Von-Neumann and Beyond: Memristor Architectures

Dissertation by
Rawan Naous

In Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy

King Abdullah University of Science and Technology
Thuwal, Kingdom of Saudi Arabia

May, 2017
The dissertation of Rawan Naous is approved by the examination committee.

Committee Chairperson: Prof. Khaled N. Salama
Committee Members: Prof. Pierre Magistretti, Prof. Fadi Kurdahi, Prof. Mohamed-Slim Alouini, Prof. Basem Shihada
ABSTRACT

Von-Neumann and Beyond: Memristor Architectures
Rawan Naous

An extensive reliance on technology, an abundance of data, and increasing processing requirements have imposed severe challenges on computing and data processing. Moreover, the roadmap for scaling electronic components faces physical and reliability limits that hinder the utilization of the transistors in conventional systems and promotes the need for faster, energy-efficient, and compact nano-devices. This work thus capitalizes on emerging non-volatile memory technologies, particularly the memristor for steering novel design directives. Moreover, aside from the conventional deterministic operation, a temporal variability is encountered in the devices functioning. This inherent stochasticity is addressed as an enabler for endorsing the stochastic electronics field of study. We tackle this approach of design by proposing and verifying a statistical approach to modelling the stochastic memristors behaviour. This mode of operation allows for innovative computing designs within the approximate computing and beyond Von-Neumann domains.

In the context of approximate computing, sacrificing functional accuracy for the sake of energy savings is proposed based on inherently stochastic electronic components. We introduce mathematical formulation and probabilistic analysis for Boolean logic operators and correspondingly incorporate them into arithmetic blocks. Gate- and system-level accuracy of operation is presented to convey configurability and the different effects that the unreliability of the underlying memristive components has on the intermediary and overall output. An image compression application is presented to reflect the efficiency attained along with the impact on the output caused by the relative precision quantification.
In contrast, in neuromorphic structures the memristors variability is mapped onto abstract models of the noisy and unreliable brain components. In one approach, we propose using the stochastic memristor as an inherent source of variability in the neuron that allows it to produce spikes stochastically. Alternatively, the stochastic memristors are mapped onto bi-stable stochastic synapses. The intrinsic variation is modelled as added noise that aids in performing the underlying computational tasks. Both aspects are tested within a probabilistic neural network operation for a handwritten MNIST digit recognition application. Synaptic adaptation and neuronal selectivity are achieved with both approaches, which demonstrates the savings, interchangeability, robustness, and relaxed design space of brain-inspired unconventional computing systems.
ACKNOWLEDGEMENTS

A lot of people have helped me throughout this journey whether at the professional or the personal side. First and foremost I would like to express my sincerest gratitude to my advisor Professor Khaled Salama to whom I am indebted, for his support, guidance and encouragement. He taught me the principles and skills of research and communication, and pushed me to my limits to achieve my best potentials.

I would also like to thank Professor Gert Cauwenberghs and Professor Emre Neftci from UCSD and UCI for introducing me to the neuromorphic engineering field and for all the technical discussions and support. Thank you to Professor Ahmed Sultan for all the advice and for always having his door open for any ideas.

Many thanks to Professor Pierre Magistretti, Professor Fadi Kurdahi, Professor Mohamed-Slim Alouini, and Professor Basem Shihada for being part of my PhD committee and whose comments have helped enhance the quality of the work.

Thanks to all my friends and colleagues at the sensors lab, especially Maruan Al-Shedivat for working together on stochastic brain-like computing. Dr. Mohammed Affan Zidan for his help in the memristor field, Hamzah AlAhmadi and Mohamed Bahloul for our fruitful discussions, and all the members of the lab. Sincere gratitude to my friends whom I had the unique opportunity to meet at KAUST and with whom I shared unforgettable moments.

Finally, special thanks to my family whose support and encouragement were a great inspiration. My Mom, my sisters, my brother and my parents in law for always being there for me and helping me move forward. My husband, Khaled, for his unconditional love and support and for bearing with me every step of the way. To my lovely daughters Yasmine and Lynn who kept me going all this time. And to the souls of special people who have left but are never forgotten, Thank you.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examination Committee Page</td>
<td>2</td>
</tr>
<tr>
<td>Copyright</td>
<td>3</td>
</tr>
<tr>
<td>Abstract</td>
<td>4</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>6</td>
</tr>
<tr>
<td>List of Figures</td>
<td>10</td>
</tr>
<tr>
<td>List of Tables</td>
<td>12</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>13</td>
</tr>
<tr>
<td><strong>2 Stochastic Electronics</strong></td>
<td>19</td>
</tr>
<tr>
<td>2.1 Emerging Devices and Stochasticity</td>
<td>20</td>
</tr>
<tr>
<td>2.1.1 Fine Behavioral Modeling</td>
<td>20</td>
</tr>
<tr>
<td>2.1.2 Abstract Switching Model</td>
<td>21</td>
</tr>
<tr>
<td>2.1.3 Resistance Variation Model</td>
<td>22</td>
</tr>
<tr>
<td>2.2 Stochastic Memristor Modeling</td>
<td>22</td>
</tr>
<tr>
<td>2.2.1 Stochasticity Distribution</td>
<td>23</td>
</tr>
<tr>
<td>2.2.1.1 Poisson Distribution</td>
<td>23</td>
</tr>
<tr>
<td>2.2.1.2 Log-Normal Distribution</td>
<td>25</td>
</tr>
<tr>
<td>2.2.2 Behavioral Stochastic Modelling</td>
<td>28</td>
</tr>
<tr>
<td>2.2.3 Verilog-a Model</td>
<td>29</td>
</tr>
<tr>
<td>2.2.4 SPICE Modelling</td>
<td>30</td>
</tr>
<tr>
<td>2.2.4.1 Probability Transformation</td>
<td>31</td>
</tr>
<tr>
<td>2.2.4.2 Stochastic Modeling</td>
<td>32</td>
</tr>
<tr>
<td>2.3 Model Verification</td>
<td>33</td>
</tr>
<tr>
<td>2.3.1 Poisson Fitting</td>
<td>34</td>
</tr>
<tr>
<td>2.3.2 Log-Normal Fitting</td>
<td>34</td>
</tr>
<tr>
<td>2.4 Memristor Models</td>
<td>35</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 Moore’s Law and Beyond .............................................. 14
1.2 Memristors Hysteresis Response .................................... 15

2.1 Physical Switching ....................................................... 21
2.2 Linear Fitting for the Poisson Distribution ......................... 24
2.3 Exponential fitting for the Log-Normal Distribution ............... 26
2.4 Fitting of the Standard Deviation for the Log-Normal Distribution 27
2.5 SPICE stochastic modeling ............................................ 31
2.6 Memristor threshold variability model ................................ 32
2.7 Experimental Poisson Data ............................................ 33
2.8 Poisson Simulation Data ............................................... 33
2.9 Log-normal Experimental fitting ..................................... 35
2.10 Log-normal Simulations .............................................. 35
2.11 Deterministic and Stochastic Dynamics of the memristor switching 36
2.12 Stochastic output response and threshold variation .............. 37
2.13 Stochastic Hysteresis in Linear Model ............................. 38
2.14 Analytical model threshold variation ................................ 40
2.15 Stochastic hysteresis in the analytical model ...................... 41
2.16 Simmons Tunnel Barrier model and threshold variation .......... 42
2.17 Stochastic hysteresis for the Simmons Tunnel Barrier model .... 43

3.1 IMPLY Logic Operation with Memristors .......................... 47
3.2 CMOS-Like Memristor Logic ......................................... 48
3.3 Sequential Logic Operation Principles ............................. 49
3.4 Sequence of application for NAND gate ............................ 50
3.5 Output Probabilities for Stochastic NAND gate .................... 53
3.6 Accuracy of the NAND Operator .................................... 54
3.7 The output probabilities and accuracy of the main boolean operators 55
3.8 Schematic and truth table for the Half Adder ...................... 57
3.9 Output accuracy of the half adder block ............................ 58
3.10 Schematic and truth table for the full adder block ............... 59
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.11</td>
<td>Accuracy of the full adder output bits</td>
<td>60</td>
</tr>
<tr>
<td>3.12</td>
<td>Output bits comparison</td>
<td>61</td>
</tr>
<tr>
<td>3.13</td>
<td>Ripple Carry Adder structure</td>
<td>62</td>
</tr>
<tr>
<td>3.14</td>
<td>Adder Structures with different carry generation blocks</td>
<td>63</td>
</tr>
<tr>
<td>3.15</td>
<td>Trade-off between the accuracy, voltage and delay in approximate computing</td>
<td>65</td>
</tr>
<tr>
<td>3.16</td>
<td>Image Compression Application</td>
<td>67</td>
</tr>
<tr>
<td>3.17</td>
<td>Reconstructed images after DFT with approximate adders</td>
<td>68</td>
</tr>
<tr>
<td>3.18</td>
<td>PSNR values and corresponding Power Saving</td>
<td>69</td>
</tr>
<tr>
<td>4.1</td>
<td>Conceptual Scaling of Machine Complexity</td>
<td>72</td>
</tr>
<tr>
<td>4.2</td>
<td>Neuromorphic Engineering multiscale level of Investigation</td>
<td>73</td>
</tr>
<tr>
<td>4.3</td>
<td>Hodgkin-Huxely Neuron Model</td>
<td>76</td>
</tr>
<tr>
<td>4.4</td>
<td>Memristor Based Stochastic Neuron</td>
<td>78</td>
</tr>
<tr>
<td>4.5</td>
<td>Neuronal response with Poisson distribution</td>
<td>79</td>
</tr>
<tr>
<td>4.6</td>
<td>Synaptic Crossbar with stochastic memristors</td>
<td>81</td>
</tr>
<tr>
<td>4.7</td>
<td>Fully connected neural network with image recognition application</td>
<td>82</td>
</tr>
<tr>
<td>4.8</td>
<td>MNIST recognition with stochastic memristive neurons</td>
<td>83</td>
</tr>
<tr>
<td>4.9</td>
<td>Neural Network structure with the memristive crossbar</td>
<td>85</td>
</tr>
<tr>
<td>4.10</td>
<td>Applied signal pulses for the memristive network</td>
<td>86</td>
</tr>
<tr>
<td>4.11</td>
<td>Simulation Sample Output</td>
<td>87</td>
</tr>
<tr>
<td>4.12</td>
<td>Synaptic Adaptation to input images</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>Stochastic AND gate for multiplication</td>
<td>94</td>
</tr>
<tr>
<td>5.2</td>
<td>Low Density Parity Check Decoding with Stochastic Computing</td>
<td>95</td>
</tr>
<tr>
<td>5.3</td>
<td>Synaptic Sampling with stochastic transistor and memristor elements</td>
<td>96</td>
</tr>
<tr>
<td>5.4</td>
<td>Potentiation and Depression of Synapse</td>
<td>97</td>
</tr>
</tbody>
</table>
LIST OF TABLES

2.1 Fitting parameters for the characteristic wait time ........................................ 27
2.2 Memristor models with incorporated stochasticity ........................................... 44
3.1 NAND Gate Truth Table with Probabilistic Output ...................................... 51
3.2 Performance metrics for the different adder structures ................................ 65
3.3 Comparison of approximate processing techniques ....................................... 70
4.1 Comparison of stochastic neural network applications ................................. 90
Chapter 1

Introduction

In an era of diverse technological advancements and revolutionized computing systems—with applications such as the Internet of Things (IoT), big data, and virtual reality, and with all of the novel insight-computing concepts under investigation [1] a radical shift is required in the design of the underlying hardware. With low cost, compact size, and high performance being the primary drivers for industrial innovations, a multidisciplinary approach must be adopted to best achieve the untapped potentials of the applications at hand. Systems, architectures, devices and materials are under extensive research to meet the demands arising from the quest after next-generation, energy-efficient computing [2, 3]. In one context, Moores law has been providing the theoretical limits for the scaling of electronic components and their consequent chips over the years [4, 5]. Nonetheless, these scaling efforts have reached their physical barrier and are falling short in providing the required level of output and cost with the miniaturization of their respective dimensions, as is shown in Figure 1.1.

