Xu, Q. Su, and B. Du, “Emerging Robust Polymer Materials for High-Performance Two-Terminal Resistive Switching Memory,” *Polymers*, vol. 15, no. 22, p. 4374, Nov. 2023, ISSN: 2073-4360. DOI: [10.3390/polym15224374](https://doi.org/10.3390/polym15224374).
- [A47] D. Xiang, X. Wang, C. Jia, T. Lee, and X. Guo, “Molecular-Scale Electronics: From Concept to Function,” *Chemical Reviews*, vol. 116, no. 7, pp. 4318–4440, Mar. 2016, ISSN: 1520-6890. DOI: [10.1021/acs.chemrev.5b00680](https://doi.org/10.1021/acs.chemrev.5b00680).
- [A48] W. Banerjee, “Challenges and Applications of Emerging Nonvolatile Memory Devices,” *Electronics*, vol. 9, no. 6, p. 1029, Jun. 2020, ISSN: 2079-9292. DOI: [10.3390/electronics9061029](https://doi.org/10.3390/electronics9061029).
- [A49] F. Ummelen, H. Swagten, and B. Koopmans, “Racetrack memory based on in-plane-field controlled domain-wall pinning,” *Scientific Reports*, vol. 7, no. 1, Apr. 2017, ISSN: 2045-2322. DOI: [10.1038/s41598-017-00837-x](https://doi.org/10.1038/s41598-017-00837-x).
- [A50] S. Lee, S. Kim, and H. Yoo, “Contribution of Polymers to Electronic Memory Devices and Applications,” *Polymers*, vol. 13, no. 21, p. 3774, Oct. 2021, ISSN: 2073-4360. DOI: [10.3390/polym13213774](https://doi.org/10.3390/polym13213774).
- [A51] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Upper Saddle River, NJ: Pearson, Mar. 2010, ISBN: 0321547748.
- [A52] Q. Cao et al., “Nonvolatile Multistates Memories for High-Density Data Storage,” *ACS Applied Materials & Interfaces*, vol. 12, no. 38, pp. 42 449–42 471, Aug. 2020, ISSN: 1944-8252. DOI: [10.1021/acsami.0c10184](https://doi.org/10.1021/acsami.0c10184).
- [A53] A. D. Kent and D. C. Worledge, “A new spin on magnetic memories,” *Nature Nanotechnology*, vol. 10, no. 3, pp. 187–191, Mar. 2015, ISSN: 1748-3395. DOI: [10.1038/nnano.2015.24](https://doi.org/10.1038/nnano.2015.24).

-
- [A54] F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, “Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multi-level Cell (mlc) Storage, Modeling, and Applications,” *Nanoscale Research Letters*, vol. 15, no. 1, Apr. 2020, ISSN: 1556-276X. DOI: [10.1186/s11671-020-03299-9](https://doi.org/10.1186/s11671-020-03299-9).
 - [A55] V. Deshpande, K. S. Nair, M. Holzer, S. Banerjee, and C. Dubourdieu, “CMOS back-end-of-line compatible ferroelectric tunnel junction devices,” *Solid-State Electronics*, vol. 186, p. 108054, Dec. 2021, ISSN: 0038-1101. DOI: [10.1016/j.sse.2021.108054](https://doi.org/10.1016/j.sse.2021.108054).
 - [A56] A. Mehonic et al., “Roadmap to neuromorphic computing with emerging technologies,” *APL Materials*, vol. 12, no. 10, Oct. 2024, ISSN: 2166-532X. DOI: [10.1063/5.0179424](https://doi.org/10.1063/5.0179424).
 - [A57] O. Zilberberg, S. Weiss, and S. Toledo, “Phase-change memory: An architectural perspective,” *ACM Computing Surveys*, vol. 45, no. 3, pp. 1–33, Jun. 2013, ISSN: 1557-7341. DOI: [10.1145/2480741.2480746](https://doi.org/10.1145/2480741.2480746).
 - [A58] G. Panagopoulos, C. Augustine, X. Fong, and K. Roy, “Exploring variability and reliability of multi-level STT-MRAM cells,” in *70th Device Research Conference*, IEEE, Jun. 2012, pp. 139–140. DOI: [10.1109/drc.2012.6257003](https://doi.org/10.1109/drc.2012.6257003).
 - [A59] S. H. Park, H. J. Lee, M. H. Park, J. Kim, and H. W. Jang, “Ferroelectric tunnel junctions: promise, achievements and challenges,” *Journal of Physics D: Applied Physics*, vol. 57, no. 25, p. 253002, Mar. 2024. DOI: [10.1088/1361-6463/ad33f5](https://doi.org/10.1088/1361-6463/ad33f5).
 - [A60] D. Edelstein et al., “A 14 nm Embedded STT-MRAM CMOS Technology,” in *2020 IEEE International Electron Devices Meeting (IEDM)*, IEEE, Dec. 2020. DOI: [10.1109/iedm13553.2020.9371922](https://doi.org/10.1109/iedm13553.2020.9371922).
 - [A61] J. B. Roldán et al., “Variability in Resistive Memories,” *Advanced Intelligent Systems*, vol. 5, no. 6, Mar. 2023, ISSN: 2640-4567. DOI: [10.1002/aisy.202200338](https://doi.org/10.1002/aisy.202200338).
 - [A62] L. Xia et al., “Stuck-at Fault Tolerance in RRAM Computing Systems,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 102–115, Mar. 2018, ISSN: 2156-3365. DOI: [10.1109/jetcas.2017.2776980](https://doi.org/10.1109/jetcas.2017.2776980).
 - [A63] J. Woo, K. Moon, J. Song, M. Kwak, J. Park, and H. Hwang, “Optimized Programming Scheme Enabling Linear Potentiation in Filamentary HfO₂ RRAM Synapse for Neuromorphic Systems,” *IEEE Transactions on Electron Devices*, vol. 63, no. 12, pp. 5064–5067, Dec. 2016, ISSN: 1557-9646. DOI: [10.1109/ted.2016.2615648](https://doi.org/10.1109/ted.2016.2615648).
 - [A64] P. Narayanan et al., “Fully On-Chip MAC at 14 nm Enabled by Accurate Row-Wise Programming of PCM-Based Weights and Parallel Vector-Transport in Duration-Format,” *IEEE Transactions on Electron Devices*, vol. 68, no. 12, pp. 6629–6636, Dec. 2021, ISSN: 1557-9646. DOI: [10.1109/ted.2021.3115993](https://doi.org/10.1109/ted.2021.3115993).
 - [A65] H. Zhao et al., “Energy-efficient high-fidelity image reconstruction with memristor arrays for medical diagnosis,” *Nature Communications*, vol. 14, no. 1, Apr. 2023, ISSN: 2041-1723. DOI: [10.1038/s41467-023-38021-7](https://doi.org/10.1038/s41467-023-38021-7).
 - [A66] H. Bao et al., “Toward memristive in-memory computing: principles and applications,” *Frontiers of Optoelectronics*, vol. 15, no. 1, May 2022, ISSN: 2095-2767. DOI: [10.1007/s12200-022-00025-4](https://doi.org/10.1007/s12200-022-00025-4).
 - [A67] F. Xia, D.-J. Jiang, J. Xiong, and N.-H. Sun, “A Survey of Phase Change Memory Systems,” *Journal of Computer Science and Technology*, vol. 30, no. 1, pp. 121–144, Jan. 2015, ISSN: 1860-4749. DOI: [10.1007/s11390-015-1509-2](https://doi.org/10.1007/s11390-015-1509-2).

