Reliable Analog In-Memory Computing with Crossbars: Memristors for Analog Neural Computing

Other titles in Foundations and Trends® in Integrated Circuits and Systems

Towards Scalable Quantum Sensors: Interface Electronics for Quantum Sensors
Michal Kern, Khubaib Khan, Philipp Hengel and Jens Anders
ISBN: 978-1-63828-490-1

QED and Symbolic QED: Dramatic Improvements in Pre-Silicon Verification and Post-Silicon Validation
Keerthikumara Devarajegowda, Florian Lonsing, Mohammad R. Fadiheh, Saranyu Chattopadhyay, David Lin, Srinivas Shashank Nuthakki, Eshan Singh, Clark Barrett, Wolfgang Ecker, Wolfgang Kunz, Yanjing Li, Dominik Stoffel and Subhasish Mitra
ISBN: 978-1-63828-998-2

Energy-Efficient Time-Domain Computation for Edge Devices: Challenges and Prospects
Hamza Al Maharmeh, Mohammed Ismail and Mohammad Alhawari
ISBN: 978-1-63828-356-0

Recent Advances in Testing Techniques for AI Hardware Accelerators
Arjun Chaudhuri, Ching-Yuan Chen and Krishnendu Chakrabarty
ISBN: 978-1-63828-240-2

Systematic Design of Analog CMOS Circuits with Lookup Tables
Paul G. A. Jespers
ISBN: 978-1-63828-194-8

Of Brains and Computers
Jan M. Rabaey
ISBN: 978-1-63828-120-7

Reliable Analog In-Memory Computing with Crossbars: Memristors for Analog Neural Computing

Alex James
Digital University Kerala
apj@ieee.org

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Integrated Circuits and Systems

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

A. James. *Reliable Analog In-Memory Computing with Crossbars: Memristors for Analog Neural Computing*. Foundations and Trends® in Integrated Circuits and Systems, vol. 4, no. 1, pp. 1–114, 2025.

ISBN: 978-1-63828-563-2

Foundations and Trends® in Integrated Circuits and Systems
Volume 4, Issue 1, 2025

Editorial Board

Editorial Scope

Foundations and Trends® in Integrated Circuits and Systems survey and tutorial articles in the following topics:

- Analog, digital and mixed-signal circuits and systems
- RF and mm-wave integrated circuits and systems
- Wireless and wireline communication circuits and systems
- Data converters and frequency generation
- Power electronics and power management circuits
- Biomedical circuits and systems
- Sensor and imager circuits and cyber physical systems
- Security and resilient circuits and systems
- Circuits and systems in emerging non-CMOS technologies
- Circuit theory, modeling, analysis and design methods

Information for Librarians

Foundations and Trends® in Integrated Circuits and Systems, 2025, Volume 4, 4 issues. ISSN paper version 2693-9347. ISSN online version 2693-9355. Also available as a combined paper and online subscription.

Contents

Reliable Analog In-Memory Computing with Crossbars: Memristors for Analog Neural Computing

Alex James

AI Chip Centre, Digital University Kerala and Kairali Semiconductor Pvt Ltd., India; apj@ieee.org

ABSTRACT

Memristors as devices, and the systems built with them, have shown to be of great promise for use in analog neural computing. Every attempt to create an energy-efficient CMOS-based general purpose neural network processor that can compete with human intelligence seems to have failed. Memristive systems and devices are compatible and scalable with CMOS technology and show response behavior to stimuli similar to a biological neuron. This has prompted a closer look at memristive systems in academia and industry through the lens of beyond CMOS technologies, algorithms and applications.

In this monograph, in-memory computing is presented with the memristor as the enabling memory element. The practical memristor device faces several challenges when targeting on-chip implementations. Often, there are conductance variabilites of different forms resulting from device-to-device variability, aging, circuit parasitics, read instabilities, various

Alex James (2025), "Reliable Analog In-Memory Computing with Crossbars: Memristors for Analog Neural Computing", Foundations and Trends® in Integrated Circuits and Systems: Vol. 4, No. 1, pp 1–114. DOI: 10.1561/3500000018.

types of noises, and conductance drifts. This variability and how it can be analysed is introduced, along with the concept of super-resolution for compensating errors in analog computing. The application of memristive processing is also shown through echo-state networks for energy-efficient computing and image filtering processing for RF applications.

Part I

Memristive Circuits and Arrays

1

Introduction

Artificial intelligence (AI) hardware is a growing area of research that focuses on implementing specialized hardware chips designed for machine learning, neural networks, and their applications [23, 37]. The AI hardware and related chips include the design of efficient processors, memory, and dedicated circuits running AI workloads at extreme efficiency and processing speeds.

At the heart of neural network implementations, there are model neurons that are primarily memory functions capable of learning and adapting to new information. Memory is essential for enabling various learning functions and is inherent in all intelligent beings.

1.1 Intelligence in the Natural World

Intelligence exists from humans to microorganisms, with humans having the highest level of intelligence abilities in the natural world (Table 1.1). Humans are able to achieve this using the long-term and short-term memory that is hardwired by approximately 86 billion neurons that can perform around 100 trillion synaptic operations per second. This processing power, paired with advanced learning abilities and an estimated memory capacity of 2.5 petabytes, allows humans to adapt

Table 1.1: Comparison of intelligence in the natural world.

Category	Memory	Neurons/ Cells	Processing Capability	Learning	Adaptability	Power Consumption	Memory Capacity (Numbers)
Humans	Long-term memory, short-term memory	86 billion neurons	100 trillion synaptic operations per second	Advanced learning through experience and education	Highly adaptable, able to modify environment	20 Watts	Estimated at 2.5 petabytes
Animals	Episodic memory, spatial memory	Millions to billions of neurons	Millions to billions of synaptic operations per second	Learning from experience, conditioning	Adaptable to different environments, capable of behavioral changes	5-10 Watts (depending on animal)	Estimated in terabytes (varies by species)
Birds	Spatial memory (especially for migratory species)	100 million to 1 billion neurons	Specialized processing for navigation and mimicry	Learning through imitation and problem-solving	Adaptable, especially in terms of migration and diet	2-5 Watts	Estimated in gigabytes to terabytes (varies by species)
Insects	Short-term memory, collective memory in colonies	Thousands to hundreds of thousands of neurons (in ganglia)	Distributed processing within colonies	Limited learning through conditioning and trial-and-error	Highly adaptable in colonies, capable of optimizing foraging paths	0.001-0.01 Watts	Estimated in megabytes to gigabytes
Plants	Cellular memory through chemical signaling	No neurons, but complex signaling pathways	Slow chemical processing	No learning in the traditional sense, but responds to environmental changes	Adaptable through growth patterns and chemical responses	Negligible	N/A (memory through chemical signalling rather than discrete units)
Micro-organisms	Chemical gradients as memory	Single cells with receptors	Basic processing via chemical reactions	Adaptation rather than learning	High adaptability, rapid response to environmental changes	Negligible	N/A (memory through chemical gradients)

to and even manipulate their environment. The power consumption for this remarkable cognitive activity is approximately 20 Watts.

Other mammals and birds also exhibit significant cognitive abilities with memory capacity ranging from gigabytes to terabytes. They also possess spatial memory and problem-solving skills, and have complex social behaviors. Birds have specialized processing and spatial memory that help them excel in navigation and mimicry. Insects have fewer neurons, but are able to work collaboratively to create collective intelligence with a memory capacity in the range of megabytes to gigabytes. In plants, the intelligence is less neuron-like, instead, the memory and adaptability are exhibited through chemical signaling and environmental responsiveness.

1.2 Comparison with Machines

When the human brain develops over time, the memory capacity, the neural connections, and the ability to learn new information also increase. Figure 1.1 shows that a 1-year-old human brain consumes around 15 Watts of power, and by adulthood, say 20 years, the human brain consumes around 20 Watts of power. The learning ability and complex

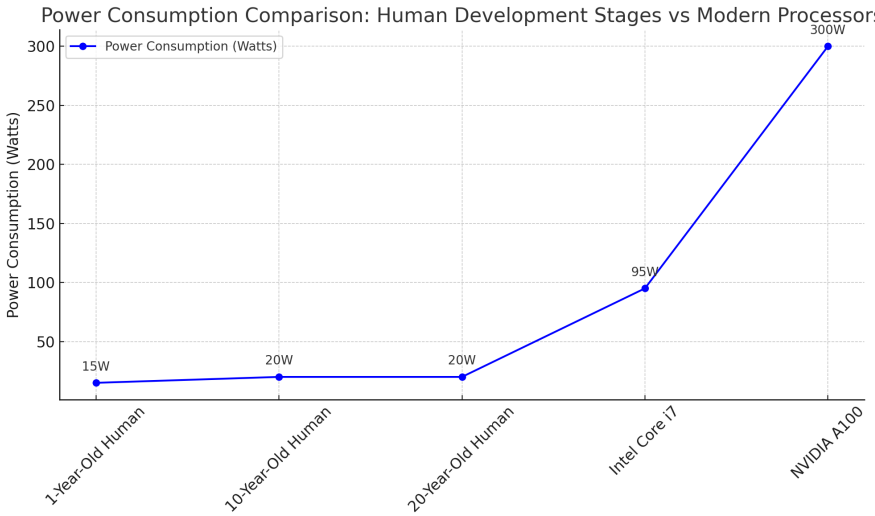

Figure 1.1: Comparison of power consumption.

cognitive skills developed dramatically over these years. On the other hand, many of the modern processors, for example the Intel Core i7 consume approximately 95 Watts, and GPUs like the NVIDIA A100 consume around 300 Watts. In comparison to the human brain, they are inefficient and do not pose the abilities of learning, adaptation, or generalization.

1.3 Generalisation and Bio-inspiration

The ability of human intelligence to generalize information and knowledge for multiple tasks and situations makes it superior to all different forms of AI that exist today. The underlying mechanisms of neural plasticity, pattern recognition, and abstraction play a pivotal role in the intelligence ability. Because of neural plasticity, it is possible for the brain to adapt to new situations, create new neural connections, and thereby reorganize the networks to perform complex tasks. This flexibility is essential when put in new and unfamiliar environments. Pattern recognition is a fundamental process in the human brain that allows patterns to be detected and relationships between patterns to

be established, drawing conclusions for higher cognitive tasks. The ability to abstract concepts and situations based on partial information or relationships between information allows for the development of generalization abilities.

Analogical reasoning involves the brain drawing similarities between experiences that help with solving problems in a creative way. When put in a society, the learner gets reinforced through the experience of others. Curiosity to learn new experiences or concepts frequently sparks new experiences, which in turn improves people's capacity for generalization. The fact that memories are at the core of every neural network means that these experiences are stored in neural networks, which also drive the experiences to be associated with emotions, thereby making emotional intelligence an important aspect of a learning brain. Aspects such as empathy and context sensitivity are critical in social contexts and help adapt the intelligent being to be inclusive in society. Together, these components enable human intelligence to learn, adapt, and generalize at a scale that is not possible today with AI.

Because human intelligence is far ahead in terms of energy efficiency, function, and generalization ability, it becomes important to draw inspiration from it. This enables us to build systems that are better in design and eventually make the AI systems capable of more generalized intelligence. The human brain is well known for generalizing across diverse contexts with limited data, while the present state-of-the-art AI systems, like deep neural networks, use vast amounts of training data to achieve limited generalization capabilities. Understanding the neural plasticity in the brain can help develop algorithms and hardware that adapt dynamically to new inputs and make AI models flexible. In deep artificial neural networks, plasticity mechanisms can be incorporated by dynamically updating weights based on new data or experiences without the need to do extensive retraining.

Beyond generalization, the ability to extract abstract information from complex inputs, i.e., analogical reasoning, is considered a hallmark ability of the human brain. By designing algorithms and architectures that can seamlessly find the similarities between information and apply these to new experiences or situations, it becomes possible to solve uncharted complex problems. Analogical problem-solving can get help

from bio-inspiration, as integrating perception, memory, and learning processes from natural systems into computational models can enable context-aware adaptability. Profound characteristics such as emotional learning, curiosity, and exploration are inevitable if AI has to find acceptance in human society, thereby making it necessary for the AI system design and hardware to be compatible for handling uncertainty and making nuanced decisions in unforeseen environments.

2

Bio-inspiration to Memristors

Along with the resistor, capacitor, and inductor, Leon Chua proposed the **memristor** as the fourth fundamental circuit component in 1971 [13]. It is a two terminal non-linear resistor that can be programmed to take a desired resistance value. By applying simuli, such as voltage pulses of a certain amplitude and period, one can program the memristor to a target conductance value. There are many two terminal resistive devices that can be mapped to the theory of memristors, possessing resistive and memory behavior [10, 16, 33, 43]. By continuously applying a series of voltage pulses, the conductance can be varied as a function of the history of the current that has flowed through it. This property shows the ability of memristors to be state-dependent and show biomimicry of adaptive learning, i.e., emulating a biological synapse that responds to the strength of previous activity [75].

2.1 Generalized Memristor State Equations

For a generalized memristor [3, 13, 14, 17], the state of the device can depend on one or more state variables, often denoted by a state vector $\mathbf{x}$. Different types of memristor implementations have different physical mechanisms and dependence, making the state variable determination

something that is specific to the type of memristor device. In general, the state equations are given as follows:

1. **Voltage-Current Relationship:**

$$V(t) = M(\mathbf{x}, I) \cdot I(t). \tag{2.1}$$

Here, $V(t)$ is the voltage across the memristor, $I(t)$ is the current flowing through memristor, and $M(\mathbf{x}, I)$ is the memristance (or memristor-resistance), which is a function of both the current state variables $\mathbf{x}$ and the current $I(t)$. This equation can also be called state-dependent Ohms law.