To this end, beyond Complementary Metal-Oxide Semiconductor (CMOS) devices from Fin Field Effect Transistors (FinFET) [6], carbon nanotubes [7], and emerging non-volatile memory technologies [8] are being investigated, both as an alternative and as a complementary technology. These novel components are mainly addressed to help alleviate the pressure being placed on current CMOS to meet stringent requirements in processing and memory [8]. One of the most prominent emerging solutions is the memristor. This resistive random-access memory (ReRAM) technology is a two-terminal device that is theoretically formulated in 1971 by Leon Chua [9] and
Figure 1.1: The scaling of the devices and its corresponding application in regards to Moore’s law and Beyond from the international technology roadmap for semiconductors [11].

experimentally realized by HP in 2008 [10]. Memristors change their resistance state according to the flux passing through them. The memristance ($M$) is defined as the derivative of the electric flux ($\phi$) with respect to the charge ($q$)

$$M = \frac{d\phi}{dq}$$ (1.1)

Contrary to the classical resistor, the memristor changes its resistance in response to the applied input. The resistance varies between two extremities of low resistance $R_{ON}$ and high resistance $R_{OFF}$. Non-linear behavior is encountered as a result of a sinusoidal input voltage, with the pinched hysteresis being the characteristic trademark of the current-voltage (I-V) relation. Figure 1.2 shows the difference in current-voltage response between the resistor and memristor elements.

Memristors are typically composed of two conducting metal electrodes that are separated by an insulating active layer that allows resistive switching under induced electrical excitation [12, 13]. Several models have been proposed to describe this
behavioral process. Propositions vary in complexity either to precisely meet fabricated devices and physical dynamics [10, 14, 15] or to offer a practical fit with a more abstract mathematical formulation of a generalized switching process [16, 17, 18, 19, 20]. In two broad categories, threshold-less device models exhibit a continuous change of resistance upon application of input stimuli. However, for threshold-based devices, the switching or shift from a low- to high-resistance level and vice versa is initiated once the input across the two terminals reaches a certain inherently set value [18]. This resistance change between resistance states, along with the non-volatility feature, allows the memristor to be employed in diverse fields whether in analog [21, 22], digital [23, 24, 25], memory [26, 27, 28], or unconventional applications [29, 30, 31, 32, 33].

In addition to the deterministic behaviour of the memristor, the variants of the composition of the underlying material and interactions impose a new domain of operation. A stochastic outcome is observed that stems from the switching characteristics and kinetics of the ionic and chemical reactions [34]. Thus, building on the concept of embracing the noise as a source of performance enhancement rather than as a drawback to be combated, the memristor could be adopted in the newly established
field of stochastic electronics \cite{35}. As a neuro-inspired design that benefits from variability and distortion, in a manner similar to how the human brain is constituted by unreliable elements within a noisy environment, stochastic electronics show a higher performance than conventional computers in processing and control tasks that tolerate levels of errors. This resilience and tolerance is inherently available when dealing with the real-world capturing, processing and transmission of data that lie at the core of the novel computing systems \cite{36, 37}.

This stochasticity feature intrinsic in the operation of the memristive elements has been observed in different devices with diverse models. Modification of the internal physical equations is required to fully capture the experimental behaviour of the device. This intrusive method of inducing variation is quite complex and requires an in depth understanding of the ionic and physical interactions, which are different for each device. Hence, a generic model is essential to allow for the easy incorporation of the stochastic memristor into the circuit models and applications. Thus, we formulated an abstract model of the variability based on the statistical properties observed along with the switching point characteristics. The model represents an agile formulation that allows for the preservation of the individual kinetics of the already-established models while adding a stochasticity feature into its operation. Chapter 2 addresses the origins of the variability in emerging technologies and its consequent impact on circuit operation. The details of our proposed model, its verification with experimental data for different probability distributions, and the corresponding current and voltage stochastic models are further elaborated. This study has led to a thorough understanding of stochastic operation principles and has paved the way for novel system-based applications.

In the context of more than Moore, a shift away from conventional approaches is crucial to adapt to the diversification principles, information processing and complex interaction with people and environments. From a processing perspective, diverse
techniques are starting to be employed, some of which include associative processing \[38\], brain-inspired designs \[39\], approximate \[40\] and stochastic \[41\] computation, and near-threshold computing \[42\]. Chapter 3 explores approximate and probabilistic computation based on stochastic memristors. We introduce a mathematical formulation of the logic operation based on the switching probability of the memristor, which acts as the main building block for Boolean operators. Probabilistic arithmetic blocks and levels of savings are portrayed along with an image-processing application that demonstrates the level of compromise possible among the different design metrics and relative precision quantification.

Chapter 4 adopts an alternative approach to computing with brain-inspired structures. Neuromorphic architectures that try to attain the functionalities of the brain by building on processing units composed of neurons and synapses are investigated based on memristive elements. The stochasticity of the memristor is utilized as a source of inherent noise injection in a novel stochastic neuronal design. Moreover, the digital-like behaviour with switching between two distinct states in a non-deterministic manner is also mapped into stochastic synapses with high- and low-value connection weights. We have established a circuit-level simulator that allows for the emulation of a stochastic neural network with a MNIST pattern recognition application. Levels of learning with synaptic and neuronal adaptation are apparent within a co-localized memory and processing platform.

The contributions of the PhD work are as follows:

1. It introduces a statistical model for stochastic memristive elements and validates its accuracy with experimentally reported data \[34\] \[43\].

2. It introduces a theoretical formulation of the probabilistic behavior of logic operators based on stochastic memristors and introduces the approximate computing approach with non-deterministic memristors. It further incorporates
probabilistic-logic behaviour into arithmetic and image-processing applications along with precision and quality assessment. [44] [45] [46] [47].

3. It proposes a memristor-based neuron circuit and analyses its behavior against probabilistic operation within a neural network recognition application. It also builds a neural-network simulation platform with comprehensive features for input data and layer size with flexibility for the modeling of stochastic neuronal and synaptic structures [33] [48] [49] [50].

4. It introduces mathematical analysis of the sneak-path effect in the passive memristor crossbar memories. It proposes a channel-estimation technique based on pilots to alleviate the effect of distortion and provide an accurate memory read-out [51] [52] [53] [54] [55].
Chapter 2

Stochastic Electronics

Variability, in its static and dynamic forms, has been a widely apparent feature in the operation of electronic components. This mode of operation is a result of the extensive dimension- and power-scaling endeavours in diverse technologies \[37\]. In general, stochasticity is regarded as a drawback and efforts are made to minimize the variability as much as possible. However, in particular contexts, the added variations are actually beneficial to computations and overall performance; thus, a novel paradigm arises that is based on embracing stochasticity rather than combating it. Stochastic electronics is a realm of study that builds on components that are inherently non-deterministic and integrates them in circuit environments that actively utilize noise in an advantageous manner \[35\]. This chapter investigates the origins of stochasticity in emerging non-volatile memory technologies. The contributions of this study are as follows

1. Introduce a statistical model of the stochastic memristor at the circuit and system levels.

2. Verify the validity of the proposed model against experimental data for different mathematical distributions.

3. Elaborate upon the incorporation of the temporal stochasticity into general current and voltage models.
2.1 Emerging Devices and Stochasticity

Emerging non-volatile memory technologies are based primarily on a resistive switching behaviour with an abrupt transition from an insulating to a conducting state [56] that results in high and low resistance values, respectively. Devices range from Phase-Change Memories (PCM) [12] in which the change from crystalline to amorphous states is thermally processed; magnetically controlled resistance change in Spin Transfer Torque Random-Access Memories (STT-RAM) [57]; and electrically activated oxide-based switching in Resistive Random-Access Memories (ReRAM). The memristor resides in the oxide-controlled process category and is the most promising amongst emerging technologies, as it offers high density and long retention times at minimum feature size [11, 2, 8]. Despite the differences in the structures and the materials used in all of the aforementioned devices, a consensus agrees on the importance of material properties and the overall interaction progression [56] on the behavioural characteristics. Hence, aside from the appealing features of the emerging devices, the intriguing stochasticity characteristic [34] imposes a new domain of operation. PCMs [58, 59], electrochemical metallization memory [60], conductive bridge RAM (CBRAM) [61], and oxide-based memristive material [62, 63, 64, 59, 65, 66] exhibit variation and stochasticity in their corresponding switching process. Stochasticity manifests in the randomization of the device parameters either temporally or through the end values of the resistances [67, 60]. Supporting mathematical formulations are provided to explain the origins of the perceived stochasticity with varying levels of investigation of the underlying physical medium.

2.1.1 Fine Behavioral Modeling

Ionic interactions and oxide vacancy generation are considered the primary factors for the formation of the conducting filament (CF) between the two metal electrodes of the memristive device [65, 68]. A detailed model of the switching of the metal ox-
ides builds upon a probabilistic migration/recombination of ions and a consequently closing gap between the filaments and the metal layer. Figure 2.1a illustrates CFs variations and Figure 2.1b shows the gap formation with the corresponding motion dynamics of the ions [69]. Gap width variation is modeled with a Gaussian distribution to account for the randomness of ion propagation along the complete dimensions of the device. A circuit-compatible model for this innate stochasticity [70] conveys the stochasticity by adding variations to the ionic mechanisms and physical equations that affect the set kinetics of the device behavior.

2.1.2 Abstract Switching Model

A quantifying model that builds primarily on experimental data acquisition and consequent outcome fitting to a statistical distribution is discussed in [62, 69]. The experiments conducted involve applying an external stimulus to the terminals of the memristor and recording the time required for the device to switch. For a particular device, the experiments are repeated several times for every input pulse. In each trial, once the memristor switches to the ON/OFF state, the time since the application of
the input pulse is recorded, and the device is then reset to its initial OFF/ON state. The experimental values are mainly fit to probability distributions for the time it requires the device to change state and for the switching event. A variety of distributions have been reported in the literature, including Poisson \[62\], log-normal \[69\], Gaussian \[65\] and Weibull \[60\] distributions.

### 2.1.3 Resistance Variation Model

This analytical modelling captures the response of the device and its corresponding resistance states \(R_{\text{ON}}\) and \(R_{\text{OFF}}\). It builds on the cycle-to-cycle variation of the resistance values with attributes originating from the probabilistic formation/rupture of the conducting filament. This status arises due to the non-deterministic residue that remains after the rupture of the filament in the reset process. The resultant output of this inherent irregularity is the variation of the high and low resistance states, where the values are found to follow a log-normal distribution as fitted to the experimental measurements \[61\], \[63\].