- [A68] W. Chen et al., “Essential Characteristics of Memristors for Neuromorphic Computing,” *Advanced Electronic Materials*, vol. 9, no. 2, Oct. 2022, ISSN: 2199-160X. DOI: [10.1002/aelm.202200833](https://doi.org/10.1002/aelm.202200833).
- [A69] D. Zhu, Y. Li, W. Shen, Z. Zhou, L. Liu, and X. Zhang, “Resistive random access memory and its applications in storage and nonvolatile logic,” *Journal of Semiconductors*, vol. 38, no. 7, p. 071 002, Jul. 2017. DOI: [10.1088/1674-4926/38/7/071002](https://doi.org/10.1088/1674-4926/38/7/071002).
- [A70] V. Gupta, S. Kapur, S. Saurabh, and A. Grover, “Resistive Random Access Memory: A Review of Device Challenges,” *IETE Technical Review*, vol. 37, no. 4, pp. 377–390, Jun. 2019, ISSN: 0974-5971. DOI: [10.1080/02564602.2019.1629341](https://doi.org/10.1080/02564602.2019.1629341).
- [A71] W. Zhao et al., “Failure Analysis in Magnetic Tunnel Junction Nanopillar with Interfacial Perpendicular Magnetic Anisotropy,” *Materials*, vol. 9, no. 1, p. 41, Jan. 2016, ISSN: 1996-1944. DOI: [10.3390/ma9010041](https://doi.org/10.3390/ma9010041).
- [A72] V. Sverdlov and S. Selberherr, “Spin-Based Devices for Digital Applications,” in *Springer Handbook of Semiconductor Devices*. Springer International Publishing, Nov. 2022, pp. 1123–1166, ISBN: 9783030798277. DOI: [10.1007/978-3-030-79827-7_31](https://doi.org/10.1007/978-3-030-79827-7_31).
- [A73] P. Krzysteczko, J. Münchenberger, M. Schäfers, G. Reiss, and A. Thomas, “The Memristive Magnetic Tunnel Junction as a Nanoscopic Synapse-Neuron System,” *Advanced Materials*, vol. 24, no. 6, pp. 762–766, Jan. 2012, ISSN: 1521-4095. DOI: [10.1002/adma.201103723](https://doi.org/10.1002/adma.201103723).
- [A74] S. Deng et al., “Overview of Ferroelectric Memory Devices and Reliability Aware Design Optimization,” in *Proceedings of the 2021 Great Lakes Symposium on VLSI*, ser. GLSVLSI ’21, Virtual Event, USA: Association for Computing Machinery, 2021, pp. 473–478, ISBN: 9781450383936. DOI: [10.1145/3453688.3461743](https://doi.org/10.1145/3453688.3461743).
- [A75] T. Mikolajick, U. Schroeder, and S. Slesazek, “The Case for Ferroelectrics in Future Memory Devices,” in *2021 5th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*, 2021, pp. 1–3. DOI: [10.1109/edtm50988.2021.9420821](https://doi.org/10.1109/edtm50988.2021.9420821).
- [A76] M. Tarkov, F. Tikhonenko, V. Popov, V. Antonov, A. Miakonkikh, and K. Rudenko, “Ferroelectric Devices for Content-Addressable Memory,” *Nanomaterials*, vol. 12, no. 24, p. 4488, Dec. 2022, ISSN: 2079-4991. DOI: [10.3390/nano12244488](https://doi.org/10.3390/nano12244488).
- [A77] M. L. Gallo and A. Sebastian, “An overview of phase-change memory device physics,” *Journal of Physics D: Applied Physics*, vol. 53, no. 21, p. 213 002, Mar. 2020. DOI: [10.1088/1361-6463/ab7794](https://doi.org/10.1088/1361-6463/ab7794).
- [A78] F. Ambriz-Vargas et al., “A Complementary Metal Oxide Semiconductor Process-Compatible Ferroelectric Tunnel Junction,” *ACS Applied Materials & Interfaces*, vol. 9, no. 15, pp. 13 262–13 268, Apr. 2017, ISSN: 1944-8252. DOI: [10.1021/acsami.6b16173](https://doi.org/10.1021/acsami.6b16173).
- [A79] E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga, “Analog Memristive Synapse in Spiking Networks Implementing Unsupervised Learning,” *Frontiers in Neuroscience*, vol. 10, Oct. 2016, ISSN: 1662-453X. DOI: [10.3389/fnins.2016.00482](https://doi.org/10.3389/fnins.2016.00482).
- [A80] I. Barraij, H. Mestiri, and M. Masmoudi, “Overview of Memristor-Based Design for Analog Applications,” *Micromachines*, vol. 15, no. 4, p. 505, Apr. 2024, ISSN: 2072-666X. DOI: [10.3390/mi15040505](https://doi.org/10.3390/mi15040505).

-
- [A81] S. Pechmann et al., “A Versatile, Voltage-Pulse Based Read and Programming Circuit for Multi-Level RRAM Cells,” *Electronics*, vol. 10, no. 5, p. 530, Feb. 2021, ISSN: 2079-9292. DOI: [10.3390/electronics10050530](https://doi.org/10.3390/electronics10050530).
- [A82] N. Papandreou et al., “Multilevel phase-change memory,” in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*, IEEE, Dec. 2010, pp. 1017–1020. DOI: [10.1109/icecs.2010.5724687](https://doi.org/10.1109/icecs.2010.5724687).
- [A83] R. Fontanini et al., “Modeling and Design of FTJs as Multi-Level Low Energy Memristors for Neuromorphic Computing,” *IEEE Journal of the Electron Devices Society*, vol. 9, pp. 1202–1209, 2021, ISSN: 2168-6734. DOI: [10.1109/jeds.2021.3120200](https://doi.org/10.1109/jeds.2021.3120200).
- [A84] D. Ielmini and G. Pedretti, “Resistive Switching Random-Access Memory (RRAM): Applications and Requirements for Memory and Computing,” *Chemical Reviews*, vol. 125, no. 12, pp. 5584–5625, May 2025, ISSN: 1520-6890. DOI: [10.1021/acs.chemrev.4c00845](https://doi.org/10.1021/acs.chemrev.4c00845).
- [A85] A. Amirsoleimani et al., “In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal-Oxide-Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives,” *Advanced Intelligent Systems*, vol. 2, no. 11, Aug. 2020, ISSN: 2640-4567. DOI: [10.1002/aisy.202000115](https://doi.org/10.1002/aisy.202000115).
- [A86] Y. Sun, S. Wang, Q. Zhang, and P. Zhou, “Evaluation of different 2D memory technologies for in-memory computing,” *Device*, vol. 2, no. 12, p. 100 509, Dec. 2024, ISSN: 2666-9986. DOI: [10.1016/j.device.2024.100509](https://doi.org/10.1016/j.device.2024.100509).
- [A87] S. Kvatinsky et al., “MAGIC—Memristor-Aided Logic,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014, ISSN: 1558-3791. DOI: [10.1109/tcsii.2014.2357292](https://doi.org/10.1109/tcsii.2014.2357292).
- [A88] Z. Sun, E. Ambrosi, A. Bricalli, and D. Ielmini, “Logic Computing with Stateful Neural Networks of Resistive Switches,” *Advanced Materials*, vol. 30, no. 38, Aug. 2018, ISSN: 1521-4095. DOI: [10.1002/adma.201802554](https://doi.org/10.1002/adma.201802554).
- [A89] A. Chen, “Utilizing the Variability of Resistive Random Access Memory to Implement Reconfigurable Physical Unclonable Functions,” *IEEE Electron Device Letters*, vol. 36, no. 2, pp. 138–140, Feb. 2015, ISSN: 1558-0563. DOI: [10.1109/led.2014.2385870](https://doi.org/10.1109/led.2014.2385870).
- [A90] R. Liu, H. Wu, Y. Pang, H. Qian, and S. Yu, “Experimental Characterization of Physical Unclonable Function Based on 1 kb Resistive Random Access Memory Arrays,” *IEEE Electron Device Letters*, vol. 36, no. 12, pp. 1380–1383, Dec. 2015, ISSN: 1558-0563. DOI: [10.1109/led.2015.2496257](https://doi.org/10.1109/led.2015.2496257).
- [A91] Y. Pang et al., “Optimization of RRAM-Based Physical Unclonable Function With a Novel Differential Read-Out Method,” *IEEE Electron Device Letters*, vol. 38, no. 2, pp. 168–171, Feb. 2017, ISSN: 1558-0563. DOI: [10.1109/led.2016.2647230](https://doi.org/10.1109/led.2016.2647230).
- [A92] C.-Y. Huang, W. C. Shen, Y.-H. Tseng, Y.-C. King, and C.-J. Lin, “A Contact-Resistive Random-Access-Memory-Based True Random Number Generator,” *IEEE Electron Device Letters*, vol. 33, no. 8, pp. 1108–1110, Aug. 2012, ISSN: 1558-0563. DOI: [10.1109/led.2012.2199734](https://doi.org/10.1109/led.2012.2199734).
- [A93] S. Balatti et al., “Physical Unbiased Generation of Random Numbers With Coupled Resistive Switching Devices,” *IEEE Transactions on Electron Devices*, vol. 63, no. 5, pp. 2029–2035, May 2016, ISSN: 1557-9646. DOI: [10.1109/ted.2016.2537792](https://doi.org/10.1109/ted.2016.2537792).