2. **State Variable Dynamics:**

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, I). \tag{2.2}$$

The state variable $\mathbf{x}$ represents the internal state of the memristor, that could be related to physical properties such as ion positions or internal charges. The function $f(\mathbf{x}, I)$ shows the dependency of internal state over time, along with dependence on the current $I(t)$ and the existing state $\mathbf{x}$.

2.2 System Behavior of a Memristor

We can represent the memristor device as a state equation showing dynamic system behaviour. Depending on the nature of the inputs and functional dependencies, various system behaviors can occur including stability, instability, chaos, and the edge of chaos.

- **Stable Behavior:**

The system is considered stable when small perturbations of state variable x decay over time such that the system returns to its equilibrium state. This means that the system is stable if eigenvalues of the Jacobian matrix of $f(x, I)$ have negative real parts. To demonstrate an example, a memristor with the following state equation can be considered:

$$\frac{dx}{dt} = -\alpha x + I(t), \tag{2.3}$$

where $\alpha > 0$ is a constant. The negative term $-\alpha x$ indicates that any form of deviation in x will decay over time, which in turn leads to stability. The system will be returning to equilibrium state if perturbed, thus exhibiting a stable behavior.

- **Unstable Behavior:**

  Contrary to a stable system, the system becomes unstable when the small perturbations grow over time, causing the system state to diverge. Mathematically, this occurs when eigenvalues of the Jacobian matrix of $f(x, I)$ have positive real parts. For example, consider the memristor with the state equation:

$$\frac{dx}{dt} = \beta x + I(t). \tag{2.4}$$

  Here, $\beta > 0$ is a constant. When the positive term βx causes small perturbations in x and it grows over time, that results in an unstable system. The state in this case thus diverges rather than returns to equilibrium.

- **Chaotic Behavior:**

  When the state evolution of a system is highly sensitive to the initial state of the system, it can lead to unpredictable dynamics and thereby chaotic behavior. In memristive systems, this can often happen if the function $f(x, I)$ is non-linear and contains feedback mechanisms that create complex attractors, where the chaos can be characterized by a positive Lyapunov exponent. A memristor can exhibit chaotic behavior with a non-linear state equation such as:

$$\frac{dx}{dt} = \gamma x(1 - x) + I(t)\sin(x), \tag{2.5}$$

  where γ is a parameter that controls the system dynamics. The non-linear interaction and dependency between x and $\sin(x)$ introduces a feedback that can lead to chaotic behavior, which is characterized by sensitivity to initial conditions and complex, unpredictable dynamics.

- **Edge of Chaos:**

  In many bio-inspired systems, often the behavior switches from being stable to exhibiting chaos. In system dynamics studies, this is known as the *edge of chaos* [15], a transitional region between order and chaos where the system can exhibit both stable and chaotic dynamics. When the system parameters are finely tuned such that it neither falls into total randomness nor settles into a static state, the edge of chaos is observed. For example, consider the state equation:

  $$\frac{dx}{dt} = \delta x(1-x) + I(t)(1-x^2), \tag{2.6}$$

  where δ is a finely tuned parameter. In this case, the dynamics of the system are balanced between stable and chaotic states, placing it at the *edge of chaos*. Such a system can exhibit both structured and unpredictable behaviors, making it highly adaptable and similar to many biological systems.

2.3 Comparison Between Memristor and Biological Neuron

The reason why the memristor grew in prominence is its ability to be linked to the behaviour of biological neurons, among others. To explore this understanding, let us consider a HP memristor with the following state equations.

2.3.1 State-Dependent Ohm's Law for HP Memristor

The memristor can be modeled as a current controlled device or as a voltage controlled device. HP came up with a memristor device that is based on TiO_2 with Pt electrodes, as a two terminal filamentary device [64]. Such a device can be modeled, with a state equation, relating voltage and current across the device as:

$$V(t) = R(w,t) \cdot I(t), \tag{2.7}$$

here, $R(w,t)$ is the memristance that uses a state variable $w(t)$ representing the position of the boundary between the doped and undoped regions of the titanium dioxide layer.

2.3.2 State Equation for HP Memristor

The state equation for such memristor is described by:

$$\frac{dw}{dt} = \mu_v \frac{R_{on}}{D} I(t) \tag{2.8}$$

Where:

- $w(t)$: State variable representing the position of the boundary between doped and undoped regions.
- μ_v: Mobility of oxygen vacancies.
- R_{on}: Resistance of the doped region.
- D: Total width of the titanium dioxide layer.

The memristor's resistance changes based on the evolution of the state variable depending on the current and the physical properties of the device.

2.3.3 Biological Neuron State Equations (Hodgkin-Huxley Model)

The Hodgkin-Huxley model [49] is a popular way to model the biological neuron looking at the membrane potential dynamics.

2.4 Membrane Potential Dynamics

The dynamics of the membrane potential of the neuron can be modeled as:

$$C_m \frac{dV_m}{dt} = -(I_{\text{Na}} + I_{\text{K}} + I_{\text{L}}) + I_{\text{ext}}, \tag{2.9}$$

where:

- C_m: Membrane capacitance.
- V_m: Membrane potential.
- $I_{\text{Na}}, I_{\text{K}}, I_{\text{L}}$: Ionic currents through sodium, potassium, and leak channels respectively.
- I_{ext}: External input current.

The sodium current (I_{Na}), i.e., the ionic currents are dependent on the membrane potential and follow gating dynamics that change over time.

$$I_{\text{Na}} = \overline{g}_{\text{Na}} m^3 h (V_m - E_{\text{Na}}), \tag{2.10}$$

where:

- $\overline{g}_{\text{Na}}$: Maximal conductance of sodium.
- m, h: Gating variables controlling sodium channels.
- E_{Na}: Sodium equilibrium potential.

2.5 Comparison Between HP Memristors and Biological Neurons

There are several parallels we can draw between HP memristors and biological neurons [14], some of which are listed below.

2.5.1 State Variables

The state variable for HP memristor $w(t)$ represents the position of the boundary between doped and undoped regions, which determines the resistance of the device. In the biological neuron model, state variable is membrane potential V_m, which is also dependent on gating variables m, h, n, that control the behavior of ion channels.

2.5.2 External Input

The current input $I(t)$ is the external stimuli for the HP memristor model shown that affects the position of the boundary and, therefore, the memristance. In the biological neuron, input current I_{ext} controls the changes in the membrane potential that affect the gating variables and the ionic currents.

2.5.3 Non-linearity and Feedback

Both models show non-linear dynamics that are useful for implementing learning functions. In the memristor model, the resistance changes

following the state dependency, based on the history of current and voltage and zero-crossing hysteresis $i - v$ behavior. While in biological neurons, non-linearity results from voltage-gated ion channels, whose opening and closing depend on the membrane potential, resulting in complex behaviors like action potentials, rhythmic firing, and chaotic spiking patterns.

2.5.4 Memory and Adaptation

Using the resistance as a state-dependent memory element can be seen as having a memory of past input currents. This is similar to the synapses that adapt their strength based on prior stimuli. On the other hand, the biological neurons use synaptic plasticity for memory behavior. Here, the synaptic strength changes based on the history of neural activity (e.g., long-term potentiation, LTP).

In these examples, both HP memristors and biological neurons demonstrate state-dependent behavior, non-linearity, and memory. This makes them well-suited to neuromorphic computing. The biological model is a far more complex model to implement than the memristor model. The simplistic yet powerful dynamics of memristors can be used for practical implementation of synaptic functions. Understanding these similarities and differences can set a direction for the development of energy-efficient neuromorphic systems inspired by biological intelligence.

3

Memristor Models

3.1 Strukov's Linear Drift (HP Memristor) Model

The **Strukov model** (HP's TiO_2 memristor) is a physical memristor model that treats the device as a two-resistor region with a moving boundary of oxygen vacancies (dopants) [64]. The doped region having a width w has low resistance R_{ON} and the undoped region has high resistance R_{OFF}. The total memristance is given by:

$$R(w) = R_{\text{ON}}\frac{w}{D} + R_{\text{OFF}}\left(1 - \frac{w}{D}\right),$$

where D is the total thickness. The ionic drift in a uniform electric field is assumed to follow

$$\frac{dw}{dt} = \mu_v \frac{R_{\text{ON}}}{D}\, i(t),$$

with μ_v representing the average ion mobility. This can be seen as a charge controlled memristor that remembers the device resistance to the history of charge flow.

This linear drift model provided the foundational theory for using memristors in non-volatile memory (RRAM). The resistive switching is explained as continuous change in resistance induced by voltage-driven dopant drift. Its accuracy to a real device is limited by the unrealistic

assumption of constant dopant drift velocity at the nanoscale and the absence of built-in mechanisms to prevent the state variable w from exceeding its physical limits (i.e., 0 or D). These limitations have led to the development of more advanced models incorporating nonlinear drift and boundary-locking mechanisms.

3.2 Joglekar's Window Function Model

The **Joglekar model** builds on top of the linear drift memristor by adding a window function that accounts for nonlinear dopant drift near the device boundaries [32]. The state equation is modified as

$$\frac{dw}{dt} = k\, i(t)\, f(x),$$

where $x = \frac{w}{D}$ is the normalized state, k is a constant, and the window function is defined as

$$f(x) = 1 - (2x - 1)^{2p},$$

with p controlling the nonlinearity which can take positive integer values. For moderate values of p, the function slows as w approaches 0 or D, while for $0 < x < 1$ it approximates unity. This means that dopant drift reduces near the boundaries similar to physical devices.

The damping of the drift in the boundary ensures that the state variable is within physical bounds. While this improves simulations, once the state variable reaches the extreme boundaries (0 or 1), the window function becomes zero. This limitation means that state changes are inhibited, even if the input polarity is reversed. The asymmetric drift phenomena observed in real devices is not captured in this model.

3.3 Biolek's Window Function Model

The **Biolek model** uses the window function to take account of dependence on the direction of current flow to address the issue of "stuck states"[5, 6]. The window function is defined as:

$$f(x, i) = 1 - |x - \mathrm{sgn}(-i)|^{2p},$$

where $\mathrm{sgn}(i)$ distinguishes the polarity of the current.

The positive current drives the device toward the ON state, then the function approaches zero at $x = 1$ but not at $x = 0$ and for negative current, the vice opposite occurs. This makes it possible for state variable changes even at extreme value as the window function becomes nonzero upon polarity reversal.

As the Biolek model produces realistic pinched hysteresis loops during alternating current sweeps, they are useful for circuit simulations. The challenge is that current direction inclusion in the model leads to discontinuities in the state derivative at the moment of current reversal causing numerical issues during simulations.

3.4 Simmons Tunnel Barrier (Pickett) Model

The **Simmons tunnel barrier model** (often also known as Pickett model) uses a physics based approach assuming the memristor as a metal–insulator–metal structure with a tunneling gap whose width serves as the state variable [70]. This model uses a tunneling resistance that depends exponentially on the barrier width w, and the tunnelling current can be expressed as:

$$i \approx A \sinh(\alpha V) \exp(-\beta w), \tag{3.1}$$

where A, α, and β are parameters fitted to the material properties. The state equation will capture how the oxygen vacancy drift (or filament growth/rupture) modulates the barrier width under applied bias. The differential equations for SET and RESET processes, incorporating threshold currents i_{ON} and i_{OFF} to reflect rapid state transitions, is developed based on this. This model provides excellent accuracy in reproducing the measured I–V characteristics and switching dynamics of physical memristive devices. However, the model's complexity, with many parameters and coupled exponential terms, leads to numerical instability in circuit simulations. The parameter adjustment is often needed to tailor to a specific device type, which limits its generality despite its high fidelity in representing physical behavior.

3.5 Yakopcic Model (Threshold Adapted Empirical Model)

The **Yakopcic model** incorporates voltage thresholds and asymmetric switching dynamics to replicate the "hard-switching" behavior observed in experiments [74]. This is a behavioral model, with the state variable x normalized between 0 and 1, having the I-V relation as:

$$I(t) = \begin{cases} a_1\, x(t) \sinh(b\, V(t)) & \text{for } V(t) > 0, \\ a_2\, x(t) \sinh(b\, V(t)) & \text{for } V(t) < 0, \end{cases} \tag{3.2}$$

where a_1 and a_2 scale the current amplitudes and b controls the nonlinearity. The evolution of the state can be shown as:

$$\frac{dx}{dt} = g(V) \cdot f(x), \tag{3.3}$$

where $g(V)$ introduces voltage thresholds V_P (for SET) and $-V_N$ (for RESET), ensuring that $g(V) = 0$ below these thresholds. The function $f(x)$ is the window function that modulates the rate of change as x approaches its limits.

The model is practically useful for simulating digital logic and memory operations as the asymmetric switching observed in many devices is accurately captured. The option for specifying thresholds and dynamics for SET and RESET processes makes it adaptable to multiple memristor technologies. However, parameter tuning is needed to ensure a match with the experimental data. The higher-order effects such as temperature dependence or stochastic variations are usually not fully captured without additional modifications.