### 2.2 Stochastic Memristor Modeling

In regards to circuit-based design and applications, the abstract switching model serves as the best fit between the very detailed physical analysis and the very high perspective on the end values of the resistances. To capture the variable operation of the memristor in a circuit-compatible manner while overlying the physical medium, and given the need to intrusively change the constituting equations, we have established a statistical model based on the probability distributions of the switching events and the corresponding variation of the threshold under weak programming conditions (i.e., conditions under which the sub-threshold voltage would have a vital effect on the subsequent behavior).
2.2.1 Stochasticity Distribution

Two distributions are frequently encountered: the Poisson distribution [62], which is mapped to amorphous silicon (a-Si) devices with a dominant filament formation principle; and the log-normal distribution encountered mostly with titanium dioxide (TiO$_2$) devices with gap width variation [69]. The implication of the switching events and the wait-time distributions on the device operation is a probabilistic switching behavior. The threshold voltage for the state change is no longer fixed but rather varies as a function of the amplitude and the temporal applicability of the input bias.

2.2.1.1 Poisson Distribution

In this model, a single or dominant CF formation is assumed to be responsible for triggering the switch to the ON state. The formation process is based on the hopping of the metal particles, which are positively charged, into the trapping sites within the a-Si layer. It is considered a step-by-step chain-growing process that leads eventually to the complete filament structure [62, 71]. A conducting path is then formed through the chain of the metal particles that leads to the resistance change. The switching rate observed is highly dependent on the input bias, and it basically roots back to the thermally activated process of the metal particle hopping. The activation energy ($E'_a$) is bias-dependent and consequently, is the hopping rate ($\Gamma$) that is determined according to

$$\Gamma = \frac{1}{\tau} = v e^{-E'_a(V)/k_B T}$$  \hspace{1cm} (2.1)

where $v$ is the attempt frequency for the particle hopping, $k_B$ is the Boltzmann’s constant, and $T$ is the absolute temperature. With the application of the input bias, the activation energy is lowered, thus resulting in a variation in the waiting times and switching rates.
A simple relation between the switching probability and its sole dependence on the time and amplitude of the voltage applied to the terminal of the device is formulated. The switching is expected to occur at an earlier time with higher input voltage compared to lower levels. An exponential relation is found between the characteristic wait time required for the device to exhibit a switching event and the applied input bias as shown in (2):

\[ \tau = \tau_0 e^{-\frac{V}{V_0}} \]

(2.2)

where \( \tau_0 \) and \( V_0 \) are considered as fitting parameters that are characteristic of the device and its constituting material. Explanation of the physical origins of these parameters is provided in the supporting information for [62].

\[ \tau_0 = \frac{1}{vE_a/k_BT} \]

(2.3)

\[ V_0 = \frac{2nk_BT}{e} \]

(2.4)

\( E_a \) is the activation energy at zero bias. It has a linear relationship with the voltage

![Figure 2.2: Linear fitting for the wait time in relation to the input voltage. Experimental fitting based on the data provided by [62].](image)
dependent activation energy, 
\[ E_a = E'_a(V) + eV/2n. \]

Where \( e \) stands for the electron charge, \( V \) is the input bias, and \( n \) corresponds to the number of trapping sites within the device. Experimentally, on a logarithmic scale, the input voltage \( V \) and the average waiting time \( \tau \) are linearly related, as shown in Figure 2.2 and according to

\[
\log_{10}(\tau) = \alpha_0 V + \epsilon
\]  

where \( \alpha_0 \) and \( \epsilon \) are fitting parameters whose values are set according to the memristor model used. Based on the physical attributions and the corresponding parameter impact, the switching event is probabilistic and follows a Poisson process. This is largely due to the conducting mechanism, which involves individual metal particles hopping into the trapping sites, thereby fitting into the model of a summation of independent events that lead to the exponential distribution of the switching wait times. At a certain point in time \( t \), the probability of occurrence of a switching event within an infinitesimal interval \( \Delta t \) is depicted as follows

\[
P(t) = \frac{\Delta t}{\tau} e^{-t/\tau}
\]  

This added feature paves the way for a control condition with respect to the needed output behavior. In other words, the level of switching is controllable with the input voltage. Although the individual switching points are random at instants of time, in general they exhibit a mean activity that follows a Poisson process.

### 2.2.1.2 Log-Normal Distribution

An alternative statistical fit to the switching behavior is observed in [69]. The experiments are conducted on a titanium-dioxide device in a fashion similar to the above-described model. However, the switching is better fit to the log-normal distribution. Moreover, different behavior is apparent in the ON and OFF switching
The switching mechanism in this type of device is primarily based on electron tunnelling through an insulating gap. The application of an input voltage or current accordingly modifies the width of this gap and the device resistance [72]. A positive bias leads to an increase in the gap width as the oxygen vacancies, which are positively charged, are repelled towards the conducting channel, thereby leading to a higher resistance or to OFF switching. On the other hand, a negative bias causes the vacancies to move away from the channel and thus leads to a reduction of the gap width and to device resistance or ON-state switching. The origins of this perceived stochasticity are attributed to density fluctuations of the distributions of the drifting vacancies/dopants. In other words, statistical variation in the vacancy concentrations rather than a smooth dispersion is the primary source of variability that leads to the
Table 2.1: Fitting parameters for the characteristic wait time

<table>
<thead>
<tr>
<th></th>
<th>Poisson</th>
<th>Log-normal (ON)</th>
<th>Log-normal (OFF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_0((s V)^{-1})$</td>
<td>-2.67</td>
<td>2.26</td>
<td>-1.49</td>
</tr>
<tr>
<td>$\epsilon(s^{-1})$</td>
<td>5.43</td>
<td>5.69</td>
<td>5.67</td>
</tr>
</tbody>
</table>

log-normal switching process depicted in,

$$F(t_{\text{switch}}, \tau, \sigma_t) = \frac{1}{2} \text{erfc} \left[ -\frac{\ln(t_{\text{switch}}/\tau)}{\sqrt{2}\sigma_t} \right]$$

(2.7)

where $F$ stands for the cumulative distribution function (CDF) for the log-normal distribution, $t_{\text{switch}}$ corresponds to the cumulative time, and $\text{erfc}$ is the complementary error function. The distribution is characterized by its median time to switch $\tau$ and the standard deviation $\sigma_t$. Thus, from the experimental data, depicted in Figure 2.4a and 2.4b, there is no clear relation and a very weak dependence is available between $(\sigma)$ and the applied voltage. Thus, reported hard values and interpolation are required to have the corresponding fitting. According to [73], the switching probability within a $(\Delta t)$ is given by,

$$P(t) = \frac{\Delta t}{\text{mean}} = \frac{\Delta t}{\tau} e^{-\sigma^2/2}$$

(2.8)

Figure 2.4: (a) The experimental sigma values represent orders of magnitude for a given threshold (measured in ($\mu$s)) extracted for the ON switching case (b) OFF switching case.
where $\tau$ is considered the median in the log-normal distribution, whereas the mean $(\mu)$ is $\mu = \tau e^{\sigma^2/2}$. The chosen setting, the distribution in particular, depends on the speed and mechanism of the switching operation. Moreover, within the realm of a single model, the range of values used for these parameters dictates the operation region across the time of the corresponding memristor. As illustrated from the above equations, infinitesimally small values of $(\tau)$ lead to almost abrupt switching, whereas with larger values of $(\tau)$, the operation reverts back to its original dynamics.

2.2.2 Behavioral Stochastic Modelling

The general equations signifying the conductance change over time are summarized in the following two functions [10, 74]

\[
\begin{align*}
  u(t) &= g(w, s, t)s(t), \\
  \frac{dw}{dt} &= f(w, s, t),
\end{align*}
\]

(2.9)

where parameters $s(t)$ and $u(t)$ are the corresponding input and output functions, respectively. These parameters form the I-V relation for the memristor depending on whether the device is voltage or current driven. Similarly, the function $g()$ represents the memconductance or memresistance according to the set parameters. The liberty in the behavioral modelling of the device is mainly captured within the formulation of the state derivative $dw$. On the other hand, a general incorporation of the added stochasticity is modelled as a function setting for the threshold and is passed over to the state function $w$. It could be easily mapped into any of the threshold-based functions where the switching location is the sole parameter affected with stochasticity while preserving the kinetics of the original models. The variability of the threshold
voltage with the incorporation of the stochastic term is reflected in the following:

\[
d\mathcal{V}_T = \alpha \theta(V_{T_0} - \mathcal{V}_T)dt + (|V| - \Delta \mathcal{V} - V_{T_0})dN(\tau)
\]  

(2.10)

The functions \(\theta()\) and \(N(\tau)\) are the step function and the Poissonian/log-normal process, respectively. \(V\) is the input voltage \(\delta V\) stands for the infinitesimal change for the voltage to allow for the setting of the threshold voltage, \(\alpha\) is a fitting parameter set according to the model used, and \(V_{T_0}\) is the almost deterministic threshold voltage of the device. Thus, at each instant of time, the switching voltage has a probability of changing according to the stochastic process specified in \(N(\tau)\).

---

**Algorithm 1: Threshold voltage Statistical Variation**

Choose the switching distribution;  
Calculate the characteristic parameters;  
for every time instant do  
    Calculate the corresponding probability \(P\);  
    Sample from a uniform random distribution \(R\);  
    Compare;  
    if \(P \geq R\) then  
        threshold voltage is set to the current input voltage;  
    else  
        The threshold voltage is not changed;  
    Pass on the threshold voltage

---

2.2.3 Verilog-a Model

To allow for simulation and testing in an easy integration with circuit simulators, Verilog-a provides an intermediate level of abstraction, as it supports the integration of random variables and processes into the sequence of instructions. Hence, building on the statistical characteristics of the switching process, the variability lies upon the shift in the switching point for the memristor, as its particular value depends on the probability calculated at every time step. Cumulatively, the complete operation fol-
ows a Poisson/log-normal distribution, whereas the corresponding switching events are random in time and decided upon at every instant, as depicted in Algorithm 1. At each instant of time, the instantaneous switching probability for the particular distribution is calculated and compared to a sample from a uniform random distribution. In case the calculated probability is greater than the chosen sample, the threshold voltage takes the value of the instantaneous voltage, and the switching occurs accordingly. Otherwise, the threshold voltage remains intact at the value that is almost deterministic $V_{to}$. This stochasticity modelling is added to any of the readily established memristor models. The main idea is to have the threshold conditions variable at instants of time and to provide them as an input to the corresponding switching criteria. The Verilog-a code for the induction of the variability is illustrated in Appendix A.1.

2.2.4 SPICE Modelling

Simulation Program with Integrated Circuit Emphasis (SPICE) is the most common general-purpose, circuit-level simulation environment [75]. However, adding dynamic variability into the simulation is a challenging task, as it does not directly support the generation of variables with a certain probability distribution. Nonetheless, a source of randomness is induced with the use of a thermal resistor. In this mode of operation, a regular resistor would be mapped into an equivalent voltage source in series with the actual resistor [76], as depicted in Figure 2.5a. The resultant output is white noise with a mean square value of,

$$v_n^2 = 4k_B TRB$$

where $k_B$ corresponds to the Boltzmann constant of $1.38x10^{-23}J/k$, T is the temperature in Kelvin, R is the actual resistor value, and B is the bandwidth of operation that is specified at run time. The generated noise parameter’s probability density
Figure 2.5: (a) A circuit diagram for the resistor with activated thermal noise, where the resistor will have a noise source in series with it. (b) A histogram of the thermal noise with a Gaussian distribution, that is centered at the origin with the standard deviation of the noise. set to $\sigma = \sqrt{v_n^2}$.