-
- [A94] W. Wang et al., “Learning of spatiotemporal patterns in a spiking neural network with resistive switching synapses,” *Science Advances*, vol. 4, no. 9, Sep. 2018, ISSN: 2375-2548. DOI: [10.1126/sciadv.aat4752](https://doi.org/10.1126/sciadv.aat4752).
 - [A95] B. Yan et al., “RRAM-based Spiking Nonvolatile Computing-In-Memory Processing Engine with Precision-Configurable In Situ Nonlinear Activation,” in *2019 Symposium on VLSI Technology*, IEEE, Jun. 2019, T86–T87. DOI: [10.23919/vlsit.2019.8776485](https://doi.org/10.23919/vlsit.2019.8776485).
 - [A96] K. Suliskas and M. Borg, “Advantages of binary stochastic synapses for hardware spiking neural networks with realistic memristors,” *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 034008, Aug. 2022, ISSN: 2634-4386. DOI: [10.1088/2634-4386/ac7c89](https://doi.org/10.1088/2634-4386/ac7c89).
 - [A97] S. Jenderny, K. Ochs, and P. Hövel, “A memristor-based circuit approximation of the Hindmarsh-Rose model,” *The European Physical Journal B*, vol. 96, no. 8, Aug. 2023, ISSN: 1434-6036. DOI: [10.1140/epjb/s10051-023-00578-z](https://doi.org/10.1140/epjb/s10051-023-00578-z).
 - [A98] A. H K, A. Kulkarni, A. B, C. K T, and P. Mala S, “A Study on Circuit Optimization and Performance of Memristor-Integrated 3 bit Pipeline DAC,” in *2024 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, IEEE, Sep. 2024, pp. 1–6. DOI: [10.1109/nkcon62728.2024.10774831](https://doi.org/10.1109/nkcon62728.2024.10774831).
 - [A99] N. Elashkar, G. Ibrahim, M. Aboudina, H. Fahmy, and A. Hussein, “Memristor-Based Phase-Frequency Detector for Phase-Locked Loop Applications,” in *2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, IEEE, Apr. 2024, pp. 1–5. DOI: [10.1109/estream61684.2024.10542599](https://doi.org/10.1109/estream61684.2024.10542599).
 - [A100] Y. Lee and A. Waterman, “Managing Chip Design Complexity in the Domain-Specific SoC Era,” in *2020 IEEE Symposium on VLSI Circuits*, 2020, pp. 1–2. DOI: [10.1109/vlsicircuits18222.2020.9162812](https://doi.org/10.1109/vlsicircuits18222.2020.9162812).
 - [A101] F. Schirrmeister, *Design Complexity In The Golden Age Of Semiconductors — semiengineering.com*, <https://semiengineering.com/design-complexity-in-the-golden-age-of-semiconductors/>, [Accessed 14.01.2025], 2023.
 - [A102] S. Ha et al., “Introduction to Hardware/Software Codesign,” in *Handbook of Hardware/Software Codesign*. Springer Netherlands, 2017, pp. 3–26, ISBN: 9789401772679. DOI: [10.1007/978-94-017-7267-9_41](https://doi.org/10.1007/978-94-017-7267-9_41).
 - [A103] B. Bailey, G. Martin, and A. Piziali, “Chapter 1 - What Is ESL?” In *ESL Design and Verification* (Systems on Silicon), B. Bailey, G. Martin, and A. Piziali, Eds., Systems on Silicon. Burlington: Morgan Kaufmann, 2007, pp. 1–9. DOI: [10.1016/b978-012373551-5/50064-2](https://doi.org/10.1016/b978-012373551-5/50064-2).
 - [A104] A. Küster, R. Dorsch, and C. Haubelt, “Towards Fast Heterogeneous Virtual Prototypes: Increasing the Solver Efficiency in SystemC AMS,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025. DOI: [10.1109/tcad.2025.3554612](https://doi.org/10.1109/tcad.2025.3554612).
 - [A105] Gajski and Kuhn, “Guest Editors’ Introduction: New VLSI Tools,” *Computer*, vol. 16, no. 12, pp. 11–14, Dec. 1983, ISSN: 0018-9162. DOI: [10.1109/mc.1983.1654264](https://doi.org/10.1109/mc.1983.1654264).
 - [A106] I. Grout, *Digital systems design with FPGAs and CPLDs*. Jan. 2010, ISBN: 075068397X.
 - [A107] B. J. LaMeres, *Quick Start Guide to Verilog*. Springer International Publishing, 2024, ISBN: 9783031441042. DOI: [10.1007/978-3-031-44104-2](https://doi.org/10.1007/978-3-031-44104-2).