3.6 VTEAM Model (Voltage Threshold Adaptive Memristor)

The Voltage Threshold Adaptive Memristor (**VTEAM**) model provides a generalized memristor model based on voltage-controlled thresholds [36]. VTEAM defines the state evolution in a piecewise manner according to the applied voltage $v(t)$, where the state variable w remains unchanged when $|v(t)|$ is below the set (V_{on}) or reset (V_{off}) thresholds. When $v(t)$ exceeds thresholds, the state variable changes following the equation:

$$\frac{dw}{dt} = \begin{cases} k_{\text{on}} \, (v(t) - V_{\text{on}})^{\alpha}, & v(t) > V_{\text{on}}, \\ k_{\text{off}} \, (v(t) - V_{\text{off}})^{\alpha}, & v(t) < V_{\text{off}}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

The I-V relation can be adjusted to be either a linear or nonlinear $v = R(w)i$ depending on the device under consideration. This model is popular for circuit simulations due to its simplicity and versatility. The parameters, threshold voltages and switching coefficients make it easy to calibrate using experimental data, and are useful for large circuit simulations. Since this is a phenomenological model, it does not capture the details of analog switching dynamics or gradual state changes below the nominal thresholds.

3.7 Energy-efficient Computation Using Memristors

Memristor-based circuits, such as crossbar arrangements of the devices, hold significant potential for **energy-efficient computation** [29]. It essentially combines the idea of memory with computation through such network arrangements, thereby avoiding the need to have separate computational units and memories.

There are many ways in which the resistive switching memristor devices could be used for creating energy-efficient computing systems. Some examples are listed below:

- **Non-Volatile Memory:**

  Most popular memristors are non-volatile, i.e., they can retain their state without the need for continuous power supply. As opposed to DRAM or SRAM, as a memory, an idealistic memristor will need relatively less area on chip and can be used as a discrete analog memory. This increases the memory density as well as the number of peripheral circuits needed. In practice, the memristor is far from ideal, as it has fewer numbers of discrete states and slower write cycles, making it not yet suitable for replacing DRAM or SRAM in traditional systems. However, as an analog memory, memristors in specific applications such as neural architectures will have advantages over area and power requirements.

- **Analog Computation and Neuromorphic Systems:**

  Neuromorphic systems are characterized by artificial synapses that contain analog computations and spiking networks. Much like the brain, the spike-based networks can be efficiently implemented with memristor networks. The processing and memory are integrated into the memristor networks; thereby, this *in-memory computing* reduces the energy cost associated with data communications between memory and processing nodes.

- **Massive Parallelism:**

  Being an analog or discrete memory element that can be programmed to a large number of resistance states, it is possible to create high-density arrays capable of massively parallel computations. Such a way of computation is suitable for *pattern recognition* and *machine learning*, enabling efficient, low-energy implementations of algorithms.

  Crossbar configurations can be used for implementing multiply and accumulate operations, which is at the heart of many neural networks and machine learning techniques. These can be translated to *matrix-vector multiplications* when in a crossbar array. In comparison, digital processing requires the need for digital arithmetic logic units, and the crossbar performs matrix multiplications with latency and overhead of data transfers. This makes the memristor-based computation energy-efficient compared to digital processing.

The dynamic system properties and uses of the memristor place it as a promising device for developing hardware that can perform **general-purpose AI computation** more efficiently, where energy consumption is a critical issue. Drawing inspiration from the biological synapses, memristors can provide a foundation for developing newer neuromorphic systems that are able to learn and generalize information in a manner similar to human brains but with significantly lower energy requirements.

4

Tutorial on Memristor Crossbar with Open-source Skywater PDK

The practical implementation of memristor circuits requires the use of foundry ready models. Since memristors are still an experimental device in many of the semiconductor foundries today, it is important to have a sense of how to go about implementing them for building practical solutions. ReRAMs (Resistive RAMs) form the closest practical implementations of memristor models. Skywater PDK provides an open, free-to-use model that is supported for fabrication in a 130nm technology. A tutorial on how to install the open-source software tools for design is shown on the Makerchips GitHub website.[1] The tutorial provided in this section is a guide for using open-source tools. Although Skywater PDK is used as an example, there is a wide effort in the scientific community to build open-source alternatives with many foundries beyond the Skywater foundry.

This section provides the list of steps to be followed to install the software tools, followed up with integrating the ReRAM models, and further an example of how to create a crossbar circuit.

[1] https://github.com/Makerchips-OSCM

4.1 Installation Instructions

Start with a Linux system with version: **Ubuntu 22.04.4 LTS** (fresh installation).

4.1.1 Step 1: System Update

It is necessary to update the system. Open a terminal in the desktop/home directory and run the command:

```
sudo apt-get update
```

4.1.2 Step 2: Specific Folder Creation

Create a new folder for installing all the required tools:

```
mkdir -p <Installation_directory_name>
cd <Installation_directory_name>
```

Example:

```
mkdir -p chip_design
cd chip_design
```

4.1.3 Step 3: Pre-requisites

Install all the necessary supporting libraries for the installation of open-source tools for Ubuntu LTS version:

```
sudo apt install git
sudo apt-get install build-essential clang bison flex
sudo apt-get install libreadline-dev gawk tcl-dev libffi-
    dev
sudo apt-get install git graphviz xdot pkg-config python3
sudo apt-get install libboost-system-dev libboost-python-
    dev
sudo apt-get install libboost-filesystem-dev zlib1g-dev
    make m4
sudo apt-get install tcsh csh libx11-dev gperf tcl8.6-dev
     tk8.6
sudo apt-get install tk8.6-dev libxmp4 libxpm-dev libxcb1
sudo apt-get install libcairo2 libxrender-dev libx11-xcb-
    dev
sudo apt-get install libxaw7-dev freeglut3-dev automake
    yosys
```

```
sudo apt-get update
sudo apt-get -y install libtool
```

Check whether `.local/bin` is included in `$PATH`:

```
echo $PATH | grep '\.local'
```

If it is not there, add it to your path before starting the install by adding this to your `/.bashrc` file:

```
export PATH="$HOME/.local/bin:$PATH"
```

4.1.4 Step 4: Installing Xschem Tool

To install `xschem`, clone the required repository from GitHub:

```
cd ~/<Installation_directory_name>/
git clone https://github.com/StefanSchippers/xschem.git
    xschem
cd xschem
./configure
make
sudo make install
```

To run `xschem`:

```
xschem
```

4.1.5 Step 5: Installing Magic Tool

To install `magic`, run the following commands:

```
cd ~/<Installation_directory_name>/
git clone git://opencircuitdesign.com/magic
cd magic
./configure
make
sudo make install
```

4.1.6 Step 6: Installing Ngspice Tool

Before installing `ngspice`, update the system:

```
sudo apt-get update
```

Then run the following commands:

```
wget https://sourceforge.net/projects/ngspice/files/ng-spice-rework/43/ngspice-43.tar.gz/download
# Extract the downloaded file
sudo apt-get install unzip  # If unzip is not installed
unzip ngspice-43.tar.gz
cd ngspice-43
./configure
make
sudo make install
```

Check the version:

```
ngspice --version
```

4.1.7 Step 7: Installing Open_PDKs

To install `open_pdks`:

```
sudo apt-get update
git clone git://opencircuitdesign.com/open_pdks
cd open_pdks
./configure --enable-sky130-pdk
sudo make
sudo make install
```

Copy the `.magicrc` file:

```
cd ~
cd ~/.xschem
cp ~/<Installation_directory_name>/open_pdks/sky130/sky130A/libs.tech/xs/xschemrc
cd ~
cp ~/<Installation_directory_name>/open_pdks/sky130/sky130A/libs.tech/ma/sky130A.magicrc ~/.magicrc
sudo apt -y install vim-gtk3 xterm
```

Step 8: Installing LVS and Klayout

To install LVS tool:

```
sudo apt-get install netgen-lvs
```

To install `klayout`:

```
sudo apt-get install klayout
```

If not installed properly, download the latest Klayout and install manually:

```
sudo apt install gcc ruby-full
sudo apt-get install libqt5opengl5 libhttp-parser2.9
    libmbedcrypto7 libmbedtls14 libmbedx509-1 libssh2-1
    libgit2-1.1
sudo dpkg -i klayout_0.29.5-1_amd64.deb
sudo apt --fix-broken install
sudo dpkg -i klayout_0.29.5-1_amd64.deb
```

4.2 Integrating ReRAM with Xschem

Once the installation of the software tools is completed, we now need to integrate the ReRAM (Figure 4.1) to Xschem (Figure 4.2). For this, ensure that you have the latest ngspice version.

Git clone the following repository: https://github.com/barakhoffer/sky130_ngspice_reram.

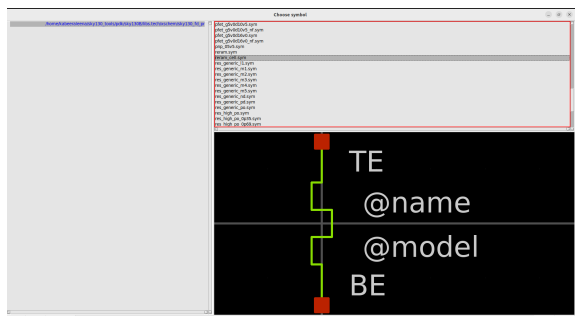

Figure 4.1: Symbol of ReRAM cell.

The location mentioned in Figure 4.3 should be the location you select for the ReRAM cell as shown in Figure 4.4. Figure 4.5 shows the simulation command for plotting the I-V characteristics of ReRAM and Figure 4.6 shows the generated plot, indicating the pinched hysteresis characteristics.

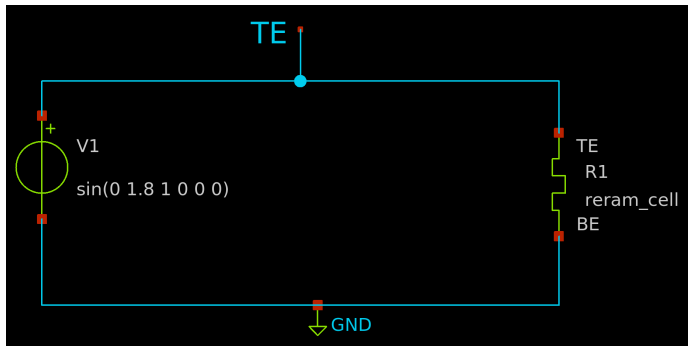

Figure 4.2: Circuit diagram for simulating a ReRAM cell in Xschem.

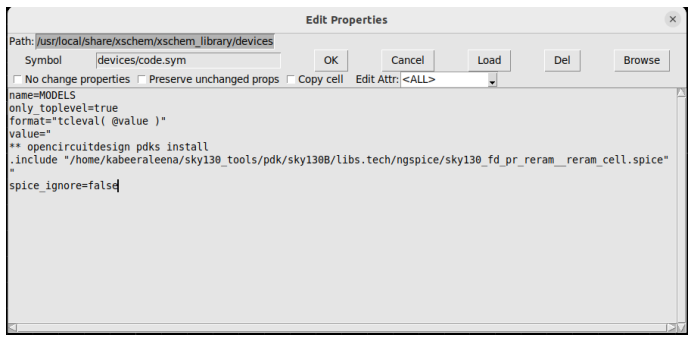

Figure 4.3: Model of ReRAM cell included for simulation in Xschem.

4.3 Layout of ReRAM

The layout of a ReRAM drawn with the help of the open-source tool magic is shown in Figure 4.8. It should be noted that in Skywater130 PDK ReRAM material lies in between the layers metal1 and metal2. The top electrode is connected to metal2, and the bottom electrode is connected to metal1. Figure 4.7 shows the commands and steps involved in drawing the layout of the ReRAM cell in magic. Figure 4.8 shows the layout in magic for the commands executed.

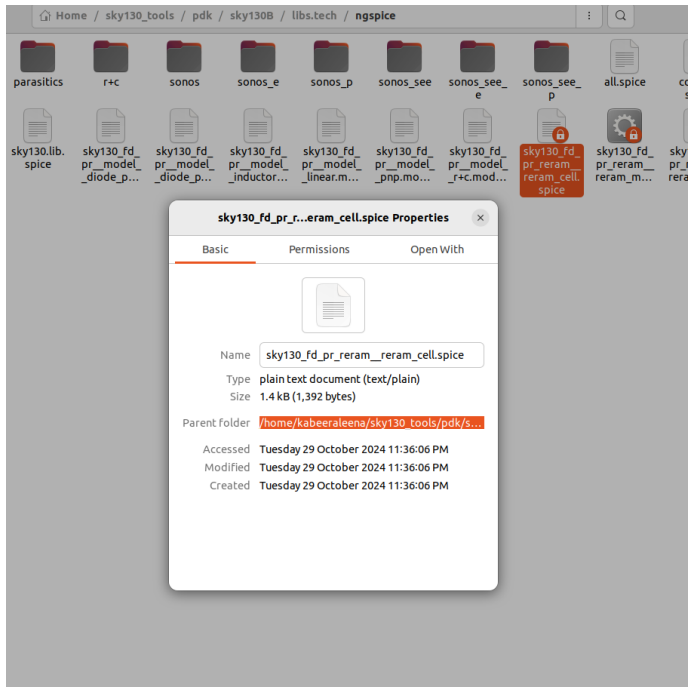

Figure 4.4: Location of ReRAM cell.