This function reflects a Gaussian distribution, as shown in Figure 2.5b.

2.2.4.1 Probability Transformation

In the stochastic memristor modelling, a uniformly distributed variable is required at each instant of time to compare it with the instantaneous probability of the memristors statistical model. Hence a Probability-transformation step is initially performed to change the Gaussian values generated from the thermal resistor into uniformly distributed values. The transformation of the Gaussian distribution $X \sim N(\mu, \sigma)$ to a uniform distribution $U \sim [0, 1]$ builds on the cumulative distribution function and the inverse transform method,

$$U = \frac{1}{2} \text{erfc}(\frac{-y}{\sqrt{2}})$$

(2.12)

where erfc stands for the complementary error function that is linearly approximated as

$$\text{erfc}(y) \approx \sqrt{1 - e^{-(4y^2/\pi)}}$$

(2.13)

and $Y$ is the normalized Gaussian variable ($Y \sim N(0, 1)$) that is obtained by di-
Figure 2.6: The variability of the threshold is mapped into a voltage controlled switch that either keeps the threshold intact or takes the value of the input in case the condition of $V_c$ is satisfied. This threshold variation is an add-on to any of the threshold-based models to model the dynamic stochasticity over time.

Providing the thermal noise generated from the resistor $X \sim N(0, \sigma)$ by the standard deviation ($\sigma = \sqrt{V_n^2}$).

### 2.2.4.2 Stochastic Modeling

The basic concept in the statistical modelling lies in having the variable threshold set according to the satisfaction of the switching condition. Either the threshold remains set to the original deterministic threshold of the device, or to the current input voltage. Hence, a voltage-controlled switch is used to model this change. The two terminals of the switch are the deterministic threshold and the input voltage, which are designated by $V_{\text{set}}$ and $V_{\text{in}}$, respectively. As depicted in Figure 2.6, the switch will be connected to Terminal 1, corresponding to the original threshold, and will be shifted to Terminal 2, corresponding to the instantaneous input voltage once triggered by the controlling voltage $V_c$. The controlling voltage is the difference between the probability calculated from the distribution ($P = \frac{\Delta t}{\mu}$) and the uniform random sample transformed ($R$). The switch shifts to Position 2 in case the value of $V_c$ is positive, and it remains intact otherwise. The SPICE code for the transformation of the thermal noise values and its incorporation into a threshold-based model of the...
memristor is illustrated in Appendix A.2.

### 2.3 Model Verification

The proposed statistical model is to be verified against the experimental data generated in the literature. It is fit to different probability distributions. It needs to capture the temporal behavior of the device under diverse input excitation and the impact of the probabilistic output. The Poisson and log-normal distribution models are put to the test.

![Figure 2.7](image1)

**Figure 2.7:** (a) Data fitting for the switching times reported in [62] for varying input voltage pulse of 2.6V (b) 3.2V and (c) 3.6V respectively. The characteristic switching times extracted increased with the smaller input voltage. The Distributions were fitted with the characteristics parameters accordingly.

![Figure 2.8](image2)

**Figure 2.8:** (a) The simulated model incorporating the variability of the threshold voltage for the same set of input voltages for the reported measurements. The voltage pulses of 2.6V (b) 3.2V and (c) 3.6V were applied and mapped into the fitted Poisson.
2.3.1 Poisson Fitting

The basis of the proposed model is rooted in the quantification process applied to the abstract switching events of the memristor. In [62], a Poisson-like process is experimentally extracted from the behavior of the fabricated devices. The extent to which the statistical model fits into the distribution is measured through the application of a set of voltage pulses with different amplitudes. The applied input voltages are chosen to match the reported values for the measurements. The process of the measurement is set for each particular voltage.

For every trial, a pulse is applied, the time at which the switching occurs is recorded, and the device is reset back to its original state. And then another trial starts. For simulation, a Python script is used to generate the SPICE netlist and to extract the resulting switching times. The algorithm is run for ten thousand trials for each voltage input.

2.3.2 Log-Normal Fitting

The verification of this particular distribution is based on the cumulative switching times and their corresponding probability, as extracted from the experimental measurements in [69]. As depicted in Figures 2.9a and 2.9b, the ON and OFF cumulative switching probabilities are fit to the Log-normal sigmoid-shaped function. Similarly, as illustrated by the earlier simulations, the trains are run ten thousand times and the switching times are recorded. The switching probabilities are then calculated by dividing the number of occurring events for each switching time over the total number of trials. Figures 2.10a and 2.10b show the corresponding simulation results. A clear dependence on the voltage is apparent in both cases of switching. The probability of switching becomes higher much faster with higher input voltages. The time constants also follow the same behavior as that of the experimental data where longer times are required to switch for smaller voltages. However, there is a minor shift in terms
of the time constants and the shaping parameters from that of the experimental data due to the dual approximation that is proposed in [69] for \( \tau \) and \( \sigma \). Nonetheless, the overall behaviour of the system is captured by the simulation model by using the fitting parameters from Table 2.1.

### 2.4 Memristor Models

The dynamics of the memristor device, and the modelling attempts to adequately capture its non-linear behaviour have been under thorough investigation. Models proposed along several domains have set the standards in terms of complexity, accuracy, and practicality in simulation and circuit design [78, 79, 80, 81, 82, 83, 84, 85, 86]. In general, charge- or flux-controlled operations are the two main variants imposing
Figure 2.11: (a) Internal dynamics of the voltage controlled model in the deterministic operation. (b) Resultant variation in the threshold voltage with the mere shift in the location of switching and preservation of the original function kinetics.

a dependence on the history of the applied current or voltage across the terminals of this nano-scale structure. We are considering three distinct models that cover a broad spectrum of current- and voltage-driven memristors, and we are inducing stochastic threshold variations that allow for their incorporation in further applications. The abstract switching model with the bipolar memristors, the analytical memristor model, and the original Picket model are further illustrated.

2.4.1 Bipolar Memristors with Threshold

In this model [16, 87, 88], the authors introduce a general model of the circuit element that exhibits a memory characteristic with a rate of change related to the applied voltage. In its deterministic version, no response is perceived in the memristor below the threshold voltage. At input voltages higher than the threshold, the switching state starts to increase at a certain rate $\beta$ that acts as the slope of the internal dynamics of the memristor. Figure 2.11a depicts the general shape of the threshold-based memristive device. The behaviour is modelled by a linear I-V relationship and a shaping function that describe the internal process and the sensitivity to the limiting
parameters as illustrated in,

\[ I = X^{-1}V_M \] (2.14)

\[
\frac{dX}{dt} = f(V_M)[\theta(V_M)\theta(R_{off} - X) + \theta(-V_M)\theta(X - R_{on})] \] (2.15)

\[
f(V_M) = \beta V_M - 0.5\beta[|V_M + V_t| - |V_M - V_t|],
\]

\[
dV_T = \alpha \theta(V_{T_0} - V_T)dt + (|V| - \Delta V - V_{T_0})dN(\tau), \] (2.16)

The memristance is reflected within the state variable \(X\) and is limited by the upper and lower boundaries \(R_{off}\) and \(R_{on}\). The function \(\theta\) is the step function responsible for enforcing boundary conditions. Incorporating the variability into this model follows a straightforward approach with the threshold voltage \(V_t\) becoming variable in a way abiding by the stochasticity measures explained earlier and according to (2.16). This behaviour is reflected in the shaping function fluctuation at the point of switching while preserving the kinetics of the system, as shown in

Figure 2.12: (a) Response to a sinusoidal input voltage showing the larger variation particularly at closer voltages to the original threshold. (b) A finer resolution of the threshold variation.
Figure 2.13: (a) Deterministic output of the model with the parameters specified in [88] (b) The stochastic output with the outer hysteresis setting confining the sub-threshold stochastic switching

Moreover, the added variability allows for sub-threshold switching in a non-deterministic fashion, where the value of the point of switching is related to the input voltage application time and amplitude. The variation of the threshold is apparent in Figure 2.11b (and in the finer-resolution version in Figure 2.12b), which depicts the variation in response to the input voltage application. The impact on the current-voltage relationship is directly visible in the characteristic hysteresis. Instead of having a single response for a sinusoidal input, the output is tainted with added inner hysteresis embedded within the outer limits set by the deterministic threshold voltage. Figure 2.13a shows the original deterministic output, whereas Figure 2.13b shows the stochastic behavior.

### 2.4.2 Analytical Memristor Model

A slower version of a memristor particularly suited for real-time and neuromorphic applications is presented in [89]. Several fabricated devices have been aligned with the equations of the proposed model based on a fine tuning of the fitting parameters. It is set a generalized form that is characterized by a nonlinear I-V relationship as
shown in (2.17),

\[
I(t) = \begin{cases} 
  a_1x(t)sinh(bV(t)), & V(t) \geq 0 \\
  a_2x(t)sinh(bV(t)), & V(t) < 0 
\end{cases}  
\]  

(2.17)

where the parameters \(a_1\) and \(a_2\) are used as amplitude parameters to account for the conductivity of the different device structures along with polarity of the applied input. Moreover, a control factor is also included, where \(b\) calibrates the intensity of the threshold in relation to the voltage amplitude. The second characteristic equation mainly reflects the state variable \(x\) with two functions responsible for controlling the operation: a threshold imposing factor \(g(V(t))\) and a boundary condition \(f(x(t))\).

\[
\frac{dx}{dt} = g(V(t)).f(x(t)) 
\]  

(2.18)

The function \(g(V(t))\) introduces the threshold voltage in the positive and negative regions \(V_p\) and \(V_n\) respectively, with no change allowable between these limiting points. Furthermore, the rate of change beyond the set threshold is controlled by the variables \(A_p\) and \(A_n\):

\[
g(V(t)) = \begin{cases} 
  A_p(e^{V(t)} - e^{V_p}) & V(t) > V_p \\
  -A_n(e^{-V(t)} - e^{-V_n}) & V(t) < -V_n \\
  0 & -V_n \leq V(t) \leq V_p 
\end{cases}  
\]  

(2.19)

Incorporating the stochasticity into this model adds a lot of value to it, as a reasonable level of variability is inherent in the operation of neuromorphic circuits \[90, 91\]. Hence, the stochastic version of the model allows for an added noise feature embodied within a single circuit element that exhibits intrinsic agility and integration over large-scale systems. The mathematical form of the endorsed stochasticity on the
positive and negative thresholds is as follows:

\[
dv_{p/n} = \alpha \cdot \theta(v_{p/n} - v_{p/n_0})dt + (V - \delta V - v_{p/n})dN(\tau)
\]  

From a modelling perspective, tackling the stochasticity application in this model requires scaling the parameter \(\alpha_0\) in response to the threshold voltage modification according to the fit device under consideration. Based on the proposed fitting parameters of the device described in [92], the positive threshold voltage is set to 1.5 V and the negative threshold is set to 0.5 V. Figures 2.14a and 2.14b convey the abrupt spikes in the threshold voltage that occur as a direct consequence of the stochasticity applications.