-
- [A108] J. Teich, “Hardware/Software Codesign: The Past, the Present, and Predicting the Future,” *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1411–1430, May 2012, ISSN: 1558-2256. DOI: [10.1109/jproc.2011.2182009](https://doi.org/10.1109/jproc.2011.2182009).
 - [A109] P. Pouyan, E. Amat, and A. Rubio, “Reliability challenges in design of memristive memories,” in *2014 5th European Workshop on CMOS Variability (VARI)*, 2014, pp. 1–6. DOI: [10.1109/vari.2014.6957074](https://doi.org/10.1109/vari.2014.6957074).
 - [A110] D. Ielmini and H.-S. P. Wong, “In-memory computing with resistive switching devices,” *Nature Electronics*, vol. 1, no. 6, pp. 333–343, Jun. 2018, ISSN: 2520-1131. DOI: [10.1038/s41928-018-0092-2](https://doi.org/10.1038/s41928-018-0092-2).
 - [A111] R. Khan, N. U. Rehman, S. Iqbal, S. Abdullaev, and H. M. Aldosari, “Resistive Switching Properties in Memristors for Optoelectronic Synaptic Memristors: Deposition Techniques, Key Performance Parameters, and Applications,” *ACS Applied Electronic Materials*, vol. 6, no. 1, pp. 73–119, Dec. 2023, ISSN: 2637-6113. DOI: [10.1021/acsaelm.3c01323](https://doi.org/10.1021/acsaelm.3c01323).
 - [A112] O. Mutlu and L. Subramanian, “Research Problems and Opportunities in Memory Systems,” *Supercomputing Frontiers and Innovations*, vol. 1, no. 3, Sep. 2014, ISSN: 2313-8734. DOI: [10.14529/jsfi140302](https://doi.org/10.14529/jsfi140302).
 - [A113] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, “FlashSim: A Simulator for NAND Flash-Based Solid-State Drives,” in *2009 First International Conference on Advances in System Simulation*, IEEE, Sep. 2009, pp. 125–131. DOI: [10.1109/simul.2009.17](https://doi.org/10.1109/simul.2009.17).
 - [A114] R. Canal et al., “TRAMS Project: Variability and Reliability of SRAM Memories in sub-22nm Bulk-CMOS Technologies,” *Procedia Computer Science*, vol. 7, pp. 148–149, 2011, ISSN: 1877-0509. DOI: [10.1016/j.procs.2011.09.010](https://doi.org/10.1016/j.procs.2011.09.010).
 - [A115] C. Sanz, J. I. Gómez, C. Tenllado, M. Prieto, and F. Catthoor, “System-level memory management based on statistical variability compensation for frame-based applications,” *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 1s, pp. 1–28, Nov. 2013, ISSN: 1558-3465. DOI: [10.1145/2536747.2536757](https://doi.org/10.1145/2536747.2536757).
 - [A116] D. Lee et al., *Understanding and Exploiting Design-Induced Latency Variation in Modern DRAM Chips*, 2016. DOI: [10.48550/arxiv.1610.09604](https://doi.org/10.48550/arxiv.1610.09604).
 - [A117] L. Steiner, M. Jung, F. S. Prado, K. Bykov, and N. Wehn, “DRAMSys4.0: An Open-Source Simulation Framework for In-depth DRAM Analyses,” *International Journal of Parallel Programming*, vol. 50, no. 2, pp. 217–242, Mar. 2022, ISSN: 1573-7640. DOI: [10.1007/s10766-022-00727-4](https://doi.org/10.1007/s10766-022-00727-4).
 - [A118] C.-H. Chang, C. Liu, L. Zhang, and Z. Kong, “Sizing of SRAM Cell with Voltage Biasing Techniques for Reliability Enhancement of Memory and PUF Functions,” *Journal of Low Power Electronics and Applications*, vol. 6, no. 3, p. 16, Aug. 2016, ISSN: 2079-9268. DOI: [10.3390/jlpea6030016](https://doi.org/10.3390/jlpea6030016).
 - [A119] S. Shen et al., *OpenYield: An Open-Source SRAM Yield Analysis and Optimization Benchmark Suite*, 2025. DOI: [10.48550/arxiv.2508.04106](https://doi.org/10.48550/arxiv.2508.04106).
 - [A120] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, “Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1859–1880, 2005. DOI: [10.1109/tcad.2005.852295](https://doi.org/10.1109/tcad.2005.852295).
 - [A121] P. Zuber, M. Miranda, P. Dobrovolný, K. van der Zanden, and J.-H. Jung, “Statistical SRAM analysis for yield enhancement,” in *Proceedings of the Conference on Design,*

- Automation and Test in Europe*, ser. DATE '10, Dresden, Germany: European Design and Automation Association, 2010, pp. 57–62, ISBN: 9783981080162.
- [A122] I. Bhati, M.-T. Chang, Z. Chishti, S.-L. Lu, and B. Jacob, “DRAM Refresh Mechanisms, Penalties, and Trade-Offs,” *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 108–121, 2016. DOI: [10.1109/tc.2015.2417540](https://doi.org/10.1109/tc.2015.2417540).
 - [A123] K. M. Koo, W. Y. Chung, S. Y. Lee, G. H. Yoon, and W. Y. Choi, “Modeling of Statistical Variation Effects on DRAM Sense Amplifier Offset Voltage,” *Micromachines*, vol. 12, no. 10, p. 1145, Sep. 2021, ISSN: 2072-666X. DOI: [10.3390/mi12101145](https://doi.org/10.3390/mi12101145).
 - [A124] X. Jimenez, D. Novo, and P. Ienne, “Wear Unleveling: Improving NAND Flash Lifetime by Balancing Page Endurance,” in *12th USENIX Conference on File and Storage Technologies (FAST 14)*, Santa Clara, CA: USENIX Association, Feb. 2014, pp. 47–59, ISBN: ISBN 978-1-931971-08-9. [Online]. Available: <https://www.usenix.org/conference/fast14/technical-sessions/presentation/jimenez>.
 - [A125] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, “Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives,” *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1666–1704, 2017. DOI: [10.1109/jproc.2017.2713127](https://doi.org/10.1109/jproc.2017.2713127).
 - [A126] M. Buddhanoy, S. Sakib, and B. Ray, “Runtime Variability Monitor for Data Retention Characteristics of Commercial NAND Flash Memory,” in *2021 IEEE International Reliability Physics Symposium (IRPS)*, 2021, pp. 1–5. DOI: [10.1109/irps46558.2021.9405143](https://doi.org/10.1109/irps46558.2021.9405143).
 - [A127] M. Indaco, P. Prinetto, and E. I. Vatajelu, “On the impact of process variability and aging on the reliability of emerging memories (Embedded tutorial),” in *2014 19th IEEE European Test Symposium (ETS)*, 2014, pp. 1–10. DOI: [10.1109/ets.2014.6847813](https://doi.org/10.1109/ets.2014.6847813).
 - [A128] O. Kapur, *Dataset in support of the journal article 'An ultra high-endurance memristor using back-end-of-line amorphous SiC'*, May 2024. DOI: [10.5258/soton/d3073](https://doi.org/10.5258/soton/d3073).
 - [A129] M. Lanza et al., “Recommended Methods to Study Resistive Switching Devices,” *Advanced Electronic Materials*, vol. 5, no. 1, Sep. 2018, ISSN: 2199-160X. DOI: [10.1002/aelm.201800143](https://doi.org/10.1002/aelm.201800143).
 - [A130] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, “Memristor SPICE model and crossbar simulation based on devices with nanosecond switching time,” in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Aug. 2013, pp. 1–7. DOI: [10.1109/ijcnn.2013.6706773](https://doi.org/10.1109/ijcnn.2013.6706773).
 - [A131] B. Hajri, M. M. Mansour, A. Chehab, and H. Aziza, “Oxide-based RRAM models for circuit designers: A comparative analysis,” in *2017 12th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*, IEEE, Apr. 2017. DOI: [10.1109/dtis.2017.7930176](https://doi.org/10.1109/dtis.2017.7930176).
 - [A132] B. Hajri, M. M. Mansour, A. Chehab, and H. Aziza, “Memristor models optimization for large-scale 1T1R memory arrays,” in *2018 International Conference on IC Design & Technology (ICICDT)*, IEEE, Jun. 2018, pp. 109–112. DOI: [10.1109/icicdt.2018.8399768](https://doi.org/10.1109/icicdt.2018.8399768).
 - [A133] B. Hajri, H. Aziza, M. M. Mansour, and A. Chehab, “RRAM Device Models: A Comparative Analysis With Experimental Validation,” *IEEE Access*, vol. 7, pp. 168 963–168 980, 2019, ISSN: 2169-3536. DOI: [10.1109/access.2019.2954753](https://doi.org/10.1109/access.2019.2954753).
 - [A134] M. H. Maruf and S. I. Ali, “Review and comparative study of I-V characteristics of different memristor models with sinusoidal input,” *International Journal of Elec-*