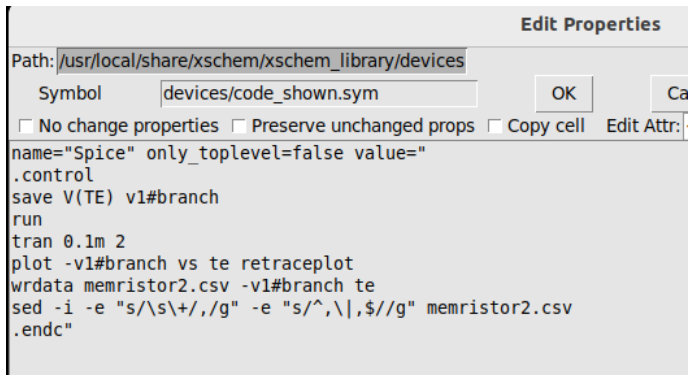

Figure 4.5: Simulation command for plotting I-V characteristics of ReRAM.

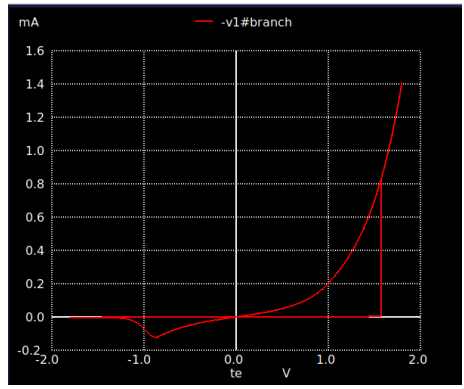

Figure 4.6: I-V characteristics of ReRAM.

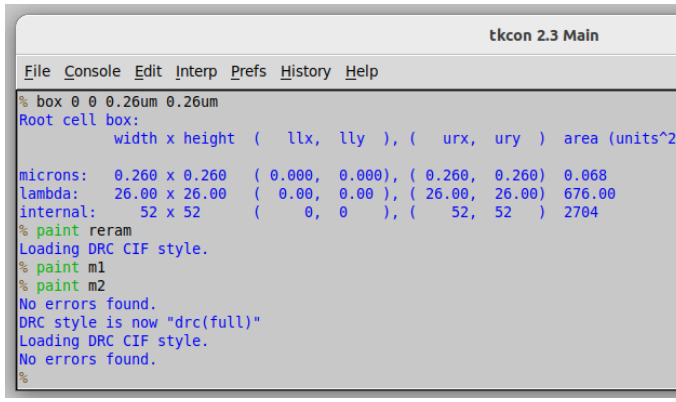

Figure 4.7: Steps of drawing layout of a ReRAM cell in magic.

4.4 Crossbar Array

When it comes to placing memristors (or ReRAMs) in the crossbar configuration, to avoid the sneak path currents and to ensure better control of programming each memristor cell, a selector or transistor switch is connected in series to the memristor (1T-1R cell).

The schematic of the single 1T-1R cell drawn is shown in Figure 4.9. The layout of the same is shown in Figure 4.10. Using these basic cells, they can be placed in a crossbar configuration as shown in Figure 4.11, with the layout of the crossbar shown in Figure 4.12.

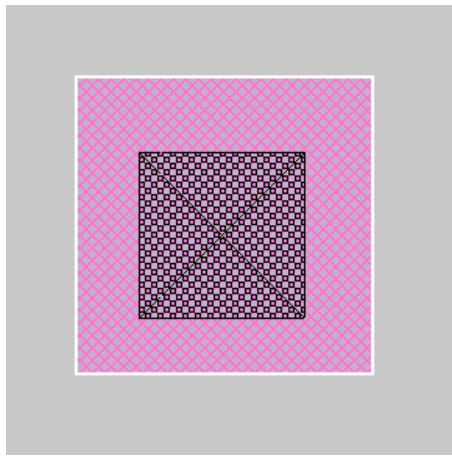

Figure 4.8: Layout of a ReRAM cell.

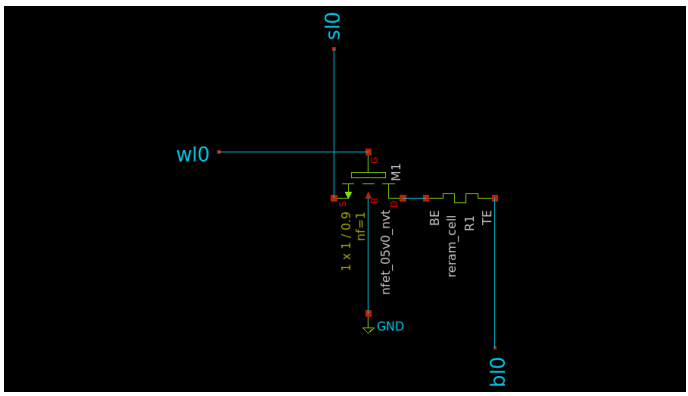

Figure 4.9: Schematic of a 1T1R unit cell.

4.5 Beyond Crossbar

Once the crossbar array is constructed, to make this a functional array for multiply and accumulate operations, there are several other circuit blocks to be built. We call these blocks peripheral blocks, such as that is needed to program the memristors, sense amplifiers to read the column currents, and the necessary activation function blocks for converting each of the columns to a neuron. In addition, one needs to also consider the data path logic circuits for input and outputs from the crossbars,

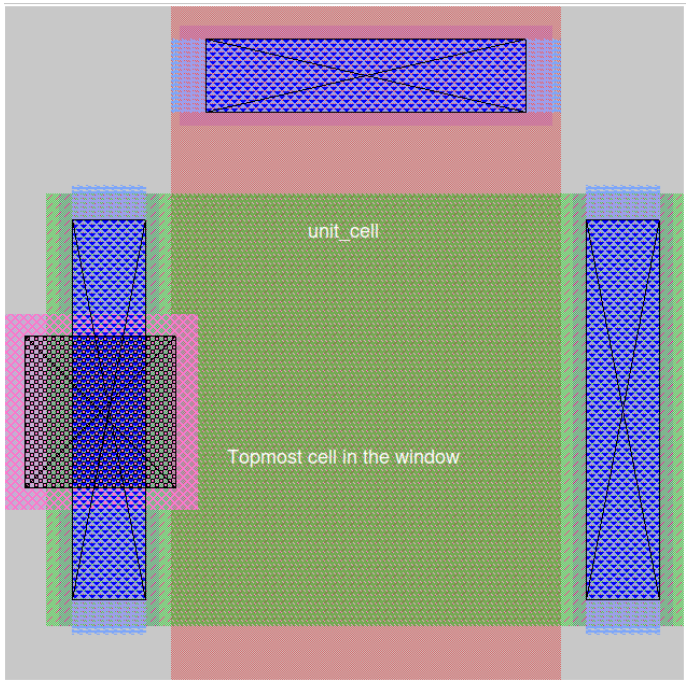

Figure 4.10: Layout of a 1T1R unit cell.

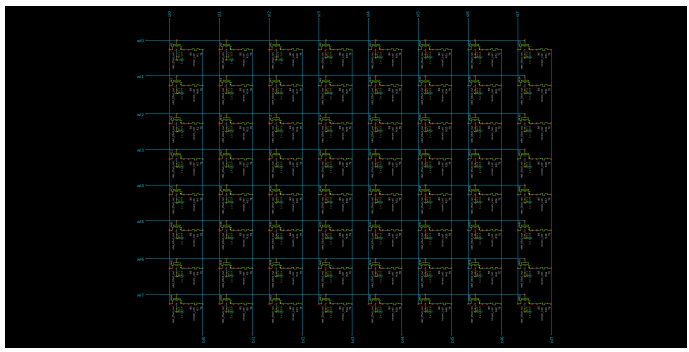

Figure 4.11: Schematic of an 8×8 crossbar array.

especially when implementing the neural networks, such considerations becomes critical.

The discussion on this topic is incomplete without hinting on extending this discussion to the manufacturing aspects, where these PDKs

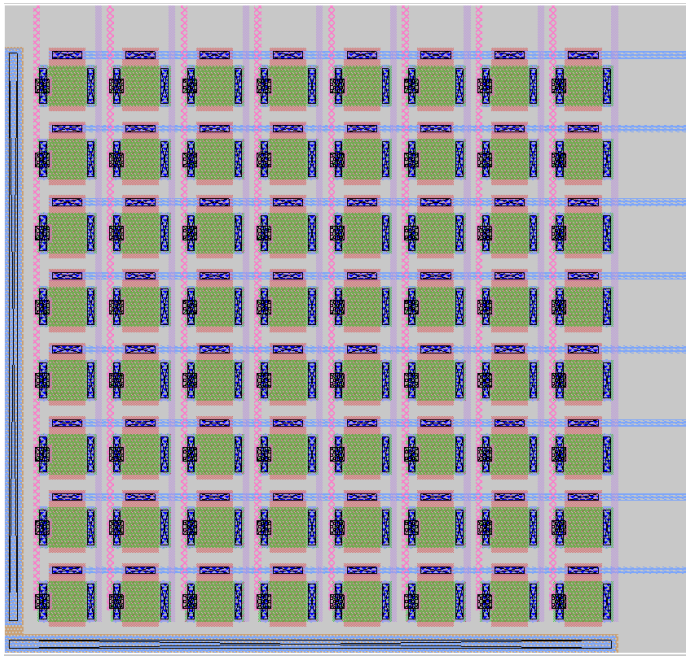

Figure 4.12: Layout of an 8×8 crossbar array magic.

translate to the SkyWater's CMOS fabrication facility. The foundry in this case supports integrating memristor devices onto silicon wafers. Precise process optimisation is needed to integrate memristors beyond typical CMOS processes. This includes uniform memristor device characteristics across the wafer showing reliability in the manufacturing process, minimizing variability in resistance states, and ensuring stable switching behavior across several operating cycles.

Atomic layer deposition (ALD) or sputtering that are controlled deposition techniques are needed to form reliable memristive oxide layers and electrode interfaces. Precise patterning, accuracy of lithography and optimal etching techniques are essential for ensuring uniformity and to help reduce the parasitic resistances. Backend-of-line (BEOL) processing steps to include memristor layers need considerations of thermal budget management for avoiding damage or alteration of underlying CMOS layers.

A multidisciplinary effort that combines device-level engineering, material optimization, and precise process control is required to develop reproducible memristor crossbar arrays. Translating the neural networks to high density, low power consumption, and intrinsic analog computation capabilities in hardware requires practical considerations in both design and manufacturing.

Part III

Applications

7

Making Use of Memristive Variability with Energy-Efficient Echo State Networks

Artificial Neural Networks (ANNs) emerged as computational models inspired by biological neural systems, initially appearing as feedforward networks where information flows unidirectionally from input to output layers. The fundamental building block, the artificial neuron, processes inputs through a weighted sum followed by a nonlinear activation function: $y = f(\sum_{i=1}^{n} w_i x_i + b)$, where w_i represents synaptic weights, x_i represents inputs, b is the bias term, and f is the activation function. While feedforward networks proved effective for static pattern recognition, they lacked the ability to process temporal sequences effectively. This limitation led to the development of Recurrent Neural Networks (RNNs), which introduce feedback connections, allowing the network to maintain an internal state. The basic RNN computation can be expressed as: $h_t = f(Wx_t + Uh_{t-1} + b)$, where h_t represents the hidden state at time t, W is the input weight matrix, U is the recurrent weight matrix, and b is the bias vector.

However, traditional RNNs suffered from vanishing and exploding gradient problems during training, making it difficult to capture long-term dependencies. Long Short-Term Memory (LSTM) networks addressed these issues by introducing a more complex memory cell

structure with gating mechanisms. The LSTM updates are governed by:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \text{ (forget gate)}, \tag{7.1}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \text{ (input gate)}, \tag{7.2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \text{ (output gate)}, \tag{7.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \text{ (cell state)}, \tag{7.4}$$

$$h_t = o_t \odot \tanh(c_t) \text{ (hidden state)}. \tag{7.5}$$

where σ represents the sigmoid function and $\odot$ denotes element-wise multiplication.

7.1 Reservoir Computing and Echo State Networks

Reservoir Computing (RC) uses an interesting concept of *reservoir* that is a fixed, high-dimensional dynamical system that transforms an input sequence into a feature space that can be linearly read out. Because of this, they are useful for time-series prediction tasks. To explore this machine learning framework, the model consists of three main components: an input layer, a reservoir, and an output layer.

7.1.1 Reservoir State Update

Considering this as a dynamical system, the reservoir state can be represented as a vector $\mathbf{x}(t) \in \mathbb{R}^N$ that can be updated based on current input $\mathbf{u}(t) \in \mathbb{R}^M$ and the previous reservoir state $\mathbf{x}(t)$, which can be mathematically summarized as:

$$\mathbf{x}(t+1) = f(\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t)), \tag{7.6}$$

where:

- $\mathbf{W}_{in} \in \mathbb{R}^{N \times M}$ is the input weight matrix that projects the input into the reservoir.
- $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the internal weight matrix that are the recurrent connections within the reservoir.
- $f(\cdot)$ is any nonlinear activation function, for example tanh or ReLU.

7.1.2 Output Computation

After the reservoir state $\mathbf{x}(t)$ is updated, the output $\mathbf{y}(t) \in \mathbb{R}^K$ need to be computed using a linear combination of the reservoir states, representing a dense neural network layer:

$$\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t), \tag{7.7}$$

where, $\mathbf{W}_{out} \in \mathbb{R}^{K \times N}$ is the output weight matrix that undergoes training.