The threshold of the device is modified based on the probability of operation and consequent switching. Furthermore, the variation is conveyed in the I-V relationship, where the point of operation diverges from the deterministic case due to the induced non-determinism of the underlying behaviour. Figure 2.15a shows the original output response, whereas Figure 2.15b reflects the stochastic output in comparison with the deterministic states.
Figure 2.15: (a) Deterministic output of the model parameters fit for the model in [92] for a sinusoidal input, $V_p = 1.5$ V, $V_n = 0.5$ V, $A_p = 0.005$, $A_n = 0.08$, $x_p = 0.2$, $x_n = 0.5$, $\alpha_p = 1.2$, $\alpha_n = 3$, $a_1 = 3.7(10^{-7})$, $a_2 = 4.35(10^{-7})$, $b = 0.7$, and $x_o = 0.1$. (b) Stochastic output with the same parameters with a modified intensity parameter $\alpha_0$ to fit the positive and negative threshold accordingly.

### 2.4.3 Simmons Tunnel Barrier Model

The first reported fabricated device is presented in Pickett’s model [72]. The model is fit to the measurement and experimental data with a non-linear representation of the bipolar switching. The conductance change is based on a Simmons tunnel barrier width $w$ that lies in series with an internal resistor that cumulatively affects the overall device characteristics. The model is current driven, where the applied current triggers the change of the state variable $w$ with time due to its exponential effect on the velocity of the ionized dopants. This dependence leads to non-symmetrical ON and OFF switching dynamics with an ionic diffusion that is highly susceptible to the polarity of the input. The general equations that portray the behavior of the memristor model are rooted back to the physical realization attained for the fabricated devices and fitted through and extensive regression analysis.

**Off switching ($i>0$)**

$$\frac{dw}{dt} = f_{ogs\sinh} \left( \frac{i}{i_{off}} \right) \exp \left[ -\exp \left( \frac{w - a_{off}}{w_c} - \frac{|i|}{b} \right) - \frac{w}{w_c} \right]$$

(2.21)
ON switching $(i<0)$

\[
\frac{dw}{dt} = f_{on} \sinh \left( \frac{i}{i_{on}} \right) \exp \left[ - \exp \left( -w - a_{on} \frac{w}{w_c} - \frac{|i|}{b} \right) - w \right] \quad (2.22)
\]

The rate of change in the memristance is tuned through the fitting parameters, $f_{on}$ and $f_{off}$. The state variable $x$ is confined within its set boundaries through the parameters $a_{on}$ and $a_{off}$ on both edges. The current-voltage relationship is based on the Simmons tunnelling model [93] with fitting parameters set according to the measured data and the SPICE model provided in [94]:

\[
i = \frac{j_0 A}{(\Delta w)^2} \left( \phi_1 e^{-B\sqrt{\phi_1}} - (\phi_1 + e|v_g|)e^{-B\sqrt{\phi_1 + e|v_g|}} \right) \quad (2.23)
\]

The stochasticity is then incorporated as in the prior models where the current threshold is the affected variable:

\[
di_{on/off} = \alpha \cdot \theta(i_0 - i_{on/off})dt + (w - \delta w - i_0)dN(\tau). \quad (2.24)
\]

On a finer scale, with respect to the model-fitting parameters, the equation for the variability is adjusted by multiplying the instantaneous resistance and the input
Figure 2.17: (a) Deterministic output of the model parameters for a sinusoidal input, $i_{\text{off}} = 115 \, \mu\text{A}$, $i_{\text{on}} = 8.9 \, \mu\text{A}$, $a_{\text{on}} = 2e-9$, $a_{\text{off}} = 1.2e-9$, $f_{\text{on}} = 40e-6$, $f_{\text{off}} = 3.5e-6$, $b = 500e-6$, $w_c = 107e-12$, $D = 3e-9$, $R_{\text{on}} = 100 \, \omega$, $R_{\text{off}} = 20 \, k\omega$. (b) Stochastic output with the same parameters with a modified intensity parameter $\alpha_0$ to fit the positive and negative threshold accordingly. The Simmons tunnel-barrier model has threshold currents in the range of $\mu\text{A}$. Thus, the instantaneous modification of the threshold is not easily conveyed in the hysteresis figure. The primary impact is basically apparent in the values of the underlying resistance. However, holding the threshold voltage at the point of variation for an arbitrary period prior to resetting it to its original value allows for capturing the variation visually in the output behavior. Figure 2.16 shows the threshold voltage variation over the sinusoidal input.

The threshold variation has a direct effect on the hysteresis output. Figure 2.17a shows the original deterministic output, whereas Figure 2.17b reflects the stochastic output. As depicted, the internal kinetics are smooth and slow resulting in minor variations in the location of the switching point with conservation of the overall system mechanism in terms of the speed of operation and dynamics.
Table 2.2: Memristor models with incorporated stochasticity

<table>
<thead>
<tr>
<th>Model</th>
<th>State Equation</th>
<th>Stochastic Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bipolar Memristor Model [16]</strong></td>
<td>$\frac{dX}{dt} = af(V_M)\theta(R_{off} - X) + \theta(-V_M)\theta(X - R_{on})$</td>
<td>Threshold Voltage $V_t$</td>
</tr>
<tr>
<td></td>
<td>$af(V_M) = -b(V_M - 0.5)|V_M + V_t| - |V_M - V_t|$.</td>
<td></td>
</tr>
</tbody>
</table>
| **Analytical Memristor Model [89]**        | $g(V(t)) = \begin{cases} 
\frac{dV}{dt} = g(V(t), f(x(t))) & , V(t) > V_p \\
A_p(e^{V(t)} - e^{-V(t)}) & , V(t) < V_n \\
-A_n(e^{-V(t)} - e^{V(t)}) & , -V_n \leq V(t) \leq V_p \\
0 & , \text{otherwise} \end{cases}$ | Threshold voltages $V_p$ and $V_n$ |
| **Simmons Tunnel Barrier Model [72]**      | $\frac{dv}{dt} = f_{on} \sinh \left( \frac{v}{v_{on}} \right) \exp \left[ -\exp \left( \frac{v - v_{off}}{v_{on}} \right) - \frac{v}{v_{on}} \right]$ | Threshold parameters $i_{off}$ and $i_{on}$ |
|                                            | $\frac{dv}{dt} = f_{on} \sinh \left( \frac{v}{v_{on}} \right) \exp \left[ -\exp \left( \frac{v - v_{on}}{v_{on}} \right) - \frac{v}{v_{on}} \right]$ |                     |

2.5 Summary

This chapter constitutes a study of stochasticity in emerging non-volatile memory technologies and presents a circuit-level statistical model for the probabilistic switching behavior of the memristor under weak programming conditions. Model verification via distribution fitting and mapping to the original experimentally extracted device characteristics reveals the accuracy and validity of the proposed model. We induce stochasticity into several established models and discuss the corresponding adjustment and fitting parameters set with the depiction of the output hysteresis modification with the added variability.
Chapter 3

Approximate Computing

The concept of a sensible machine has recently emerged as a result of the onset of the cognitive computing era [95], with operation principles that are mainly rooted in inspirations from human brain behaviour. The cognitive abilities, dimensionality, operation demands, and costs associated with the concept are quite intriguing and have attracted significant attention from the research community. The attainment of such a machine is contingent upon radical change in the design of computing systems [11], whether in terms of shifting from the conventional von Neumann architecture (with its neuromorphic applications), having processing and memory co-localized with in-memory computing designs [96], or extending the current architectures with approaches such as stochastic [41] and approximate computing [97]. Conventional logic design usually prioritizes the repeatability and accuracy of operation, which is a practice that pushes the use of the available resources to their limits and hence leads to overdesigning and costly error-correcting schemes to ensure precision [98]. However, when dealing with applications in which the notion of accuracy cannot be formally set and is instead controversial or dependent on perceptions, extra design margins are added. A trade-off arises between the level of accuracy and other performance metrics (such as energy efficiency and delay), which promotes the main operating principles of approximate computing [99].

The approximate mode of performing computations has been examined at different levels of abstraction. This includes the application and algorithmic levels, with program execution and instruction misses [100]; the architectural level, with dedi-
cated processors and accelerators \[99\]; and the circuit level, with the modification of the number of electronic components or the scaling of the supply voltage \[101\]. Nonetheless, with Moore’s law reaching an end, traditional electronic devices are facing extreme challenges to cope with the advancement and integration requirements. Alternative solutions are starting to be established with more CMOS and additional Moore design streams \[11\].

In this chapter, we introduce the following

1. A concept of approximate computing based on stochastic memristive elements, which represents the first proposition to combine emerging technologies and unconventional computing approaches;

2. A novel circuit-based technique to achieve approximation and adaptability, which mostly builds on the variability of the memristor and is related to the amplitude of the applied voltage and the time period;

3. A mode of operation that allows the accuracy, energy level, and delay to be configured based on the design requirements and available resources;

4. A formulation for the output of the gates and its corresponding cascades that is based on probabilistic digital gates and mathematical distributions for switching events; and

5. An incorporation of probabilistic logic into error-resilient applications and a demonstration of the corresponding impact on overall system performance.

### 3.1 Memristor-Based Logic

In their abstract form, bipolar memristors can be seen as ideal switches and consequently used in the design of logic operations. They can hold one of two states: high resistance (noted as $R_{\text{off}}$ or what corresponds to logic 0) and low resistance (noted
as \( R_{on} \), corresponding to logic 1). Several designs have been proposed for using the memristor in the logic design, starting with the imply operator that is considered the basis for the Boolean gates [102]. A CMOS-like logic is suggested in [25], and a third approach builds on sequential logic [103] with several cycles needed to perform a Boolean operation.

### 3.1.1 Imply Logic

The basic operation principle of the imply operator lies in the conditional switching of a secondary memristor in response to the state of the primary one. Figure 3.1(b) depicts the underlying circuit for this digital application. The two memristors, primary (p) and secondary (q), are separately connected to tri-state voltage drivers at one end and to a common resistor \( R_{G} \) at the other [102, 104]. The imply operation is based on a change of the state of the output (i.e. the secondary memristor q) once the primary memristor is 0. Ideally, once the primary memristor is in the low state and the application of a toggling voltage of \( V_{COND} \) is smaller than the switching threshold,

![Diagram](image)

**Figure 3.1:** (a) Imply logic circuit utilizing the memristors as abstract switches to perform the *imply* (\( \Rightarrow \)) operation [102]. (b) The truth table for the 2-input *imply* operation.

<table>
<thead>
<tr>
<th>Case</th>
<th>p</th>
<th>q</th>
<th>( p \rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
the state of the memristor is preserved. In contrast, the secondary memristor (i.e. the output) is set, seeing as the voltage drop across its terminal is sufficient to switch it to the ON state. A truth table for the operation that highlights the different cases for the inputs and corresponding output is presented in Figure 3.1b.