- tronics, vol. 107, no. 3, pp. 349–375, Sep. 2019, ISSN: 1362-3060. DOI: [10.1080/00207217.2019.1661021](https://doi.org/10.1080/00207217.2019.1661021).
- [A135] S. Kvatinsky, E. G. Friedman, A. Kolodny, and U. C. Weiser, “TEAM: ThrEshold Adaptive Memristor Model,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 211–221, 2013. DOI: [10.1109/tcsi.2012.2215714](https://doi.org/10.1109/tcsi.2012.2215714).
- [A136] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, “VTEAM: A General Model for Voltage-Controlled Memristors,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 786–790, 2015. DOI: [10.1109/tcsii.2015.2433536](https://doi.org/10.1109/tcsii.2015.2433536).
- [A137] S. Wiefels, C. Bengel, N. Kopperberg, K. Zhang, R. Waser, and S. Menzel, “HRS Instability in Oxide-Based Bipolar Resistive Switching Cells,” *IEEE Transactions on Electron Devices*, vol. 67, no. 10, pp. 4208–4215, Oct. 2020, ISSN: 1557-9646. DOI: [10.1109/ted.2020.3018096](https://doi.org/10.1109/ted.2020.3018096).
- [A138] A. Siemon, D. Wouters, S. Hamdioui, and S. Menzel, “Memristive Device Modeling and Circuit Design Exploration for Computation-in-Memory,” in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, May 2019, pp. 1–5. DOI: [10.1109/iscas.2019.8702600](https://doi.org/10.1109/iscas.2019.8702600).
- [A139] C. La Torre, A. F. Zurhelle, T. Breuer, R. Waser, and S. Menzel, “Compact Modeling of Complementary Switching in Oxide-Based ReRAM Devices,” *IEEE Transactions on Electron Devices*, vol. 66, no. 3, pp. 1268–1275, Mar. 2019, ISSN: 1557-9646. DOI: [10.1109/ted.2019.2892997](https://doi.org/10.1109/ted.2019.2892997).
- [A140] M. Lübben, S. Menzel, S. G. Park, M. Yang, R. Waser, and I. Valov, “SET kinetics of electrochemical metallization cells: influence of counter-electrodes in SiO₂/Ag based systems,” *Nanotechnology*, vol. 28, no. 13, p. 135 205, Mar. 2017, ISSN: 1361-6528. DOI: [10.1088/1361-6528/aa5e59](https://doi.org/10.1088/1361-6528/aa5e59).
- [A141] Y. N. Joglekar and S. J. Wolf, “The elusive memristor: properties of basic electrical circuits,” *European Journal of physics*, vol. 30, no. 4, p. 661, 2009. DOI: [10.1088/0143-0807/30/4/001](https://doi.org/10.1088/0143-0807/30/4/001).
- [A142] Z. Biolek, D. Biolek, and V. Biolkova, “SPICE Model of Memristor with Nonlinear Dopant Drift,” *Radioengineering*, vol. 18, no. 2, 2009. [Online]. Available: https://www.radioeng.cz/fulltexts/2009/09_02_210_214.pdf.
- [A143] C. Yakopcic, T. M. Taha, G. Subramanyam, and R. E. Pino, “Generalized Memristive Device SPICE Model and its Application in Circuit Design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 8, pp. 1201–1214, 2013. DOI: [10.1109/tcad.2013.2252057](https://doi.org/10.1109/tcad.2013.2252057).
- [A144] C. Yakopcic, T. M. Taha, D. J. Mountain, T. Salter, M. J. Marinella, and M. McLean, “Memristor Model Optimization Based on Parameter Extraction From Device Characterization Data,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 5, pp. 1084–1095, 2020. DOI: [10.1109/tcad.2019.2912946](https://doi.org/10.1109/tcad.2019.2912946).
- [A145] M. D. Pickett et al., “Switching dynamics in titanium dioxide memristive devices,” *Journal of Applied Physics*, vol. 106, no. 7, Oct. 2009, ISSN: 1089-7550. DOI: [10.1063/1.3236506](https://doi.org/10.1063/1.3236506).
- [A146] M. Bocquet et al., “Compact Modeling Solutions for Oxide-Based Resistive Switching Memories (OxRAM),” *Journal of Low Power Electronics and Applications*, vol. 4, no. 1, pp. 1–14, Jan. 2014, ISSN: 2079-9268. DOI: [10.3390/jlpea4010001](https://doi.org/10.3390/jlpea4010001).

-
- [A147] X. Guan, S. Yu, and H.-S. P. Wong, “A SPICE Compact Model of Metal Oxide Resistive Switching Memory With Variations,” *IEEE Electron Device Letters*, vol. 33, no. 10, pp. 1405–1407, Oct. 2012, ISSN: 1558-0563. DOI: [10.1109/led.2012.2210856](https://doi.org/10.1109/led.2012.2210856).
- [A148] Z. Jiang, S. Yu, Y. Wu, J. H. Engel, X. Guan, and H.-S. P. Wong, “Verilog-A compact model for oxide-based resistive random access memory (RRAM),” in *2014 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, IEEE, Sep. 2014. DOI: [10.1109/sispad.2014.6931558](https://doi.org/10.1109/sispad.2014.6931558).
- [A149] Z. Jiang et al., “A Compact Model for Metal-Oxide Resistive Random Access Memory With Experiment Verification,” *IEEE Transactions on Electron Devices*, vol. 63, no. 5, pp. 1884–1892, May 2016, ISSN: 1557-9646. DOI: [10.1109/ted.2016.2545412](https://doi.org/10.1109/ted.2016.2545412).
- [A150] F. L. Aguirre, J. Suñé, and E. Miranda, “SPICE Implementation of the Dynamic Memdiode Model for Bipolar Resistive Switching Devices,” *Micromachines*, vol. 13, no. 2, p. 330, Feb. 2022, ISSN: 2072-666X. DOI: [10.3390/mi13020330](https://doi.org/10.3390/mi13020330).
- [A151] I. Messaris, A. Serb, S. Stathopoulos, A. Khiat, S. Nikolaidis, and T. Prodromakis, “A Data-Driven Verilog-A ReRAM Model,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3151–3162, Dec. 2018, ISSN: 1937-4151. DOI: [10.1109/tcad.2018.2791468](https://doi.org/10.1109/tcad.2018.2791468).
- [A152] A. Serb, T. Prodromakis, and I. Messaris, *Dataset for A computationally efficient Verilog-A ReRAM model (v.2)*, 2017. DOI: [10.5258/soton/d0132](https://doi.org/10.5258/soton/d0132).
- [A153] D. Duncan, B. Magyari-Köpe, and Y. Nishi, “Filament-Induced Anisotropic Oxygen Vacancy Diffusion and Charge Trapping Effects in Hafnium Oxide RRAM,” *IEEE Electron Device Letters*, vol. 37, no. 4, pp. 400–403, 2016. DOI: [10.1109/led.2016.2524450](https://doi.org/10.1109/led.2016.2524450).
- [A154] L. Zhao, B. Magyari-Köpe, and Y. Nishi, “Polaronic interactions between oxygen vacancies in rutile TiO₂,” *Phys. Rev. B*, vol. 95, p. 054104, 5 Feb. 2017. DOI: [10.1103/physrevb.95.054104](https://doi.org/10.1103/physrevb.95.054104).
- [A155] M. Poremba and Y. Xie, “NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories,” in *2012 IEEE Computer Society Annual Symposium on VLSI*, IEEE, Aug. 2012, pp. 392–397. DOI: [10.1109/isvlsi.2012.82](https://doi.org/10.1109/isvlsi.2012.82).
- [A156] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, “DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies,” in *2019 IEEE International Electron Devices Meeting (IEDM)*, IEEE, Dec. 2019. DOI: [10.1109/iedm19573.2019.8993491](https://doi.org/10.1109/iedm19573.2019.8993491).
- [A157] Z. Zhu et al., “MNSIM 2.0: A Behavior-Level Modeling Tool for Processing-In-Memory Architectures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4112–4125, Nov. 2023, ISSN: 1937-4151. DOI: [10.1109/tcad.2023.3251696](https://doi.org/10.1109/tcad.2023.3251696).
- [A158] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, “RxNN: A Framework for Evaluating Deep Neural Networks on Resistive Crossbars,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 2, pp. 326–338, Feb. 2021, ISSN: 1937-4151. DOI: [10.1109/tcad.2020.3000185](https://doi.org/10.1109/tcad.2020.3000185).
- [A159] H. Gong et al., “ACCLAIM: An End-to-End SystemC-AMS Simulation Framework for Analog In-Memory-Computing,” in *2023 IEEE International Conference on In-*