7.1.3 Training Process

During training, only the output weight matrix $\mathbf{W}_{out}$ is updated using linear regression to minimize the mean squared error between the target output $\mathbf{y}_{target}(t)$ and the predicted output $\mathbf{y}(t)$:

$$\min_{\mathbf{W}_{out}} \sum_t \|\mathbf{y}_{target}(t) - \mathbf{W}_{out}\mathbf{x}(t)\|^2 . \tag{7.8}$$

The internal reservoir weights $\mathbf{W}$ and the input weights $\mathbf{W}_{in}$ are not updated or trained, making it computationally less expensive compared to a traditional neural network.

7.2 Example: Predicting a Sine Wave

As an example, let us consider the problem of predicting a sine wave. For the input sequence $\sin(t)$ the challenge is to predict the next value $\sin(t+1)$. The following are the steps for solving this using the reservoir computing:

1. **Define Input and Target Data**: The input is $\mathbf{u}(t) = \sin(t)$, and the target is $\mathbf{y}_{target}(t) = \sin(t+1)$.

2. **Initialize Reservoir**: Take an example of reservoir with $N = 100$ neurons, input weights $\mathbf{W}_{in}$ and recurrent weights $\mathbf{W}$ that are initialized randomly.

3. **State Update**: The reservoir state can be updated as:

$$\mathbf{x}(t+1) = \tanh(\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t)). \tag{7.9}$$

4. **Train Output Weights**: In the next stage, train $\mathbf{W}_{out}$ using linear regression to predict $\sin(t+1)$.

5. **Testing**: Testing involves using the trained model to predict future values of the sine wave.

7.3 Echo State Networks (ESNs)

Echo State Networks (ESNs) is an extended model of Reservoir Computing. ESN combines the idea of Reservoir Computing with the idea of feedback, which forms the *echo state property*. The addition of feedback can help model temporal dependencies in the input data, reducing the influence of the initial conditions on the reservoir state.

7.3.1 Mathematical Framework of Echo State Networks

Similar to the Reservoir Computing model, for ESNs the reservoir state is updated as:

$$\mathbf{x}(t+1) = \tanh(\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t) + \mathbf{W}_{fb}\mathbf{y}(t)), \tag{7.10}$$

where:

- $\mathbf{W}_{in} \in \mathbb{R}^{N\times M}$ is the input weight matrix.
- $\mathbf{W} \in \mathbb{R}^{N\times N}$ is the recurrent weight matrix of the reservoir.
- $\mathbf{W}_{fb} \in \mathbb{R}^{N\times K}$ is the feedback weight matrix that incorporates feedback from the output into the reservoir state.
- $\tanh(\cdot)$ is the nonlinear activation function.

The addition of the feedback weight matrix $\mathbf{W}_{fb}$ is the primary difference between ESNs and general RC models.

7.3.2 Training in Echo State Networks

The training in ESNs involves adjusting only the output weight matrix $\mathbf{W}_{out}$:

$$\mathbf{y}(t) = \mathbf{W}_{out}\mathbf{x}(t). \tag{7.11}$$

The training aims to minimize the error between the observed output $\mathbf{y}(t)$ and the target output $\mathbf{y}_{target}(t)$, which can be summarized as:

$$\min_{\mathbf{W}_{out}} \sum_{t} \|\mathbf{y}_{target}(t) - \mathbf{W}_{out}\mathbf{x}(t)\|^2 . \tag{7.12}$$

The internal weights $\mathbf{W}$, $\mathbf{W}_{in}$, and $\mathbf{W}_{fb}$ are not trained similar to Reservoir Computing models.

7.3.3 Summary of Echo State Networks

The concept of reservoir computing emerged as an alternative approach to traditional RNN training, introducing the idea of a fixed, randomly connected recurrent layer (the reservoir) coupled with a trainable readout layer. Echo State Networks (ESNs) represent a prominent implementation of this paradigm, characterized by their simple training procedure and powerful temporal processing capabilities. The architecture of an ESN is comprised as an input layer, a reservoir, and an output layer, as depicted in Figure 7.1

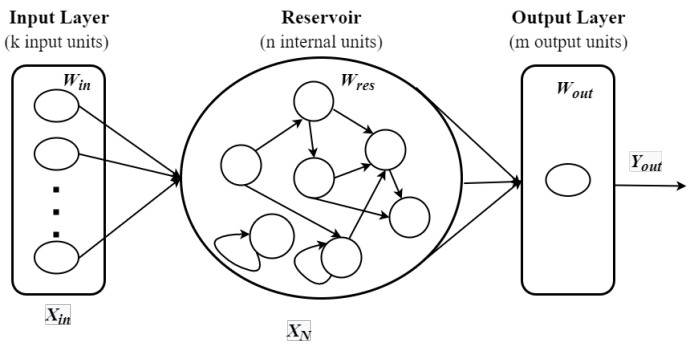

Figure 7.1: Echo state network architecture.

In an ESN, the reservoir state update is governed by:

$$x(t+1) = f(W_{in}u(t+1) + W_{res}x(t) + W_{fb}y(t)), \tag{7.13}$$

where:

- $x(t)$ is the reservoir state at time t
- $u(t)$ is the input signal

- $y(t)$ is the output signal
- W_{in} is the input weight matrix
- W_{res} is the reservoir weight matrix
- W_{fb} is the feedback weight matrix
- f is typically a hyperbolic tangent activation function.

In tasks where no output feedback is required, W_{fb} is nulled. The output is computed through a linear combination of reservoir states:

$$y(t) = W_{out}[1; u(t); x(t)], \tag{7.14}$$

where W_{out} represents the output weights, and $[1; u(t); x(t)]$ denotes the concatenation of a bias term, input signal, and reservoir states. The key property of ESNs is the echo state property, which requires the reservoir's dynamics to asymptotically depend on the input history rather than initial conditions. This is typically ensured by scaling the reservoir weight matrix W to have a spectral radius less than 1: $\rho(W) < 1$, where ρ denotes the largest absolute eigenvalue. Training an ESN primarily involves optimizing the output weights W_{out} while keeping the reservoir weights fixed. This can be accomplished through linear regression methods such as:

$$W_{out} = YX^T(XX^T + \beta I)^{-1}, \tag{7.15}$$

where Y is the target output matrix, X is the collected state matrix, β is a regularization parameter, and I is the identity matrix. The simplicity of training, combined with the rich dynamics of the reservoir, makes ESNs particularly suitable for temporal processing tasks while avoiding the computational complexity associated with training traditional RNNs or LSTMs.

7.4 Memristive ESN implementation

The reservoir weights can be randomly assigned without the need for any training. The memristor crossbar can be used to implement the reservoir, as it is a weight summation operation. This means that there

is no need for programming the memristors and devices, even with variability they will be suitable to be used for the reservoir weights. The crossbar networks can be used to build both the reservoir and the output network. Many different applications can be developed for practical use, even under the issues of parasitics and variability.

7.4.1 Application 1: Sampling ESN implementation

The internal reservoir weights are sampled from a given probability distribution at each time, and/or during the initialization phase, for Synaptic Sampling ESNs. This introduces variability in the reservoir. The state equation for Synaptic Sampling ESNs can be written as:

$$\mathbf{x}(t+1) = f(\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{W}_{syn}\mathbf{x}(t) + \mathbf{W}_{fb}\mathbf{y}(t)), \tag{7.16}$$

where:

- $\mathbf{W}_{syn} \in \mathbb{R}^{N\times N}$ is the synaptic weight matrix, with each element in it sampled from a probability distribution (e.g., Gaussian or Bernoulli).
- $\mathbf{W}_{in}$, $\mathbf{W}_{fb}$ is the input and feedback weight matrices as defined in traditional ESNs.
- $f(\cdot)$ is the typical nonlinear activation function such as tanh or ReLU.

The $\mathbf{W}_{syn}$ having the stochastic nature allows the Synaptic Sampling ESNs to have a wider range of dynamical behaviors. This can lead to better generalization and helps increase the expressiveness of the reservoir to a wider range of problems.

An ESN extensively uses the concept of randomness in its reservoir networks. In biological neural networks, synaptic stochasticity is something that is observed to play an important role for learning and adaptation. Embracing stochasticity, the memristive devices can be used for building an Extended Synaptic Sampling Machine (ESSM) [47]. The random number generation can be implemented by Circular Shift Registers (CSRs) that follow the Bernoulli distribution. This results in reduced memory requirements while maintaining processing capabilities.

Figure 7.2 shows the crossbar-based architecture that can be used to implement the sampling ESN. While $\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{W}_{syn}\mathbf{x}(t)$ is shown in the crossbar, a similar crossbar is used for $\mathbf{W}_{fb}\mathbf{y}(t))$ to complete the implementation. The columns of the network need to be further connected to the activation function circuitry to complete the circuit.

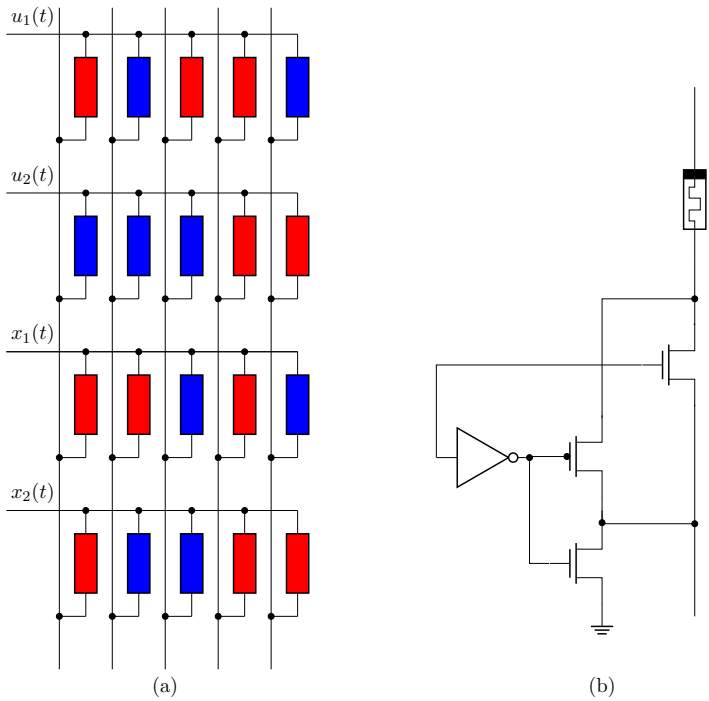

Figure 7.2: The circuit illustration in (a) shows the crossbar structure of implementing the ESN. The vectors $\mathbf{u}(t)$ and $\mathbf{x}(t)$ can be computed in the same crossbar. The red color indicates nodes that are OFF and blue indicates nodes that are ON as an example. The selector and sampling circuit for each node is shown in (b).

Mimicking biological neural plasticity, the synaptic sampling cells are designed to be adaptively reprogrammable. The memristive implementation of the ESN can be used for ECG analysis to complex image classification tasks like MNIST, Fashion MNIST, and CIFAR10. Noise sensitivity analysis of ESSM-ESN using different sampling rates on multiple datasets is as represented in Table 7.1. Given the high density of memristive arrays with CSR implementation, it offers advantages over

Table 7.1: Evaluating ESSM-ESN's robustness to noise at different sampling levels across four datasets: ECG, MNIST, Fashion MNIST, and CIFAR10 as reported by Nair *et al.* [47].

Dataset	Recognition accuracy					
	σ^2=0.01			σ^2=0.1		
	s=0%	s=20%	s=50%	s=0%	s=20%	s=50%
ECG	98.05	96.3	95.6	92.2	91.2	87.8
MNIST	96.42	93.87	84.9	84.54	83.0	80.62
Fashion MNIST	95.73	92.8	82.62	83.13	80.9	77.54
CIFAR10	92.45	85.4	74.62	79.0	75.68	68.71

traditional CMOS approaches, particularly in terms of power efficiency and integration density.

Given that the weight of the neural network is initiated randomly, factors such as resistance variations, noise, and quantization effects of memristive networks have very little impact on the overall performance of ESNs. The ESSM-ESN architecture demonstrates robustness in handling these variations while maintaining competitive performance metrics. In comparison with traditional neural networks, higher sparsity can be introduced in the network along with random weights that can reduce the power consumption with computational capability. Since the computations can be analog, the crossbars without the need for large compensation circuits form a realistic way to include biological plausibility with these networks. Since there is a better control over energy efficiency and variability, the circuit design challenges are reduced and they can be used in edge systems that need power efficiency and adaptive learning.

This architectural approach suggests new directions for neuromorphic computing, where stochastic elements are not just tolerated but leveraged to enhance system performance and efficiency. The success of this implementation strategy across diverse applications indicates its potential as a foundational approach for future neuromorphic systems.

7.4.2 Application 2: Novel Approach to Brain Tumor Detection: Integration of RF Sensing and Memristive Echo State Networks

One of the edge applications of ESN is to integrate classification and predictive analysis in wearable devices. Creating solutions that are low cost, easy to use and easily replaceable are important in continuous medical monitoring. The possibility of RF sensing for brain tumor detection and monitoring offers an inexpensive solution compared with traditional imaging methods like X-ray and MRI.

This work combines the radio frequency (RF) sensing technology with ESN implementations, as shown in Figure 7.3, to create an inexpensive and accurate solution for tumor detection. The concept utilizes a simple yet effective setup of dual antennas positioned around the head to monitor transmission coefficient variations. The changes in RF signal characteristics offer unique signatures on the state of tissue formations, including tumors, which allow for non-invasive and continuous monitoring.