### 3.1.2 CMOS-Like Memristor Logic

In an alternative approach to designing Boolean operators, a memristor can be utilized to realize logic circuits [25]. Polarized memristors resembling pmos and nmos transistors are connected to form the complete set of logic operators. The different polarities for the memristors’ switching behaviour aid in achieving the required output behaviour. Figure 3.2a shows the schematic for an inverter circuit using memristors, in which the operation is based on having a positive/negative input voltage that corresponds to 1/0, respectively. In terms of the output, values closer to 0 are

![Figure 3.2: ((a) The schematic for the memristor-based inverter in a similar manner to the CMOS [25]. (b) The input signal and the output signal. The output is considered high if the $|V_{out}|$ is close to $|V_{dd}|$ and low in case it is closer to 0.)](image-url)
considered low or ‘0’ and values closer to $|Vdd|$ are considered high or ‘1’. With a pulsed input voltage, the inverter operation results in the inverse of the input, as shown in Figure 3.2b.

### 3.1.3 Sequential Logic

The principles of operation of the sequential logic approach rely on using the memristor’s two terminals as the inputs of a logic operator. Building on a bipolar switching in which opposite polarities are used for the SET and RESET of the memristor state, a positive voltage (greater than the threshold) across the terminals ($T_1$ and $T_2$) moves the memristor into the ON state or ‘1’; in contrast, a negative voltage shifts it back to the OFF state or ‘0’. No change is reflected in the memristor state ($z$) in case the $T_1$ and $T_2$ terminals are at the same potential. The truth table of this behaviour is depicted in Figure 3.3.

To realize a Boolean operation, a sequence of cycles needs to be applied. For the logic gate, three cycles are necessary for the correct output of the operation. The first cycle is considered the initialization stage, in which the memristor is put in a predetermined state that also facilitates the attainment of the required operation for

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$T_2$</th>
<th>Voltage</th>
<th>State ($z$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$z$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$-V$</td>
<td>OFF</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$V$</td>
<td>ON</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$z$</td>
</tr>
</tbody>
</table>

Figure 3.3: The memristor terminals $T_1$ and $T_2$ correspond to the two inputs ‘p’ and ‘q’ for logic operations. The truth table is shown for the state of the memristor and the corresponding input voltage across its terminals.
the applied input parameters. The complete list of Boolean operators and the corresponding application sequence are discussed in details in [103]. The main condition for the correct operation is to have the applied voltage larger than the threshold. Considering a device with set/reset threshold voltage of ±4.6V, the applied voltage should thus be around ±5V to ensure a proper switching and state change.

3.2 Probabilistic Memristor Logic

Based on the sequential logic highlighted earlier, a single memristor behaviour governs the output values. Under sub-threshold excitation, the memristor is hence characterized by a probability of switching $P_s(t)$. With the incorporation of this variability, a set of assumptions is put forward in this analysis to ensure a common ground and provide a simple overview of the expected outcome. These assumptions are as follows:

1. The initialization stage is deterministic;
2. The inputs $p$ and $q$ to the logic gates are deterministic; and
3. The operation of the memristor is probabilistic with a generic switching probability $P_s(t)$

Instead of having a certain correct output, the values are thus masked with the probability of switching and the sequence of applied input parameters. For each entry

\[
\begin{array}{ccc}
\text{Cycle 1} & \text{Cycle 2} & \text{Cycle 3} \\
T_1 & '1' & '0' & '1' \\
T_2 & '0' & q & p
\end{array}
\]

Figure 3.4: The three cycles needed for obtaining the NAND operation of the input pair $(p, q)$. The cycles are applied sequentially with the high and low values of the input determining the polarity and level of the applied input voltage.
in the truth table, the certainty of having a correct output is characterized by $P_{\text{out}}$ (depending on the input setting). For each gate, the output of every entry in the truth table has a probability of being correct. This probability is determined by the memristor’s switching probability and the sequence of input application. We present a detailed analysis for a NAND gate below; with the sequence of cycles shown in Figure 3.4, all remaining gates would follow the same construct and flow, but with their own set of probabilities and consequent behaviour.

### 3.2.1 Probabilistic NAND Gate

As shown in Table 3.1, four different combinations are available for a NAND gate with the input pair $(p; q)$. In a deterministic manner, the sequence of the cycles and corresponding voltages and input combinations control the output values. However, when the stochasticity kicks in, lower voltages could trigger the switching to a stable state that will not change unless the reverse polarity voltage is applied. As a result, the gates will behave in a probabilistic manner. For the pair $(0; 0)$, the first cycle puts the memristor in the ON state. However, the second cycle does not have an effect on the memristor’s state, as a zero potential is put across its terminals. For the third cycle, a positive potential will also not alter the state as it is already ON. Having the pair $(0; 0)$ thus always produces a correct output, and $P(1) = 1$ disregarding the switching probability of the memristor.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>State ($z_1$)</th>
<th>State ($z_2$)</th>
<th>State ($z_3$)</th>
<th>Output</th>
<th>$P_{\text{out}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>‘1’</td>
<td>$P_1 = 1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>‘1’</td>
<td>$P_1 = 1 - P_s^2(t) + P_s(t)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>‘1’</td>
<td>$P_1 = 1$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>‘0’</td>
<td>$P_0 = P_s(t)$</td>
</tr>
</tbody>
</table>

Table 3.1: The truth table for the NAND gate operation with the added stochasticity of the bipolar memristor. $P_s(t)$ is the switching probability of the memristor under a particular input voltage across its terminals.
To get a correct ‘1’ output for the pair (0; 1), the memristor should switch to the ON state in the third cycle or not switch in the second and third cycles. The probability of getting the correct output is then

\[ P_{(0,1)}(1) = P_s(t) + (1 - P_s(t))^2 \]  \hspace{1cm} (3.1)

\[ P_{(0,1)}(1) = 1 - P_s(t) + P_s^2(t) \]

For the pair (1; 0), zero potential is applied in the second and third cycles. This condition moves the initialization state directly to the output, so \( P(1) = 1 \). For the pair (1; 1), the second cycle determines whether the output is correct, as the third cycle has zero potential and the previous state thus affects the output. To have a correct ‘0’ at the output, the memristor should therefore switch to the OFF state in the second cycle. Thus, \( P(0) = P_s(t) \). The overall probabilities of the NAND gate obtaining the correct ‘0’ and the ‘1’ at the output are

\[ P_o(0) = P_s(t) \]

\[ P_o(1) = \frac{1}{3}[3 - P_s(t) + P_s^2(t)] \]  \hspace{1cm} (3.2)

These theoretical formulations show the impact of the switching probability on the correctness of the output bits. The NAND gate is more biased to yield a ‘1’ value and that is clear with the high probability of getting the logic ‘1’ at the output and the minor effect of the stochasticity on the result. On the other hand, the ‘0’ output has a direct linear relationship with the switching probability: the higher the switching probability, the more accurate the ‘0’ output. Figure 3.5 shows the output probabilities for \( P_{out}(1) \) and \( P_{out}(0) \). The solid line represents the theoretical probabilistic analysis presented earlier, whereas the points represent the simulation output with the stochastic memristor element. The three-cycle operation of the NAND gate is
3.2.2 Accuracy

In addition to the single-bit probabilities, the gate’s behaviour is further quantified with the notion of accuracy. This concept is specified as

$$\text{Accuracy} = \frac{N_0}{N} P_{\text{out}}(0) + \frac{N_1}{N} P_{\text{out}}(1)$$  \hspace{1cm} (3.3)

where \( N \) corresponds to the total number of entries in a particular truth table, \( N_1 \) is the number of entries with the output set to ‘1’, and \( N_0 \) is the number of entries with the output set to ‘0’. For the NAND gate, there are four entries in the truth table (three cases of ‘1’ and only one of ‘0’). As such, the NAND operator’s accuracy
would be calculated as follows:

\[
\text{Accuracy}(\text{NAND}) = \frac{1}{4} [3 + P_s^2(t)]
\]  

(3.4)

Figure 3.6 shows the accuracy of the NAND gate as formulated earlier with respect to the memristor’s switching probability. The theoretical accuracy attained is verified by simulating the memristor 100 times for all input pairs and recording the output states. The NAND gate shows a high accuracy rate that starts from 75%. The deterministic initialization state shifts the ON state to the output in two of the possible gate input combinations. As a result, great savings in terms of simple logic operation is possible by scaling the voltage and benefiting from the memristor’s stochastic feature.

### 3.2.3 Boolean Operators

A similar analysis is conducted for all of the remaining gates that require three cycles for operation, namely AND, OR, and NOR. Figure 3.7 summarizes the probabilistic outcomes for these gates and shows the corresponding effects on performance. The theoretical and simulation results are in consensus in terms of the expected output
Figure 3.7: The probabilistic analysis for the logic operators AND, OR, & NOR with the corresponding output probabilities $P_{\text{out}}(0)$ and $P_{\text{out}}(1)$ in addition to the accuracy of operation.
for the logic operations. However, when the gates are cascaded to form an arithmetic operator, a probabilistic dimension is added into the analysis and requires further parameters to be taken into account.

### 3.3 Approximate Arithmetic Circuit

Basic arithmetic operations build on the main logic gates that are formed by a combination of the prior presented set. This section analyses the gates under probabilistic components, details the cascade required for the formation of a larger function, and presents a case study of an adder.

#### 3.3.1 Half-Adder

For incorporating the stochastic memristor-based gates into an arithmetic operator, the basic building block is a half adder. It is mainly composed of an XOR gate (for calculating the sum bit) and an AND gate (for calculating the carry bit). Figure 3.8a shows a corresponding schematic. The logic equations for the output bits $S_o$ and $C_o$ are as follows:

\[
S_o = p \ (XOR) \ q \\
C_o = p \ (AND) \ q
\]  

(3.5)

However, as illustrated earlier, only four out of the six logic gate operators are possible with a single memristor. The remaining gates XOR and XNOR gates can be attained through a combination of the basic gates. For example, an XOR gate is obtained by applying an OR and NAND to the inputs $p$ and $q$ and then putting the results back into an AND gate; the equation of the operation is

\[
p(XOR)q = (p(OR)q) \ AND \ (p(NAND)q)
\]  

(3.6)
Figure 3.8: (a) The schematic of the half-adder encompassing the XOR operation for the sum and the AND gate for the carry bit. (b) The truth table for the half adder showing the carry \( C_o \) and sum \( S_o \) values.