- egrated Circuits, Technologies and Applications (ICTA)*, IEEE, Oct. 2023, pp. 134–135. DOI: [10.1109/icta60488.2023.10364310](https://doi.org/10.1109/icta60488.2023.10364310).
- [A160] H. Liu, J. Xu, X. Liao, H. Jin, Y. Zhang, and F. Mao, “A Simulation Framework for Memristor-Based Heterogeneous Computing Architectures,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 12, pp. 5476–5488, Dec. 2022, ISSN: 1937-4151. DOI: [10.1109/tcad.2022.3152385](https://doi.org/10.1109/tcad.2022.3152385).
- [A161] D. Sanchez and C. Kozyrakis, “ZSim: fast and accurate microarchitectural simulation of thousand-core systems,” *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 475–486, Jun. 2013, ISSN: 0163-5964. DOI: [10.1145/2508148.2485963](https://doi.org/10.1145/2508148.2485963).
- [A162] W. Song et al., “Programming memristor arrays with arbitrarily high precision for analog computing,” *Science*, vol. 383, no. 6685, pp. 903–910, Feb. 2024, ISSN: 1095-9203. DOI: [10.1126/science.adi9405](https://doi.org/10.1126/science.adi9405).
- [A163] H. Jiang et al., “Sub-10 nm Ta Channel Responsible for Superior Performance of a HfO₂ Memristor,” *Scientific Reports*, vol. 6, no. 1, Jun. 2016, ISSN: 2045-2322. DOI: [10.1038/srep28525](https://doi.org/10.1038/srep28525).
- [A164] D. Chettri, A. Thapa, S. Rai, P. Chettri, C. K. Sarkar, and B. Sharma, “Comparative computational study of LD-HfO₂ and TiO₂ as layered dielectrics in RRAM,” in *2019 Devices for Integrated Circuit (DevIC)*, IEEE, Mar. 2019, pp. 470–473. DOI: [10.1109/devic.2019.8783290](https://doi.org/10.1109/devic.2019.8783290).
- [A165] T. H. Park et al., “Thickness effect of ultra-thin Ta₂O₅ resistance switching layer in 28 nm-diameter memory cell,” *Scientific Reports*, vol. 5, no. 1, Nov. 2015, ISSN: 2045-2322. DOI: [10.1038/srep15965](https://doi.org/10.1038/srep15965).
- [A166] E. Perez et al., “A Comparison of Resistive Switching Parameters for Memristive Devices with HfO₂ Mono Layers and Al₂O₃/HfO₂ Bilayers at the Wafer Scale,” in *2023 14th Spanish Conference on Electron Devices (CDE)*, IEEE, Jun. 2023, pp. 1–5. DOI: [10.1109/cde58627.2023.10339417](https://doi.org/10.1109/cde58627.2023.10339417).
- [A167] H. Kim, M. R. Mahmoodi, H. Nili, and D. B. Strukov, “4K-memristor analog-grade passive crossbar circuit,” *Nature Communications*, vol. 12, no. 1, Aug. 2021, ISSN: 2041-1723. DOI: [10.1038/s41467-021-25455-0](https://doi.org/10.1038/s41467-021-25455-0).
- [A168] M. K. Mahadevaiah et al., “Integration of Memristive Devices into a 130 nm CMOS Baseline Technology,” in *Bio-Inspired Information Pathways*. Springer International Publishing, Sep. 2023, pp. 177–190, ISBN: 9783031367052. DOI: [10.1007/978-3-031-36705-2_7](https://doi.org/10.1007/978-3-031-36705-2_7).
- [A169] K. Kumari, S. Majumder, A. D. Thakur, and S. Ray, “Temperature-dependent resistive switching behaviour of an oxide memristor,” *Materials Letters*, vol. 303, p. 130451, Nov. 2021, ISSN: 0167-577X. DOI: [10.1016/j.matlet.2021.130451](https://doi.org/10.1016/j.matlet.2021.130451).
- [A170] T. Rizzi et al., “Process-Voltage-Temperature Variations Assessment in Energy-Aware Resistive RAM-Based FPGAs,” *IEEE Transactions on Device and Materials Reliability*, vol. 23, no. 3, pp. 328–336, Sep. 2023, ISSN: 1558-2574. DOI: [10.1109/tdmr.2023.3259015](https://doi.org/10.1109/tdmr.2023.3259015).
- [A171] X. Zhang et al., “Humidity Effect on Resistive Switching Characteristics of the CH₃NH₃PbI₃ Memristor,” *ACS Applied Materials & Interfaces*, vol. 13, no. 24, pp. 28555–28563, Jun. 2021, ISSN: 1944-8252. DOI: [10.1021/acsami.1c05590](https://doi.org/10.1021/acsami.1c05590).
- [A172] Y. Gonzalez-Velo, H. J. Barnaby, and M. N. Kozicki, “Review of radiation effects on ReRAM devices and technology,” *Semiconductor Science and Technology*, vol. 32, no. 8, p. 083002, Jul. 2017, ISSN: 1361-6641. DOI: [10.1088/1361-6641/aa6124](https://doi.org/10.1088/1361-6641/aa6124).