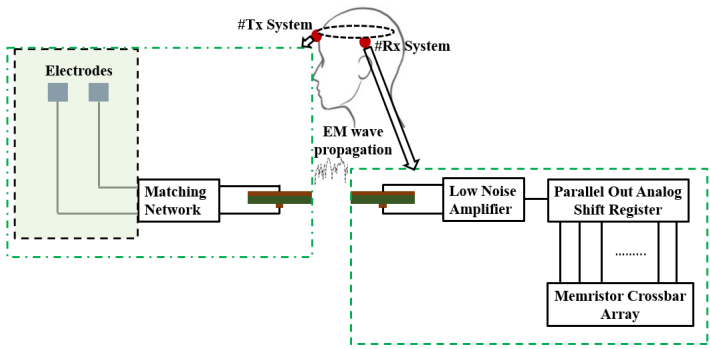

Figure 7.3: Tumor detection framework.

The RF sensors or antenna outputs are fed to memristive crossbar networks that implement Echo State Networks (ESNs) [46]. To simulate and analyse the classification performance memristor (VTEAM) models for the crossbar, realistic data is collected from the fabricated antenna on phantom setup of chicken meat with and without tumors. The dynamic nature of an ESN to be responsive to time-series problems, along with

efficient hardware implementation offered by memristive devices, offers a practical solution for building continuous monitoring wearable systems. We observe that even with the variability of memristive hardware, the system is able to detect brain abnormalities with the accuracy needed in real-time settings.

A low-cost alternative for fast diagnosis, the fact that the proposed solution can operate continuously and with minimal infrastructure requirements suggests potential applications in remote healthcare monitoring and early warning systems.

This approach opens new possibilities for developing more accessible and continuous health monitoring solutions, particularly valuable in regions with limited access to advanced medical facilities.

7.4.3 Application 3: Intelligent Power Transformer Monitoring: Advanced Applications of Echo State Networks

High voltage transformers are critical infrastructure to ensure the energy security of a country. These transformers are prone to several kinds of attack, including environmental and malicious human actors. In remote locations, one major problem is transformer oil thefts that cause damage to the transformers and major losses to the electricity companies. The economic loss around this is also high as businesses have to pause or put a hold on production during these power outage scenarios. Timely detection of the conditions of transformers can help the electricity companies take necessary actions to rectify the issues and restore power on time.

The real-time condition monitoring of the transformer requires sensory information on various transformer conditions including oil level, current levels, temperature, and humidity. These sensor data are fed to an ESN-based hardware attached to the transformer to perform classification and detection at the edge [48], as depicted in Figure 7.4.

Similar to the application in biomedical tumor monitoring discussed in the previous section, the memristive implementations of ESNs are energy-efficient, have small form factors, and in this case can be useful for the real-time monitoring and predictive maintenance of transformers.

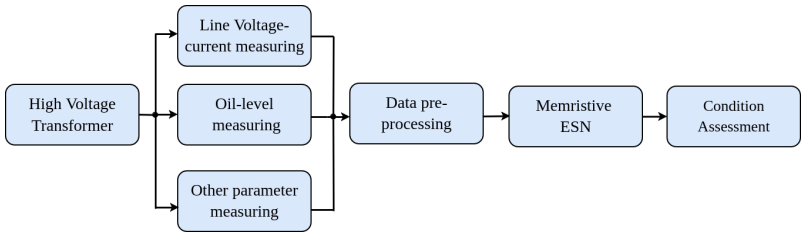

Figure 7.4: Transformer ESN integration architecture.

The architectural flexibility of ESNs (i.e., the control over sparsity and topology, the need for less intensive training, and high generality) offers reduced computational overhead, thereby making them suitable for stand-alone applications. The ability to control the computational load implies better control over energy efficiency and long battery life, which is critical for edge applications. Since the memristive ESN can handle analog sensory data directly to the network, there is no need to have additional data converters, and the prediction could be performed in a continuous manner. This makes the detection real-time within the hardware and avoids the need to have complex pre-processing or filtering modules.

At an application level, the solution can prove useful for electricity companies for ensuring the reliability of power distribution networks. The patterns of changes in the sensory data could be further analyzed to uncover anomalies and predict the occurrence of critical failures ahead of time. This example shows how memristive crossbars, even with variability and signal integrity challenges, could be used for a practical use-case and as a general energy-efficient solution to implement in hardware.

8

Crossbars for Real-Time EM Applications

The use of memristive crossbars as a classifier for wearables to be used with sensors is an emerging topic of interest. This interest is triggered by the fact that small crossbars in neural networks will be sufficient to perform most classification tasks if the number of sensors is less, and if the sensory data is discriminatory.

We explore the use of RF sensors and antennas in many different use-cases along with crossbar-based classifiers in wearables or edge devices, where lower power and smaller form factors are critical. Given that the memristors with super-resolution or modular arrays can largely deal with accuracy issues and reliability of analog computing, the crossbar-based solutions become promising for these applications.

8.1 Position Monitoring of Human Hand Movements

Figure 8.1 shows the cylinder body model for tracking movements, emulating a human motion. Cross-slot antennas are placed along the hand, and when the person moves, there are radiation pattern changes that occur. The hand motion tracking has several use cases such as to monitor walking, running, climbing, sleeping, and dancing. With the movements, the coordinate or position of the antenna changes, which

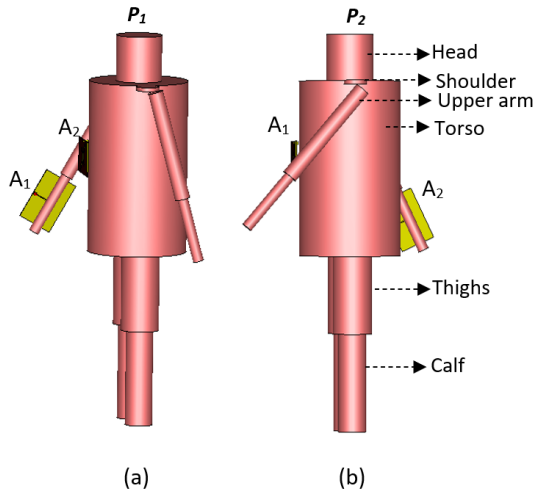

Figure 8.1: 12 cylinder body model with antenna A_2 fixed on the front side of the body: (a) Antenna A_1 fixed on right hand and position P_1 and, (b) Antenna A_1 fixed on right hand and position P_2.

can be captured from the transmission characteristics of the antenna as outlined in works by George *et al.* [22] and Pavithran *et al.* [54].

In the past, most human monitoring of human activity relied on cameras. However, cameras are often expensive and unreliable as they can be impacted by line-of-sight issues. They are also not practical for use in real-world settings as injuries may occur.

Alternatively, the use of antenna-based position classification is a safer, more accurate and cheaper alternative. Since the antennas along with the crossbar integrate as a wearable, the detections and classification can happen within the device without the need to have an additional computational unit. The neural networks such as GAN can be used for generating transmission characteristics from antennas (particularly hand swing activity) for training the networks. The analysis showed that ANN models showed superior accuracy over other ML algorithms such as Random Forest, DT and SVM, which makes them promising for use in crossbar architectures.

8.2 Parasitic Effects Prediction in On-Chip-Antennas

With the advent of wireless communication systems and the requirement for compact, efficient, and wearable devices, a huge demand exists for On-Chip-Antennas (OCA). OCAs are highly essential in various applications including security or safety, sports, medical, and space applications where miniaturization of devices is important. In contrast with the conventional antennas that exist off-chip, the OCAs are directly integrated into the silicon substrate, reducing the overall size of the entire system. However, the performance of OCAs is often compromised by the parasitic effects that arise with the interaction of the antenna with the packaging.

Packaging is essential as it protects against electromagnetic interference (EMI) and the external environment, ensuring the integrity of electrical connections and providing heat management. The packaging materials and structure play a significant role in introducing parasitic effects which vary the antenna performance factors like operating frequency, bandwidth, radiation efficiency, and gain. Understanding the effects of parasitics that arise with the packaging of OCAs helps in performance optimization by providing proper compensation.

The parasitic effects of OCAs increase as the operating frequency approaches the sub-terahertz range. In this context, machine learning (ML) techniques, and in particular neural networks, can be utilized as a promising solution for minimizing the parasitic effects due to packaging in OCAs.

The proposed work demonstrates the potential of artificial neural networks (ANNs) in the performance optimization of OCAs, providing an efficient solution for predicting the parasitic effects that arise with packaging. This is done by training the ANN with datasets of square spiral antenna operating in a sub-terahertz frequency range wrapped in a QFN package using Ansys HFSS. This allows the model to learn the correlation between the antenna design parameters and the key performance metrics [52].

8.3 Prediction of 3D Printed Substrate's Dielectric Constant Using Artificial Neural Network

The characterization of materials is of great significance in the field of avionics, aerospace, consumer wearables and biomedical applications. Techniques like time-domain spectroscopy, probes, and network analyzers are used for analyzing material characteristics. The Keysight N1501A high-temperature dielectric probe kit and SPEAG's dielectric assessment kits are some examples of dielectric measurement equipment available in the market.

In this work, the transmission line is designed on FR4, PLA, TPU, and cardboard material. The S-parameters are analyzed using a vector network analyzer and are applied to Kramer's Kronig algorithm to obtain the dielectric constant. The S-parameter data and the dielectric constant value obtained from Kramer's Kronig algorithm for four substrates, namely FR4, TPU, PLA, and cardboard material, are trained on an ANN to predict the dielectric constant of an unknown material by effectively modeling the relationship between S-parameters and the material's electromagnetic properties. The predictions from the ANN model are crucial for designing and optimizing the 3D-printed patch antennas since the antenna's performance is highly dependent on the material parameters. Furthermore, this prediction capability reduces the trial and error needed when designing antennas based on new or non-traditional 3D-printed substrates. The ANN, once trained, can offer a fast and reliable means for the estimation of dielectric characteristics, which can further automate the selection of substrate material. In the validation stage, a patch antenna is designed using the design equations to operate at 2.4 GHz with the dielectric constant predicted by ANN for the PLA substrate. This is modelled in Ansys HFSS and then fabricated on a PLA substrate which is 3D printed [53]. The block diagram shown in Figure 8.2 depicts the workflow for designing antenna from a predicted dielectric constant with ANN.

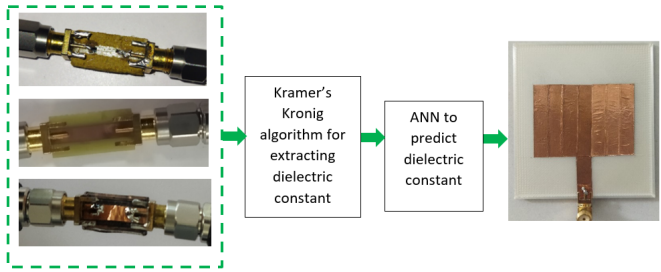

Figure 8.2: Block diagram showing the workflow of designing antenna from predicted dielectric constant from artificial neural networks.

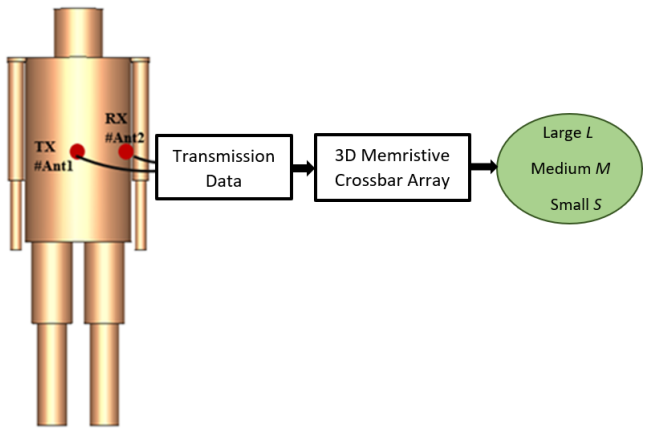

Figure 8.3: Block diagram showing cloth size prediction using transmission characteristics from $\#Ant1$ and $\#Ant2$ and 3D memristive crossbar array.

8.4 Cloth Size Prediction using Memristive Crossbar Array

The prediction of cloth size using the transmission values from antennas and memristive crossbar array is discussed in George *et al.* [21]. Figure 8.3 shows the block diagram detailing the prediction of cloth size from the transmission characteristics of the antenna and its classification

using 3D memristive crossbar array. George *et al.* [21] analyzes the creeping waves around the human torso by conducting experiments on a human volunteer. The transmitter $\#Ant1$ is fixed on the front side of the volunteer and $\#Ant2$ is moved around the body to study the creeping characteristics of EM waves. It is noted that EM waves decay as they propagate away from the transmitter antenna. A peak is observed when transmitter and receiver antennas are directly opposite to each other due to the constructive interference of the EM waves. The transmission data is given as input to the 3D memristive crossbar array for the classification of size as Large L, Medium M and Small S.

8.5 Imaging Techniques for Antenna Radiation Pattern

The works reported so far discuss the prediction of the radiation patterns of the antenna from the designed structure of the antenna. In this work, we take images of 3D radiation patterns of different types of antennas available in the literature and extract features from these images. We use pixel sampling, where different window shapes are employed to extract information from the 3D radiation pattern of the antenna. With these extracted features, we predict the type of antenna corresponding to the radiation pattern.

8.5.1 General Purpose Image Processing

Memristive crossbars perform weighted summation operations similar to convolution operations when images are used as inputs. Here, each column of a crossbar can be used for representing the weights of a filter. The image pixels can be considered as voltages applied to the rows of the crossbar, and the output current represents the filtered value.