The assumption of having deterministic inputs is thus not valid for analysing the behaviour of this XOR gate, especially with regard to the input of the AND gate used to calculate the sum. Further to the analysis of the AND gate, a distinct set of output probabilities is thus formulated along with the precise sum and carry bit. The input probabilities are assigned as \( P_{in}(1) \) and \( P_{in}(0) \). The probabilities for the inputs \( p \) and \( q \) are considered the same as they are the OR and NAND gate outputs in the first stage (which exhibit similar probabilistic operation in terms of the output probabilities for ‘1’ and ‘0’). The output probabilities for the AND gate with probabilistic inputs are

\[
P_{out}(0) = \frac{1}{3} (3 + (2P_{in}(0) - P_{in}(1) - 2)P_s(t)) + (1 - P_{in}(1) - P_{in}^2(0) + P_{in}^2(0)P_{in}(1))P_s^2(t))
\]

\[
P_{out}(1) = P_{in}(1)P_s(t) + (P_{in}(1) - 1)P_{in}(1)P_s(t)
\]

As depicted in the truth table for the half adder in Figure 3.8b, the sum bit \( S_o \) will have two entries for each ‘0’ and ‘1’. The accuracy (or precision) of the sum bit is
Figure 3.9: The accuracy of the half–adder for the sum $S_o$ and carry $C_o$ bits. The simulation results for both bits fit the theoretical analysis. Moreover, the performance of the carry (for the deterministic AND) starts lower than the sum, but catches up after 0.3 switching probability and better accuracy is attained.

hence specified as

$$Accuracy(S_o) = \frac{1}{2}(P_{out}(0) + P_{out}(1))$$  \hspace{1cm} (3.9)

For the carry bit $C_o$, the accuracy is that of the deterministic input AND gate that was shown in Figure 3.7. As its inputs are deterministic, it is primarily dependent on the switching probability $P_s(t)$. The accuracy of the carry is then calculated as

$$Accuracy(C_o) = \frac{1}{4}(1 + 4P_s(t) - P_s^2(t))$$  \hspace{1cm} (3.10)

The theoretical analysis is also verified by simulating the gates based on the stochastic memristor. Figure 3.9 shows the accuracy for the sum and carry bit in the theoretical and simulation forms. As demonstrated, the performance of the sum bit starts off at 50%, seeing as two cases are present for the high bit and the AND gate produces an accurate ‘1’. The accuracy of the carry bit starts with a lower performance than the sum, but it catches up at a switching probability of 0.3 and then continues to have better performance.
3.3.2 Full Adder

In the half adder, the XOR gate is initiated with deterministic inputs. As a result, only the last AND gate requires analysis with probabilistic inputs. However, when the arithmetic block is extended to include the carry-in bit $C_{in}$, an additional XOR operation is required to calculate the sum $S_1$. In contrast, the carry requires two AND operations and an OR operation to obtain the final carry-out bit $C_1$. The schematic and the truth tables for the full adder are shown in Figures 3.10a and 3.10b respectively. The equations for the output bits $S_1$ and $C_1$ are specified as

$$S_1 = (p \ (XOR) \ q) \ XOR \ C_{in}$$

$$C_1 = (p \ (AND) \ q) \ OR \ ((p \ (XOR) \ q) \ AND \ C_{in})$$

(3.11)

The calculation of the output bits passes through several stages of gates. The deterministic input assumption thus cannot be used, as the cascade induces the notion of variability or probabilistic outcomes. Similar to the gate analysis provided in the previous section, the output probabilities and corresponding accuracy for each logic gate are formulated but based on non-deterministic input parameters. These equations were tested with the output accuracy of the sum and carry bits. Figures 3.11a

![Figure 3.10](image)

**Figure 3.10:** (a) The full adder schematic based on the logic operators. (b) The truth table showing the different entries for the high and low bits for the sum $S_1$ and carry $C_1$ bits.
Figure 3.11: (a) The accuracy of the 1−bit full adder for the sum bit ($S_1$) for theoretical and simulation verification (b) The accuracy of the carry output ($C_1$) with respect to the memristor switching probability.

and 3.11b show the results of this analytical approach for $S_1$ and $C_1$. The simulation included the theoretical formulation based on the probabilities of the inputs and its consequent relation and cascades. System-level simulations of the memristor element-based structure are also conducted. A 100-simulation run for the full architecture is applied, and the resultant output is noted in regard to the expected output behaviour for all entries in the truth table. The simulations showed a high level of matching with the analytical equations.

Moreover, an intriguing feature surfaces when the behaviours of the half and full adders are compared from the perspective of the sum and the carry bits. With respect to the sum, as expected, the performance shows a small degradation for the full adder in comparison to the half adder. This is because while both bits $S_o$ and $S_1$ are outputs of an XOR gate, the former is with deterministic inputs and the latter is with probabilistic inputs. In this case, the stochasticity thus has a negative impact on performance. However, it is interesting that when the carry bits $C_o$ and $C_1$ are compared, performance is enhanced by the added probabilistic operation. The accuracy of the full adder is better up to around a 0.3 probability of switching, after which the accuracy of the half adder carry starts to show faster improvements.
Figure 3.12: The accuracy of the half and full adders for the sum and carry bit. The carry $C_1$ starts off with better performance at low switching probabilities and shows faster improvement at an inflection point of $P_s(t) = 0.3$ of the switching probability.

Figure 3.12 illustrates the plots for the output bits for each arithmetic stage. This comparison highlights the final gate’s effect on the precision of the arithmetic operation. Moreover, final outputs could have better results than intermediary outputs, depending on the sequence of internal gates used for a particular operation. This probabilistic feature is boosted by the variability inherent in the memristor and the gates accordingly.

### 3.3.3 N-bit Adder

An N-bit adder is analysed to obtain a general overview of the system performance under several cascades. A ripple-carry adder structure is adopted to observe the probabilistic input’s effect on the intermediary and final outputs of the arithmetic block. Figure 3.13 shows a high-level block diagram of the adder structure. It builds on blocks of 1-bit full adders (FAs) that are cascaded together with the carry out of one block propagating as the carry in to the successor block. Within this structure, the output sum bits ($S_{n-1}...S_0$) are calculated independently from each other; however, they are directly affected by the carry from the previous FA block. As depicted in Figure 3.13b, the FA block can then be further decomposed into sum generation and
Figure 3.13: (a) The block diagram of the Ripple Carry Adder (RCA) with $n$—full adder blocks (FA) and the carry propagating from one FA to the successor FA block. (b) The internal sum and carry generation blocks for each full adder.

carry generation sub-blocks. The sum bits are directly mapped to the output, whereas the carry bits are propagated to the consecutive sum and carry generation blocks. To that end, the carry bit has a crucial effect on the adder’s overall performance. Moreover, optimizing this carry generation block would lead to a more accurate result.

### 3.3.3.1 Structures

Three FA structures are explored in this section. The sum generation block in all three structures is the calculated as

$$Sum = [(A.B).[(A + B).C_{in}].[(A.B).[A + B] + C_{in}]]$$

(3.12)

On the other hand, each carry generation block investigated has a different logic formulation and connection. Figure 3.14 shows the structures considered with the
Figure 3.14: 1-bit full adder with different carry generation blocks (a) Separate Carry and sum blocks. (b) Shared structures with the sum (c) NAND gates structure

following $C_{out_i}$ bits

\[
C_{out_i} = [(A.C_{in}) + (A.B)] + [B.C_{in}] \tag{3.13}
\]

\[
C_{out_{II}} = [(A \oplus B).C_{in}] + [A.B] \tag{3.14}
\]

\[
C_{out_{III}} = [(A \oplus B).C_{in}].[A.B] \tag{3.15}
\]

In structure I, the sum and carry generation blocks are completely separate. In structure II, the first half-adder sum bit is shared with the output sum $S_1$ as well as the carry-out bit $C_1$. Structure III, which is a slight variation of the second structure, uses only NAND gates for the carry generation.

### 3.3.3.2 Comparison Metrics

In each of the structures considered, we aim to investigate the impact of the gates and the underlying structure on the output behaviour. The metrics of error distance (ED), mean error distance (MED), and mean relative error distance (MRED) are utilized to compare the overall output behaviour. As the gates behave in a probabilistic manner, bit flips are more likely to occur. As such errors are introduced into the final sum output. The ED is used to quantify the levels of approximation and the distance
away from the actual expected sum. The error distance is determined as follows:

\[ ED = |R_{\text{Approximate}} - R_{\text{Accurate}}| \]  \hspace{1cm} (3.16)

where \( R \) corresponds to the adder’s overall output. The ED represents the absolute value of the difference between the approximate result obtained and the expected accurate result \[105\]. The MED is an average of the absolute ED over a set of inputs \[106\]; it incorporates the probability of a particular sum occurring, as it is the expectation of the EDs over the output sums. The MED is determined as follows:

\[ MED = \sum ED_i . P(ED_i) \]  \hspace{1cm} (3.17)

The relative error distance (RED) takes into consideration the actual sum value; as such, it is calculated as follows

\[ RED = \frac{ED}{R} \]  \hspace{1cm} (3.18)

The MRED is the measure of the average RED over all of the samples. With \( N \) being the number of samples taken for simulation, the MRED is thus calculated as

\[ MRED = \frac{RED}{N} \]  \hspace{1cm} (3.19)

Simulations are performed with the stochastic memristor as the basis for the logic operators. A set of 100 random inputs is simulated in Matlab for the different possibilities of the sums and the entire range of the memristor’s switching probability is swept over. The three structures for the carry generation block show close behaviour. Structures II and III are almost identical, as they undergo the same sequence of logic operations and the OR, AND, and NAND also have similar probabilistic behaviour on average (as shown in Table 3.2). The slight shift of the parallel AND structure
Table 3.2: The comparison among the three carry generation structures in terms of the MED and the MRED. In addition to varying the number of bits for the adders.

<table>
<thead>
<tr>
<th>Structures</th>
<th>( P_s )</th>
<th>2b</th>
<th>4b</th>
<th>8b</th>
<th>2b</th>
<th>4b</th>
<th>8b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structure I</td>
<td>0.16</td>
<td>1.89</td>
<td>7.13</td>
<td>112.22</td>
<td>0.86</td>
<td>0.87</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>1.39</td>
<td>5.65</td>
<td>90.96</td>
<td>0.58</td>
<td>0.57</td>
<td>0.53</td>
</tr>
<tr>
<td>Structure II</td>
<td>0.16</td>
<td>1.95</td>
<td>7.23</td>
<td>116.21</td>
<td>0.91</td>
<td>0.86</td>
<td>0.76</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>1.15</td>
<td>5.23</td>
<td>85.18</td>
<td>0.47</td>
<td>0.52</td>
<td>0.47</td>
</tr>
<tr>
<td>Structure III</td>
<td>0.16</td>
<td>1.93</td>
<td>7.23</td>
<td>116.62</td>
<td>0.89</td>
<td>0.85</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>1.16</td>
<td>5.24</td>
<td>84.12</td>
<td>0.47</td>
<td>0.48</td>
<td>0.49</td>
</tr>
</tbody>
</table>

from the other two structure is due to the early processing of the carry-in bit. It is included in four logic operators, as compared to being included in two operations in the other structures. The bit’s accuracy hence degrades and propagates poorly to the consecutive blocks, which in turn affects the accuracy of both the higher sum and carry bits.

Aside from the adaptability and configurable accuracy features, the main advantage of approximate computing lies in the savings it provides. A design space is available for the energy point and delay required for a particular application at hand. Figure 3.15a shows the accuracy with respect to the applied input voltage for the SET/RESET operation. Depending on the applied time period, a 100% precision

Figure 3.15: (a) The tradeoffs available between the input voltage and the attainable accuracy of the full adder. The accuracy level is controllable with the applied input voltage and time period of application. (b) The time voltage relation for particular levels of accuracy levels required.
of operation is achievable at a lower nominal voltage at the expense of longer delay. This voltage-time relationship is further illustrated in Figure 3.15b. Fixed accuracy levels for a range of voltages and times allow for diverse options in the design space allocation.

This study of probabilistic logic behaviour and its consequent impact on arithmetic operations demonstrates the preliminary advantages of embracing the underlying memristor element’s stochasticity. However, to have a tangible quantification of the performance enhancement and possible savings, we tested a high-level application based on the underlying probabilistic logic.