- [A173] S. Ambrogio, S. Balatti, S. Choi, and D. Ielmini, “Impact of the Mechanical Stress on Switching Characteristics of Electrochemical Resistive Memory,” *Advanced Materials*, vol. 26, no. 23, pp. 3885–3892, Mar. 2014, ISSN: 1521-4095. DOI: [10.1002/adma.201306250](https://doi.org/10.1002/adma.201306250).
- [A174] N. Wald and S. Kvatinsky, “Understanding the influence of device, circuit and environmental variations on real processing in memristive memory using Memristor Aided Logic,” *Microelectronics Journal*, vol. 86, pp. 22–33, Apr. 2019, ISSN: 1879-2391. DOI: [10.1016/j.mejo.2019.02.013](https://doi.org/10.1016/j.mejo.2019.02.013).
- [A175] D. Maldonado et al., “Influence of stop and gate voltage on resistive switching of 1T1R HfO₂-based memristors, a modeling and variability analysis,” *Materials Science in Semiconductor Processing*, vol. 182, p. 108726, Nov. 2024, ISSN: 1369-8001. DOI: [10.1016/j.mssp.2024.108726](https://doi.org/10.1016/j.mssp.2024.108726).
- [A176] H. Li et al., “Variation-Aware, Reliability-Emphasized Design and Optimization of RRAM using SPICE Model,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, ser. DATE 2015, IEEE Conference Publications, 2015, pp. 1425–1430. DOI: [10.7873/date.2015.0362](https://doi.org/10.7873/date.2015.0362).
- [A177] E. Pérez et al., “Optimization of Multi-Level Operation in RRAM Arrays for In-Memory Computing,” *Electronics*, vol. 10, no. 9, p. 1084, May 2021, ISSN: 2079-9292. DOI: [10.3390/electronics10091084](https://doi.org/10.3390/electronics10091084).
- [A178] H. Fischer, *A History of the Central Limit Theorem: From Classical to Modern Probability Theory*. Springer New York, 2011, ISBN: 9780387878577. DOI: [10.1007/978-0-387-87857-7](https://doi.org/10.1007/978-0-387-87857-7).
- [A179] D. Walczyk et al., “Resistive switching behavior in TiN/HfO₂/Ti/TiN devices,” in *2012 International Semiconductor Conference Dresden-Grenoble (ISCDG)*, 2012, pp. 143–146. DOI: [10.1109/iscdg.2012.6360035](https://doi.org/10.1109/iscdg.2012.6360035).
- [A180] GCC, *Random Number Generation Implementation*, [Accessed: 26.07.2025], Jul. 2025. [Online]. Available: <https://gcc.gnu.org/git/?p=gcc.git;a=blob;f=libstdc%2B%2B-v3/include/bits/random.tcc;h=53ccacb2e38b4a6b0671d7af0657a9896cd7e735;hb=HEAD#l1807>.
- [A181] J. Reuben, D. Fey, and C. Wenger, “A Modeling Methodology for Resistive RAM Based on Stanford-PKU Model With Extended Multilevel Capability,” *IEEE Transactions on Nanotechnology*, vol. 18, pp. 647–656, 2019, ISSN: 1941-0085. DOI: [10.1109/tnano.2019.2922838](https://doi.org/10.1109/tnano.2019.2922838).
- [A182] J. Reuben, M. Biglari, and D. Fey, “Incorporating Variability of Resistive RAM in Circuit Simulations Using the Stanford-PKU Model,” *IEEE Transactions on Nanotechnology*, vol. 19, pp. 508–518, 2020, ISSN: 1941-0085. DOI: [10.1109/tnano.2020.3004666](https://doi.org/10.1109/tnano.2020.3004666).
- [A183] V. Milo et al., “Accurate Program/Verify Schemes of Resistive Switching Memory (RRAM) for In-Memory Neural Network Circuits,” *IEEE Transactions on Electron Devices*, vol. 68, no. 8, pp. 3832–3837, Aug. 2021, ISSN: 1557-9646. DOI: [10.1109/ted.2021.3089995](https://doi.org/10.1109/ted.2021.3089995).
- [A184] C. Villani, “The Wasserstein distances,” in *Optimal Transport*. Springer Berlin Heidelberg, 2009, pp. 93–111, ISBN: 9783540710509. DOI: [10.1007/978-3-540-71050-9_6](https://doi.org/10.1007/978-3-540-71050-9_6).
- [A185] Google, *Google Benchmark: A microbenchmark support library*, <https://github.com/google/benchmark>, [Accessed 01.08.2025], 2016.

-
- [A186] M. Streubühr, R. Rosales, R. Hasholzner, C. Haubelt, and J. Teich, “ESL power and performance estimation for heterogeneous MPSoCS using SystemC,” in *FDL 2011 Proceedings*, 2011, pp. 1–8. [Online]. Available: <https://ieeexplore.ieee.org/document/6069489>.
 - [A187] IEEE, “IEEE Standard for Standard SystemC® Language Reference Manual,” *IEEE Std 1666-2023 (Revision of IEEE Std 1666-2011)*, pp. 1–618, 2023. DOI: [10.1109/ieeestd.2023.10246125](https://doi.org/10.1109/ieeestd.2023.10246125).
 - [A188] S. M. Salerno, P. C. Alguire, and H. S. Waxman, “Training and Competency Evaluation for Interpretation of 12-Lead Electrocardiograms: Recommendations from the American College of Physicians*,” *Annals of Internal Medicine*, vol. 138, no. 9, pp. 747–750, May 2003, ISSN: 1539-3704. DOI: [10.7326/0003-4819-138-9-200305060-00012](https://doi.org/10.7326/0003-4819-138-9-200305060-00012).
 - [A189] A. Burger, C. Qian, G. Schiele, and D. Helms, “An Embedded CNN Implementation for On-Device ECG Analysis,” in *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, Mar. 2020, pp. 1–6. DOI: [10.1109/percomworkshops48775.2020.9156260](https://doi.org/10.1109/percomworkshops48775.2020.9156260).
 - [A190] J. M. Jenkins and S. A. Caswell, “Detection algorithms in implantable cardioverter defibrillators,” *Proceedings of the IEEE*, vol. 84, no. 3, pp. 428–445, 2002. DOI: [10.1109/5.486745](https://doi.org/10.1109/5.486745).
 - [A191] Y. Khan, A. E. Ostfeld, C. M. Lochner, A. Pierre, and A. C. Arias, “Monitoring of Vital Signs with Flexible and Wearable Medical Devices,” *Advanced Materials*, vol. 28, no. 22, pp. 4373–4395, Feb. 2016, ISSN: 1521-4095. DOI: [10.1002/adma.201504366](https://doi.org/10.1002/adma.201504366).
 - [A192] German Federal Ministry of Education and Research (BMBF), *Pilotinnovationswettbewerb „Energieeffizientes KI-System“*, [Accessed: 15.09.2025], 2021. [Online]. Available: <https://www.elektronikforschung.de/service/aktuelles/pilotinnovationswettbewerb%5C#kategorie-asic-130>.
 - [A193] Y.-H. Chen, J. Emer, and V. Sze, “Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 367–379, Jun. 2016, ISSN: 0163-5964. DOI: [10.1145/3007787.3001177](https://doi.org/10.1145/3007787.3001177).
 - [A194] IHP GmbH - Leibniz-Institut für innovative Mikroelektronik, *SiGe BiCMOS and Silicon Photonics Technologies*, [Accessed: 21.07.2025]. [Online]. Available: <https://www.ihp-microelectronics.com/services/research-and-prototyping-service/mpw-prototyping-service/sigec-bicmos-technologies>.
 - [A195] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, “Analog architectures for neural network acceleration based on non-volatile memory,” *Applied Physics Reviews*, vol. 7, no. 3, Jul. 2020, ISSN: 1931-9401. DOI: [10.1063/1.5143815](https://doi.org/10.1063/1.5143815).
 - [A196] A. Baroni et al., “Low Conductance State Drift Characterization and Mitigation in Resistive Switching Memories (RRAM) for Artificial Neural Networks,” *IEEE Transactions on Device and Materials Reliability*, vol. 22, no. 3, pp. 340–347, Sep. 2022, ISSN: 1558-2574. DOI: [10.1109/tdmr.2022.3182133](https://doi.org/10.1109/tdmr.2022.3182133).
 - [A197] S. V. Weston et al., “State-of-the-Art Small Spacecraft Technology,” National Aeronautics and Space Administration (NASA), Washington, DC, USA, NASA Technical Publication TP-20250000142, Feb. 2025, Chapter 8: Small Spacecraft Avionics. [On-