Let us start with a image vector segment represented by a 3×3 matrix X:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

This can be vectorized as column vector $\mathbf{v}$:

$$\mathbf{v} = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix}$$

Here, $\mathbf{v}$ is the pixel values of the image segment that are applied as voltages across the crossbar rows.

Now, consider a filter kernel K:

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$$

That can be vectorized as $\mathbf{w}$:

$$\mathbf{w} = \begin{bmatrix} k_{11} \\ k_{12} \\ k_{13} \\ k_{21} \\ k_{22} \\ k_{23} \\ k_{31} \\ k_{32} \\ k_{33} \end{bmatrix}$$

This represents the conductance values $\mathbf{G}$ of a memristor column in the crossbar. Then we calculate the output current I_{out} as:

$$I_{\text{out}} = \sum_{i=1}^{9} G_i V_i$$

where G_i is the i-th memristor in the column, representing the filter weights and V_i is the applied voltage in the i-th row for the pixel value v_i.

In the vector form, this is:

$$I_{\text{out}} = \mathbf{G} \cdot \mathbf{v} = \sum_{i=1}^{9} G_i v_i$$

This is nothing but a generalized convolution operation between the filter kernel and the image patch. So every crossbar can be considered as a pack of filters, that can all be processed in parallel. Hence, the crossbar can be considered as a parallel processor for image filtering operations, thereby providing an energy-efficient way to perform analog image processing on chip.

8.5.2 Advanced Pixel Sampling Techniques Using Memristive Crossbars

The use of crossbars has been largely for image classification [38], showcasing the efficiency of memristor crossbars in performing convolutional operations. It is also shown to be useful for edge detection in images [79]. Usually, the filter window shape is limited to square or rectangular. The filter size is also fixed during the operations. In fact, due to this rigid structure, preserving the edge structure often becomes difficult.

Exploring different shaped windows in convolution operations with various types of kernels, such as rectangular, circular, elliptical, and cross-shaped windows, extends the idea of extracting and identifying features that are usually not observed by square or rectangular windows. The general convolution operation on a 2D image array I and a filter kernel K for a convolution at pixel location (i, j) can be written as:

$$I_{\text{filtered}}(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} K(m,n) \cdot I(i+m, j+n), \tag{8.1}$$

where: $I(i+m, j+n)$ is the neighboring pixel values centered at (i, j), $K(m, n)$ are the kernel weights, and a and b represent half the height and width of the kernel, respectively.

A rectangular filter share is the most common with $h \times w$ sized rectangular filter K:

$$K = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1w} \\ k_{21} & k_{22} & \cdots & k_{2w} \\ \vdots & \vdots & \ddots & \vdots \\ k_{h1} & k_{h2} & \cdots & k_{hw} \end{bmatrix}$$

Then the image convolution at (i, j):

$$I_{\text{filtered}}(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} K(m+a, n+b) \cdot I(i+m, j+n), \tag{8.2}$$

where $a = \lfloor h/2 \rfloor$ and $b = \lfloor w/2 \rfloor$.

Now, changing the shape to a circular filter, of radius R, the convolution of neighboring pixels (m, n) that fall within the circle centered at (i, j) is considered:

$$(m-i)^2 + (n-j)^2 \leq R^2. \tag{8.3}$$

Therefore, the convolution becomes:

$$I_{\text{filtered}}(i,j) = \sum_{m=-R}^{R} \sum_{n=-R}^{R} K(m,n) \cdot I(i+m, j+n) \quad \text{if} \quad (m^2+n^2) \leq R^2. \tag{8.4}$$

This leads to isotropic smoothing and, in comparison to a square or rectangular window, will have reduced boundary artifacts caused by squared edges.

An elliptical filter is another filter shape that has a major axis a and a minor axis b, and is centered at (i, j):

$$\frac{(m-i)^2}{a^2} + \frac{(n-j)^2}{b^2} \leq 1. \tag{8.5}$$

The elliptical filter kernel is then:

$$I_{\text{filtered}}(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} K(m,n) \cdot I(i+m, j+n) \quad \text{if} \quad \frac{m^2}{a^2} + \frac{n^2}{b^2} \leq 1. \tag{8.6}$$

This shape is also anisotropic filtering, however, it is more dominant along a particular axis.

Yet another possibility is a cross-shaped filter that can give focus to junctions or edges. Lets consider a cross kernel K:

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

In this case, the convolution operation only needs to consider the non-zero weights as:

$$I_{\text{filtered}}(i,j) = \sum_{\substack{m,n \\ K(m,n)\neq 0}} K(m,n) \cdot I(i+m, j+n). \tag{8.7}$$

Hence, this is more useful for detecting intersections or enhancing both vertical and horizontal edges.

Some of the traditional ways of implementing filters in memristors include a framework for noise reduction in images using memristor-based median filtering, highlighting the potential of memristor crossbars in implementing traditional filtering algorithms [31]. Wang *et al.* [68] demonstrated the effectiveness of memristor crossbar arrays in implementing Laplacian and Gaussian filters for image enhancement tasks.

It can be observed that changing the shape of the filter has a strong influence on the way image features are obtained. It can reveal new information that is usually not available with traditional rectangular shaped filters. Memristive crossbars can accommodate flexible filter shapes by switching ON or OFF the desired or undesired nodes. In fact, a more generalised approach would be to have a random dropout in the weight/filter values, which can give irregular filter structures as well.

8.5.3 Predicting Antenna Type from 3D Radiation Patterns

The most common types of radiation patterns of antennas include: a) the broadside radiation pattern generated by patch antennas, (b) the omnidirectional radiation pattern given by dipole antennas, and (c) the omnidirectional radiation pattern without vertical symmetry produced by monopole antennas. Patch antennas give a broadside radiation due

to the presence of a metallic ground plane on the bottom side of the substrate. The maximum gain will be directed along the broadside direction. In the case of the dipole antenna, it has two nulls along the vertical direction of the antennas and the monopole antenna depicts two nulls along its direction like dipole but the radiation pattern is not vertically symmetrical.

Figure 8.4 shows the requirement of predicting antenna types. The prediction of antenna patterns from encapsulated devices helps engineers to analyze the design even if the device cannot be opened, as depicted in Figure 8.4a. The design of the antenna based on the area it has to cover can be easily implemented if the neural network is trained with various types of radiation patterns as represented in Figure 8.4b. The EM (electromagnetic) scanner can also be used as a tool for school students to learn about antenna types. Figure 8.4 shows the device envisioned to predict antenna types from the radiation pattern, and the pixel circuit with three transistors is depicted in Figure 8.4e.

In this section, the simulation results of predicting antenna type based on 3D radiation pattern using a) CNN, b) YOLO, c) Decision Tree, d) VG-19, e) Random Forest, f) KNN, and g) Naive Bayes are discussed. These algorithms are trained and tested using Dipole, Monopole and Patch antenna patterns and their precision, and recall that F1 scores for both with and without noise are recorded. In addition, the accuracy of each system is tested. Table 8.1 [24] represents the testing accuracy of antenna pattern recognition systems without noise impact. Table 8.2 [24] represents the testing accuracy results of Gaussian noise.

Table 8.1: Testing accuracy results using various algorithms in antenna patterns.

SlNo.	Algorithms used	Without adding noise in Antenna patterns			
		Precision	Recall	F1 Score	Accuracy
1.	CNN	0.98	1	0.98	0.99
2.	YOLO-V8	1	1	1	1
3.	Decision Tree	0.98	0.95	0.95	0.98
4.	VGG-19	1	1	1	1
5.	Random Forest	1	1	1	1
6.	KNN	1	1	1	1
7.	Naive Bayes	0.77	0.93	0.63	0.70

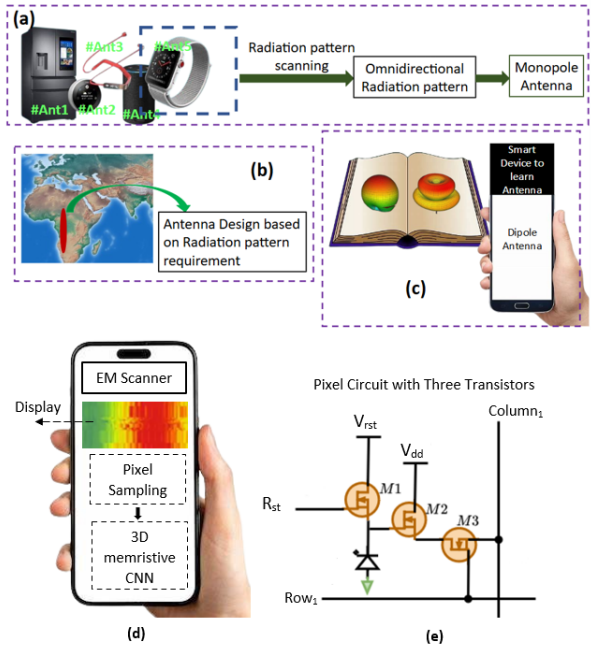

Figure 8.4: Image depicting applications of predicting antenna type: (a) Prediction of radiation pattern from encapsulated device, (b) Design of antenna based on required radiation pattern, (c) Learning device for school students to learn antenna types. EM Scanner: (d) Device to visualize radiation pattern and predict antenna type and (e) Pixel circuit with three transistors.

Table 8.2: Accuracy results of Gaussian noise testing on various algorithms.

SlNo.	Algorithms used	Impact of Gaussian Noise			
		Precision	Recall	F1 Score	Accuracy
1.	CNN	1.00	1.00	0.99	1.00
2.	YOLO-V8	1	1	1	1
3.	Decision Tree	0.99	0.98	0.97	0.97
4.	VGG-19	1	1	1	1
5.	Random Forest	1	1	1	1
6.	KNN	1	1	1	1
7.	Naive Bayes	0.87	0.93	0.77	0.88

9

Summary

The crossbar configuration using memristors can simplify the MAC operation implementation. This enables analog-based energy-efficient neural networks on chip. The memristor switching exhibits properties similar to biological synapses which enables neuromorphic applications. However, these implementations face practical challenges such as variability and aging effects, making integration with CMOS challenging.

The fabrication complexities and scalability constraints can pose significant challenges to make the on-chip implementations feasible. Minimizing fabrication defects, 3D configurations of crossbar, and scaling are critical issues. The adoption of open-source PDKs and collaborative frameworks can help to standardize the process, democratize the designs, improve device consistency, and ultimately lower costs.

The compatibility and integration challenges with CMOS technologies and emerging beyond-CMOS are critical. The integration with emerging technologies such as quantum computing could unlock novel computing paradigms capable of overcoming current system-level limitations. Adaptive learning algorithms and training can overcome memristor variability-induced inaccuracies. The use of pruning, dropout, and reinforcement learning could improve the reliability and robustness.

In summary, while there are several challenges, mitigating device variability, fabrication and scalability challenges, and refining integration and training methods can lead to reliable implementations. Once these challenges are overcome, the memristor crossbars can build efficient and adaptable AI hardware that can drive advancements in real-time and neuromorphic applications.

Acknowledgements

The author is grateful to the graduate students Aswani AR, Vineeta Nair, Sruthi P, Aleena Kabeer, Anitha Gopi, Shilpa P, and the postdoctoral fellow Elizabeth George for their reviews and feedback.

References

[1] S. Afshari, M. Musisi-Nkambwe, and I. S. Esqueda, "Analyzing the impact of memristor variability on crossbar implementation of regression algorithms with smart weight update pulsing techniques," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, 2022, pp. 2025–2034.

[2] F. Aguirre, A. Sebastian, M. Le Gallo, W. Song, T. Wang, J. J. Yang, W. Lu, M.-F. Chang, D. Ielmini, Y. Yang, *et al.*, "Hardware implementation of memristor-based artificial neural networks," *Nature communications*, vol. 15, no. 1, 2024, p. 1974.

[3] A. Ascoli, F. Corinto, V. Senger, and R. Tetzlaff, "Memristor model comparison," *IEEE Circuits and Systems Magazine*, vol. 13, no. 2, 2013, pp. 89–105.

[4] J. Ballmaier, S. Walfort, and M. Salinga, "Resistance drift of phase change materials beyond the power law," *Advanced Electronic Materials*, 2025, p. 2 400 905.

[5] D. Biolek and Z. Biolek, "Predictive models of nanodevices," *IEEE Transactions on Nanotechnology*, vol. 17, no. 5, 2018, pp. 906–913.

[6] D. Biolek, Z. Kolka, V. Biolková, Z. Biolek, M. Potrebić, and D. Tošić, "Modeling and simulation of large memristive networks," *International Journal of Circuit Theory and Applications*, vol. 46, no. 1, 2018, pp. 50–65.

[7] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature communications*, vol. 9, no. 1, 2018, pp. 1–12.

[8] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, "Sneak-path constraints in memristor crossbar arrays," in *2013 IEEE International Symposium on Information Theory*, IEEE, pp. 156–160, 2013.

[9] I. Chakraborty, M. Ali, A. Ankit, S. Jain, S. Roy, S. Sridharan, A. Agrawal, A. Raghunathan, and K. Roy, "Resistive crossbars as approximate hardware building blocks for machine learning: Opportunities and challenges," *Proceedings of the IEEE*, vol. 108, no. 12, 2020, pp. 2276–2310.

[10] Y.-F. Chang, *Memristors-The Fourth Fundamental Circuit Element-Theory, Device, and Applications*. IntechOpen, 2024.

[11] J. Chen, J. Li, Y. Li, and X. Miao, "Multiply accumulate operations in memristor crossbar arrays for analog computing," *Journal of Semiconductors*, vol. 42, no. 1, 2021.

[12] F. Chollet *et al.* (2015). "Keras," URL: https://github.com/fchollet/keras.

[13] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, 1971, pp. 507–519.

[14] L. Chua, "Memristor, hodgkin–huxley, and edge of chaos," *Nanotechnology*, vol. 24, no. 38, 2013.

[15] L. O. Chua, "Memristors on 'edge of chaos'," *Nature Reviews Electrical Engineering*, vol. 1, no. 9, 2024, pp. 614–627.

[16] F. Corinto, "Memristor computing systems: At the crossroad between circuit theory and artificial intelligence," in *Intelligence in Chip: Integrated Sensors and Memristive Computing*, River Publishers, 2024, pp. 176–194.

[17] F. Corinto, P. P. Civalleri, and L. O. Chua, "A theoretical approach to memristor devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, 2015, pp. 123–132.

[18] S. Duan, X. Hu, Z. Dong, L. Wang, and P. Mazumder, "Memristor-based cellular nonlinear/neural network: Design, analysis, and applications," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 6, 2014, pp. 1202–1213.

[19] J. K. Eshraghian, X. Wang, and W. D. Lu, "Memristor-based binarized spiking neural networks: Challenges and applications," *IEEE Nanotechnology Magazine*, vol. 16, no. 2, 2022, pp. 14–23.

[20] Y. Fang, Z. Yu, Z. Wang, T. Zhang, Y. Yang, Y. Cai, and R. Huang, "Improvement of hfox-based rram device variation by inserting ald tin buffer layer," *IEEE Electron Device Letters*, vol. 39, no. 6, 2018, pp. 819–822. DOI: 10.1109/LED.2018.2831698.

[21] E. George, S. Pallathuvalappil, and A. James, "Smart clothing using antenna and memristive ann," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2024. DOI: 10.1109/ISCAS58744.2024.10558107.

[22] E. George, A. Radhakrishnan, and A. James, "Antenna based classification of posture positions using 1t1m crossbar arrays," in *2023 18th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, pp. 1–4, 2023. DOI: 10.1109/CNNA60945.2023.10652784.

[23] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, "Ai and memory wall," *IEEE Micro*, 2024.

[24] A. Gopi, E. George, R. Rahul, and A. James, "Ai for antenna design re-engineering: Yes, radiation patterns predict antenna structures!" In *2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS)*, IEEE, pp. 66–70, 2024.

[25] W. Ham, Y.-W. Song, J. H. Yoon, S. Lee, J.-M. Park, J. Lee, and J.-Y. Kwon, "Surface roughness engineering for improvement of cycle-to-cycle variability of rram," *Applied Surface Science*, vol. 670, 2024.

[26] T. Hennen, L. Brackmann, T. Ziegler, S. Siegel, S. Menzel, R. Waser, D. J. Wouters, and D. Bedau, "Synaptogen: A cross-domain generative device model for large-scale neuromorphic circuit design," *IEEE Transactions on Electron Devices*, 2024.

[27] M. Itoh and L. Chua, "Memristor cellular automata and memristor discrete-time cellular neural networks," *Handbook of Memristor Networks*, 2019, pp. 1289–1361.

[28] A. P. James and L. O. Chua, "Variability-aware memristive crossbars—a tutorial," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 6, 2022, pp. 2570–2574. DOI: 10.1109/TCSII.2022.3169416.

[29] A. James, "Energy efficiency and design challenges in analogue memristive chips," *Nature Reviews Electrical Engineering*, vol. 1, no. 1, 2024, pp. 6–7.

[30] A. P. James and L. O. Chua, "Analog neural computing with super-resolution memristor crossbars," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 11, 2021, pp. 4470–4481. DOI: 10.1109/TCSI.2021.3079980.

[31] X. Ji, Z. Dong, G. Zhou, C. S. Lai, Y. Yan, and D. Qi, "Memristive system based image processing technology: A review and perspective," *Electronics*, vol. 10, no. 24, 2021.

[32] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *European Journal of physics*, vol. 30, no. 4, 2009.

[33] T. Kamsma, R. van Roij, and C. Spitoni, "A simple mathematical theory for simple volatile memristors and their spiking circuits," *Chaos, Solitons & Fractals*, vol. 186, 2024, pp. 115–320.

[34] O. Krestinskaya, A. Irmanova, and A. P. James, "Memristive non-idealities: Is there any practical implications for designing neural network chips?" In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2019. DOI: 10.1109/ISCAS.2019.8702245.

[35] R. Kumar, A. Chordia, A. Aswani, A. James, and J. N. Tripathi, "Uncertainty quantification of memristor crossbar array for vector matrix multiplication," in *2021 IEEE 25th Workshop on Signal and Power Integrity (SPI)*, pp. 1–4, 2021. DOI: 10.1109/SPI52361.2021.9505193.

[36] 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, 2012, pp. 211–221.

[37] J. Laydevant, L. G. Wright, T. Wang, and P. L. McMahon, "The hardware is the software," *Neuron*, vol. 112, no. 2, 2024, pp. 180–183.

[38] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature electronics*, vol. 1, no. 1, 2018, pp. 52–59.

[39] Y. Li and K.-W. Ang, "Hardware implementation of neuromorphic computing using large-scale memristor crossbar arrays," *Advanced Intelligent Systems*, vol. 3, no. 1, 2021.

[40] X. Liu and Z. Zeng, "Memristor crossbar architectures for implementing deep neural networks," *Complex & Intelligent Systems*, vol. 8, no. 2, 2022, pp. 787–802.

[41] X. Liu, Z. Zeng, and S. Wen, "Implementation of memristive neural network with full-function pavlov associative memory," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 9, 2016, pp. 1454–1463.

[42] A. Mazady, M. T. Rahman, D. Forte, and M. Anwar, "Memristor puf—a security primitive: Theory and experiment," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, 2015, pp. 222–229.

[43] I. Messaris, A. Ascoli, A. S. Demirkol, V. Ntinas, D. Prousalis, and R. Tetzlaff, "High frequency response of volatile memristors," *Advanced Electronic Materials*, vol. 10, no. 12, 2024, p. 2 400 172.

[44] D. Mikhailenko, C. Liyanagedera, A. P. James, and K. Roy, "M2ca: Modular memristive crossbar arrays," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018. DOI: 10.1109/ISCAS.2018.8351112.

[45] D. J. Mountain, M. R. McLean, and C. D. Krieger, "Memristor crossbar tiles in a flexible, general purpose neural processor," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, 2017, pp. 137–145.

[46] V. V. Nair, E. George, and A. James, "Real-time tumor detection using electromagnetic signals with memristive echo state networks," *IEEE Internet of Things Journal*, vol. 11, no. 20, 2024, pp. 33 712–33 721. DOI: 10.1109/JIOT.2024.3432763.

[47] V. V. Nair, C. Reghuvaran, D. John, B. Choubey, and A. James, "Essm: Extended synaptic sampling machine with stochastic echo state neuro-memristive circuits," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 4, 2023, pp. 965–974. DOI: 10.1109/JETCAS.2023.3328875.

[48] V. V. Nair, A. P, and A. James, "High voltage transformer condition monitoring using memristive echo state networks," in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2024. DOI: 10.1109/ISCAS58744.2024.10558533.

[49] M. Nelson and J. Rinzel, "The hodgkin-huxley model," *The book of genesis*, vol. 2, 1995.

[50] S. Pallathuvalappil, R. Kottappuzhackal, and A. James, "Explainable model prediction of memristor," *IEEE Open Journal of the Industrial Electronics Society*, 2024.

[51] G. Papandroulidakis, A. Serb, A. Khiat, G. V. Merrett, and T. Prodromakis, "Practical implementation of memristor-based threshold logic gates," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 8, 2019, pp. 3041–3051.

[52] S. Pavithran, E. George, and A. P. James, "Parasitic effects prediction in on-chip-antennas," Banglore, India, 2024.

[53] S. Pavithran, A. Pramod, E. George, and A. P. James, "Prediction of 3d printed substrate's dielectric constant using artificial neural network," in *Proceedings of the IEEE MAPCON 2024*, Hyderabad, India: IEEE, 2024.

[54] S. Pavithran, A. S, V. V. Nair, E. George, and A. P. James, "Position monitoring of human hands using cross-slot antennas and ai integration for hand swing activity analysis," 2024.

[55] R. Pelke, F. Staudigl, N. Thomas, M. Hossein, N. Bosbach, J. Cubero-Cascante, R. Leupers, and J. M. Joseph, "The show must go on: A reliability assessment platform for resistive random access memory crossbars," *Philosophical Transactions A*, vol. 383, no. 2288, 2025.

[56] PyMem Team, "PyMem python memristor," 2023. URL: https://github.com/ajiiit/PyMem.git (accessed on 06/15/2023).

[57] PyMem Team, *Website User Manual, PyMem*, 2023. URL: https://docs.google.com/document/d/1kg93byTul19aAtU9jqySZc8rUovmX-GWpzAHIyPBp1c/edit?usp=sharing.

[58] A. Radhakrishnan, S. Pallathuvalappil, B. Choubey, and A. James, "Bridge memristor super-resolution crossbars," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 4, 2022, pp. 944–951.

[59] A. Radhakrishnan, J. Palliyalil, S. Babu, A. Dorzhigulov, and A. James, "Pymem: A graphical user interface tool for neuromemristive hardware–software co-design," *IEEE Open Journal of the Industrial Electronics Society*, vol. 5, 2024, pp. 81–90. DOI: 10.1109/OJIES.2024.3363093.

[60] E. Salvador, M. Gonzalez, F. Campabadal, R. Rodriguez, and E. Miranda, "Modeling and simulation of correlated cycle-to-cycle variability in the current-voltage hysteresis loops of rram devices," *IEEE Transactions on Nanotechnology*, 2024.

[61] L. Shi, G. Zheng, B. Tian, B. Dkhil, and C. Duan, "Research progress on solutions to the sneak path issue in memristor crossbar arrays," *Nanoscale Advances*, vol. 2, no. 5, 2020, pp. 1811–1827.

[62] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky, "Memristor-based multilayer neural networks with online gradient descent training," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, 2015, pp. 2408–2421.

[63] J. A. Starzyk *et al.*, "Memristor crossbar architecture for synchronous neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 8, 2014, pp. 2390–2401.

[64] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, 2008, pp. 80–83.

[65] T. Tanikawa, K. Ohnishi, M. Kanoh, T. Mukai, and T. Matsuoka, "Three-dimensional imaging of threading dislocations in gan crystals using two-photon excitation photoluminescence," *Applied Physics Express*, vol. 11, no. 3, 2018. DOI: 10.7567/APEX.11.031004.

[66] A. Thomas, "Memristor-based neural networks," *Journal of Physics D: Applied Physics*, vol. 46, no. 9, 2013, p. 093 001.

[67] I. Vourkas, D. Stathis, G. C. Sirakoulis, and S. Hamdioui, "Alternative architectures toward reliable memristive crossbar memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, 2016, pp. 206–217. DOI: 10.1109/TVLSI.2015.2388587.

[68] L. Wang, Q. Meng, H. Wang, J. Jiang, X. Wan, X. Liu, X. Lian, and Z. Cai, "Digital image processing realized by memristor-based technologies," *Discover Nano*, vol. 18, no. 1, 2023.

[69] Z. Wang, X. Wang, Z. Lu, W. Wu, and Z. Zeng, "The design of memristive circuit for affective multi-associative learning," *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 2, 2020, pp. 173–185.

[70] R. S. Williams and M. D. Pickett, "The art and science of constructing a memristor model," in *Memristors and Memristive Systems*, Springer, 2013, pp. 93–104.

[71] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature materials*, vol. 18, no. 4, 2019, pp. 309–323.

[72] L. Xie, H. A. Du Nguyen, M. Taouil, S. Hamdioui, and K. Bertels, "Interconnect networks for memristor crossbar," in *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'15)*, IEEE, pp. 124–129, 2015.

[73] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Extremely parallel memristor crossbar architecture for convolutional neural network implementation," in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 1696–1703, 2017.

[74] 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, pp. 1–7, 2013.

[75] X. Yang, B. Taylor, A. Wu, Y. Chen, and L. O. Chua, "Research progress on memristor: From synapses to computing systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, 2022, pp. 1845–1857.

[76] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, 2020, pp. 641–646.

[77] S. Zhang, G. L. Zhang, B. Li, H. H. Li, and U. Schlichtmann, "Aging-aware lifetime enhancement for memristor-based neuromorphic computing," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1751–1756, 2019. DOI: 10.23919/DATE.2019.8714954.

[78] Y. Zhang, X. Wang, and E. G. Friedman, "Memristor-based circuit design for multilayer neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, 2018, pp. 677–686.

[79] S. Zhu, L. Wang, Z. Dong, and S. Duan, "Convolution kernel operations on a two-dimensional spin memristor cross array," *Sensors*, vol. 20, no. 21, 2020.

[80] M. A. Zidan, Y. Jeong, J. Lee, B. Chen, S. Huang, M. J. Kushner, and W. D. Lu, "A general memristor-based partial differential equation solver," *Nature Electronics*, vol. 1, no. 7, 2018, pp. 411–420.