- line]. Available: <https://www.nasa.gov/smallsat-institute/sst-soa/small-spacecraft-avionics/>.
- [A198] M. Ulbricht, L. Lu, J. Chen, and M. Krstic, "The TETRISC SoC—A resilient quad-core system based on the ResiliCell approach," *Microelectronics Reliability*, vol. 148, p. 115 173, Sep. 2023, ISSN: 0026-2714. DOI: [10.1016/j.microrel.2023.115173](https://doi.org/10.1016/j.microrel.2023.115173).
 - [A199] P. D. Schiavone, D. Rossi, A. Pullini, A. Di Mauro, F. Conti, and L. Benini, "Quentin: an Ultra-Low-Power PULPissimo SoC in 22nm FDX," in *2018 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, 2018, pp. 1–3. DOI: [10.1109/s3s.2018.8640145](https://doi.org/10.1109/s3s.2018.8640145).
 - [A200] J. Chen, T. Lange, M. Andjelkovic, A. Simevski, L. Lu, and M. Krstic, "Solar Particle Event and Single Event Upset Prediction from SRAM-Based Monitor and Supervised Machine Learning," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 564–580, 2022. DOI: [10.1109/tetc.2022.3147376](https://doi.org/10.1109/tetc.2022.3147376).
 - [A201] W. G. Bennett et al., "Single- and Multiple-Event Induced Upsets in HfO₂/Hf 1T1R RRAM," *IEEE Transactions on Nuclear Science*, vol. 61, no. 4, pp. 1717–1725, 2014. DOI: [10.1109/tns.2014.2321833](https://doi.org/10.1109/tns.2014.2321833).
 - [A202] D. Chen et al., "Radiation Effects of Commercial Resistive Random Access Memories," in *NASA Electronic Parts and Packaging (NEPP) Electronics Technology Workshop (ETW)*, 2014. [Online]. Available: <https://ntrs.nasa.gov/citations/20140017349>.
 - [A203] H. Song, K.-k. Ni, Y. Tang, J. Wang, H. Guo, and X. Zhong, "Total ionizing dose effects of 60Co- ray radiation on the resistive switching and its bending performance of Al-in-O/InOx-based flexible RRAM device," *Radiation Physics and Chemistry*, vol. 182, p. 109 394, May 2021, ISSN: 0969-806X. DOI: [10.1016/j.radphyschem.2021.109394](https://doi.org/10.1016/j.radphyschem.2021.109394).
 - [A204] K.-E. Harabi et al., "A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '23, ACM, Jan. 2023, pp. 184–185. DOI: [10.1145/3566097.3567944](https://doi.org/10.1145/3566097.3567944).
 - [A205] J. L. Hindmarsh, R. M. Rose, and A. F. Huxley, "A model of neuronal bursting using three coupled first order differential equations," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 221, no. 1222, pp. 87–102, 1984. DOI: [10.1098/rspb.1984.0024](https://doi.org/10.1098/rspb.1984.0024).
 - [A206] A. Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986, ISSN: 0018-9219. DOI: [10.1109/proc.1986.13458](https://doi.org/10.1109/proc.1986.13458).
 - [A207] E. Solan, E. Pérez, D. Michaelis, C. Wenger, and K. Ochs, "Wave digital model of a TiN/Ti/HfO₂/TiN memristor," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 32, no. 5, Mar. 2019, ISSN: 1099-1204. DOI: [10.1002/jnm.2588](https://doi.org/10.1002/jnm.2588).

Appendix C

List of Peer-Reviewed Publications

- [B1] J. Knodtel, M. Fritscher, D. Reiser, D. Fey, M. Breiling, and M. Reichenbach, “A Model-to-Circuit Compiler for Evaluation of DNN Accelerators based on Systolic Arrays and Multibit Emerging Memories,” *2020 9th International Conference on Modern Circuits and Systems Technologies, MOCAST 2020*, 2020. DOI: [10.1109/MOCAST49295.2020.9200241](https://doi.org/10.1109/MOCAST49295.2020.9200241).
- [B2] M. Fritscher et al., “Simulating large neural networks embedding MLC RRAM as weight storage considering device variations,” *IEEE*, Feb. 2021. DOI: [10.1109/lascas51355.2021.9459159](https://doi.org/10.1109/lascas51355.2021.9459159).
- [B3] S. Pechmann et al., “A Low-Power RRAM Memory Block for Embedded, Multi-Level Weight and Bias Storage in Artificial Neural Networks,” *Micromachines*, vol. 12, no. 11, p. 1277, Oct. 2021. DOI: [10.3390/mi12111277](https://doi.org/10.3390/mi12111277).
- [B4] D. Reiser et al., “A Framework for Ultra Low-Power Hardware Accelerators Using NNs for Embedded Time Series Classification,” *Journal of Low Power Electronics and Applications*, vol. 12, no. 1, p. 2, Dec. 2021, ISSN: 2079-9268. DOI: [10.3390/jlpea12010002](https://doi.org/10.3390/jlpea12010002).
- [B5] P. Holzinger, D. Reiser, T. Hahn, and M. Reichenbach, “Fast HBM Access with FPGAs: Analysis, Architectures, and Applications,” *IEEE*, Jun. 2021. DOI: [10.1109/ipdpsw52791.2021.00030](https://doi.org/10.1109/ipdpsw52791.2021.00030).
- [B6] D. Reiser et al., “Technology-Aware Drift Resilience Analysis of RRAM Crossbar Array Configurations,” in *2023 21st IEEE Interregional NEWCAS Conference (NEWCAS)*, IEEE, Jun. 2023. DOI: [10.1109/newcas57931.2023.10198076](https://doi.org/10.1109/newcas57931.2023.10198076).
- [B7] D. Reiser, J. Chen, J. Knödtel, A. Baroni, M. Krstić, and M. Reichenbach, “Design and analysis of an adaptive radiation resilient RRAM subsystem for processing systems in satellites,” *Design Automation for Embedded Systems*, vol. 28, no. 2, pp. 111–137, Apr. 2024, ISSN: 1572-8080. DOI: [10.1007/s10617-024-09285-z](https://doi.org/10.1007/s10617-024-09285-z).
- [B8] D. Reiser et al., “HyRPF: Hybrid RRAM Prototyping on FPGA,” in *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Springer Nature Switzerland, 2025, pp. 199–215, ISBN: 9783031783777. DOI: [10.1007/978-3-031-78377-7_14](https://doi.org/10.1007/978-3-031-78377-7_14).
- [B9] J. Wen et al., “Cycle-Accurate FPGA Emulation of RRAM Crossbar Array: Efficient Device and Variability Modeling with Energy Consumption Assessment,” in *2024*

- IEEE 25th Latin American Test Symposium (LATS)*, IEEE, Apr. 2024, pp. 1–6. DOI: [10.1109/lats62223.2024.10534601](https://doi.org/10.1109/lats62223.2024.10534601).
- [B10] J. Wen et al., “RRAMulator: An efficient FPGA-based emulator for RRAM crossbar with device variability and energy consumption evaluation,” *Microelectronics Reliability*, vol. 168, p. 115 630, May 2025, ISSN: 0026-2714. DOI: [10.1016/j.microrel.2025.115630](https://doi.org/10.1016/j.microrel.2025.115630).
- [B11] S. Jenderny, D. Reiser, K. Ochs, and M. Reichenbach, “Towards a Memristor-Based Circuit Implementation of the Hindmarsh-Rose Model,” *International Journal of Circuit Theory and Applications*, Sep. 2025, ISSN: 1097-007X. DOI: [10.1002/cta.70126](https://doi.org/10.1002/cta.70126).
- [B12] M. Fritscher et al., “RISC-V CPU Design Using RRAM-CMOS Standard Cells,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1–9, 2025, ISSN: 1557-9999. DOI: [10.1109/tvlsi.2025.3554476](https://doi.org/10.1109/tvlsi.2025.3554476).

Appendix D

List of Other Publications

- [C1] D. Reiser and M. Reichenbach, “Emerging Non-Volatile Memory for Smart Medical Devices,” in *3rd Workshop on smart medical devices and systems - RC-SMDS '22*. BTU Cottbus - Senftenberg, 2023. DOI: [10.26127/btuopen-6304](https://doi.org/10.26127/btuopen-6304).
- [C2] D. Reiser, M. Fritscher, and M. Reichenbach, “MIMEC: Memristive In-Memory-Computing,” in *1. Memristor-Symposium 2023*. BTU Cottbus - Senftenberg, 2023. DOI: [10.26127/btuopen-6348](https://doi.org/10.26127/btuopen-6348).