3.4 Error Resilient Applications

Smart devices that are capable of real data sensing, image processing, and communication are the main building blocks of the Internet of Things (IoT). The key enablers for realizing these functionalities within a single miniaturized platform are the feasibility of integration and novelties in the underlying computing paradigms. Moreover, the inherent resiliency in the nature of IoT applications paves the way for incorporating techniques such as approximate computing into the realm of processing. Image compression plays a major role in scenarios in which energy-constrained smart nodes are to transmit captured images over the network [36]. Compression schemes act to reduce the number of bits used to represent the pixels within an image by removing spectral redundancies. Two-point discrete Fourier transform (DFT) is a prominent technique that builds on simple addition and subtraction operations. Levels of error in the arithmetic operations are tolerated within the DFT due to the relative perceptive measures used to quantify the output characteristics [37]. In other words, the error’s impact on the compressed image is tolerated without affecting quality due to the limitations of human perception. Extending the probabilistic logic into the corresponding arithmetic blocks thus creates a further space for energy savings and
Figure 3.16: The sequence of operation for the compression of the image using DFT with probabilistic addition and error-free reconstruction of the image with the inverse DFT (IDFT).

3.4.1 2-Point Discrete Fourier Transform

In relation to an image compression application, the principles of the DFT rely on having two arithmetic operations: addition and subtraction. With $x[i]$ and $x[i + 1]$ being two consecutive pixels, the DFT output is calculated using butterfly operations in which the output values $y[i]$ and $y[i + 1]$ are a direct function of the input values:

\[
y[i] = x[i] + x[i + 1]
\]

\[
y[i + 1] = x[i] - x[i + 1]
\]

The subtraction is transformed into addition by using the 2’s complement of the corresponding subtrahend. Figure 3.16 shows a block diagram of the sequence of operations performed; it starts with the original image and ends with reconstruction after compression. A 100x100 pixel grayscale image is used for the compression application. The input pixels are fed sequentially into the 8-bit probabilistic ripple carry adder (RCA) executing the DFT. The RCA’s building block is based on a 1-bit FA built from memristive logic gates [44]. The DFT’s addition and subtraction are performed for several operating voltages to assess the output characteristics and the probabilistic behaviour of the underlying operations.
3.4.2 Image Reconstruction

Reconstruction of the images is achieved using an error-free two-point inverse DFT (IDFT). Moreover, the quality of the compression is characterized through two quality metrics.

3.4.2.1 Peak Signal to Noise Ratio

PSNR is the measure of the noise in the output signals in comparison to the original image. It is calculated as follows:

\[ PSNR = 10 \log_{10} \left( \frac{PV^2}{MSE} \right) \]  

(3.21)

where PV stands for the peak value of the pixel. With eight bits used to represent the pixel, the PV is considered to be 255. The MSE is the mean square error between the pixel’s reference value in the original image and its value after reconstruction. Figure 3.17 shows the PSNR values for the reconstructed images under different switching probabilities \( P_s(t) \) of the memristor. As noticed, downscaling the voltage and consequently the memristor’s switching probability adds errors to...
the image reconstruction process. Nonetheless, despite the low PSNR values attained, the features within the images are still recognizable asserting by that the inherent resiliency in these applications.

### 3.4.2.2 Structural Similarity Index

SSIM is an assessment of the perceived image quality based on the quantification of the visibility of errors [107]. This measure builds on the adaptability of the human visual system in extracting structural information. Considering two image signals \( x \) and \( y \) from the original and the reconstructed image respectively, the SSIM \( S(x,y) \) comprises three components

\[
S(x, y) = f(l(x, y), c(x, y), s(x, y))
\]

\[
S(x, y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}
\]

where \( l(x, y) \) is the luminance comparison, \( c(x, y) \) is the contrast comparison, \( s(x, y) \) is the structural comparison, \( \mu \) and \( \sigma \) are the mean and the standard deviation, and \( C_1 \) and \( C_2 \) are added constants to avoid instability. The SSIM is measured for the reconstructed images over different switching probabilities and corresponding operating voltages. The higher the value of the SSIM index, the closer the reconstructed image
Table 3.3: Comparison of alternative approximate adders and applications in image compression.

<table>
<thead>
<tr>
<th>Adder</th>
<th>Compression</th>
<th>PSNR (dB)</th>
<th>Technology</th>
<th>Area (µm²)</th>
<th>Power Saving (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truncation (8 LSB) [108]</td>
<td>DCT</td>
<td>19.04</td>
<td>Transistor 90nm</td>
<td>20.4</td>
<td>58.3</td>
</tr>
<tr>
<td>Approximate Adder 1 [108] (sum error (8 LSB))</td>
<td>DCT</td>
<td>30.02</td>
<td>Transistor 90nm</td>
<td>29.31</td>
<td>42.4</td>
</tr>
<tr>
<td>Approximate Adder 2 [108] (carry error (8 LSB))</td>
<td>DCT</td>
<td>28.3</td>
<td>Transistor 90nm</td>
<td>25.5</td>
<td>36.5</td>
</tr>
<tr>
<td>Approximate Adder 3 [108] (sum and carry error (8 LSB))</td>
<td>DCT</td>
<td>22.7</td>
<td>Transistor 90nm</td>
<td>22.56</td>
<td>52.1</td>
</tr>
<tr>
<td>Approximate Adder with [44] stochastic memristors (8 Bits)</td>
<td>DFT</td>
<td>&gt;30</td>
<td>Memristor ~10nm</td>
<td>0.08</td>
<td>~60</td>
</tr>
</tbody>
</table>

is to the original one until reaching a value of 1 that corresponds to complete similarity. As depicted in Figure 3.17 increasing similarity is achieved with larger input voltage which correlates with the values achieved with the PSNR quality metric.

Further quantification in terms of the power saving is shown in Figure 3.18 that plots the PSNR values over the range of $P_s(t)$ and the power saving as well. In image processing applications, the quality of the image is considered good if the PSNR value is greater than 30dB. As such, power savings from 25% up to 60% are feasible, depending on the operating frequency, while maintaining the quality of the images at hand.

### 3.4.3 Techniques Comparison

In comparison to alternative techniques that employ approximate adders for image compression application [108], our proposed stochastic memristor-based adder outperforms them in the PSNR, area, and power saving attainable. This performance is mainly due to the small feature size of the memristor, and its possible incorporation into an area efficient crossbar structure. Moreover, the probabilistic nature of the
memristor behavior and its corresponding configurable operation with respect to the input voltage allows for ample space in the power saving while preserving high quality of the image processing application. Table 3.3 shows the performance of different approximate adders used in image compression applications.

3.5 Summary

In this chapter, we presented a cornerstone study in the approximate computing domain using memristive elements. The logic gates’ performance in response to the stochastic operation of their underlying components was analysed. The incorporation of stochastic gates into an arithmetic block resulted in intriguing features of adaptability and reconfiguration in the performance metrics. Theoretical and memristor-based simulations were used to verify the operation, and a set of characteristics for the approximate computing approach was provided. Image processing applications offer a robust medium that benefits greatly from the probabilistic nature of the adopted unconventional computing paradigms.
Chapter 4

Neuromorphic Architectures

With the increasing complexity of environmental stimuli and collective demands on data and processing potentials, the von Neumann architectures are starting to fall short in providing the levels of cognition required for the next generation of computing systems. A shift towards brain-inspired designs has hence surfaced with several grand-scale projects such as the Human Brain Project [109, 110] in Europe and the Brain Initiative in the United States [111]. As depicted in Figure 4.1, machine complexity increases exponentially with added input stimuli and will eventually be outperformed by unconventional systems that build on the emulation of human brain functionalities, particularly neuromorphic system architectures [112].

![Figure 4.1: The scaling of the machine complexity with the increasing complexity of the environmental stimuli](image)

Figure 4.1: The scaling of the machine complexity with the increasing complexity of the environmental stimuli [113].
Nonetheless, in order for machine intelligence to reach the level of human cognition, convergence in advances related to deep learning methods and computing resources is required [113]. This advancement should map to the connectivity and energy efficiency levels of computing and communication in the brain that collide and help data to adaptively reduce algorithmic complexity. This gives rise to the field of neuromorphic engineering with brain-like circuits.

Neuromorphic circuits are hardware implementations that tend to convey the fundamental operational principles of neural systems. Multi-levels of nervous system analysis exist, starting with a thorough ionic transport and going through layers of neurons and synapses until the brain is abstracted as a complete system. As

Figure 4.2: The nervous system multiscale analysis with its corresponding neuromorphic engineering mapping into silicon microsystems [113].
depicted in Figure 4.2, at each of these levels neuromorphic system engineering aims to synthesize underlying functions via efficient silicon microsystems [113]. In a simplified implementation process, the principal components are comprised of neurons and synapses that are designed to capture the fundamental spiking behaviour; the neuron serves as the processing and spiking unit [114], whereas the synapses serve as the communication channel and weight-holding component [115]. Further to classical implementation modes, the memristor has been integrated into these neuromorphic structures in diverse forms [116, 117]. In this chapter, we:

1. Introduce a novel stochastic neuron model that builds on the inherent variability of memristor switching;

2. Demonstrate that the new neuron circuit acts as both a source of noise that aids in the learning process and an efficient replacement for alternative costly noise injection techniques;

3. Investigate stochastic bi-stable synapses based on a stochastic memristive cross-bar;

4. Build system- and circuit-level simulators for a winner-take-all stochastic neural network and provide the design characteristics; and

5. Apply and test an image recognition application with benchmark MNIST dataset. The accuracy rates attained support the added advantages of having noise in the network in terms of design simplicity and learning.

4.1 Probabilistic Neural Networks

An interesting insight related to brain functionality is that the level of human cognition arises from imprecise, unreliable components within the brain, which strongly
suggests the probabilistic and statistical optimality of perception, learning, and decision-making processes [118, 119]. Hardware emulations of these non-deterministic architectures incur extra circuitry to be able to map such probabilistic behaviour [120]. In contrast, alongside adhering to energy and scaling requirements, the memristor has an innate feature of stochastic switching between its low and high resistance states. The stochastic memristor thus allows for the inherent inclusion of a variable operation into the underlying neural building blocks. The following sections elaborate on the proposed neuronal and synaptic design in addition to the system level testing within a probabilistic neural network activity.

4.1.1 Neuron Models

The neuron is the main computing element used for encoding and processing information. The hardware models employed to emulate this operation can be split into two main approaches [121]. The first technique, which is referred to as the Hodgkin-Huxley model [122, 123] adopts a detailed mapping of the physiological conductance-based model with ionic interactions and the dynamics of the internal current. The second technique, which is known as the integrate-and-fire model, entails a high-level abstraction that deals with the basic properties of the accumulation of charges and producing a spike output [124, 125].

4.1.1.1 Hodgkin-Huxley Model

This classical description of the neuron operation accounts for charge conservation across the membrane with capacitance $C$ of the neuronal soma under input current excitation. The current in the soma is then calculated as follows

$$C \frac{du}{dt} = - \sum_k I_k + I(t)$$

(4.1)

where $u$ is the membrane potential, $I(t)$ is the external current, and $\sum_k I_k$ is the sum.
of the ionic channels of Sodium (Na), Potassium (K), and leakage (L) current across the membrane [126, 127]. Figure 4.3 shows the schematic for the model with the corresponding conductance of the ionic channels and the input current excitation.

![Figure 4.3: Schematic of the Hodgkin-Huxley model for the neuronal soma charge conservation and input excitation.](image)

The difference in the concentration of the ions between the interior and exterior of the membrane causes the neuron cell’s interior to be negatively charged and the membrane to act like a capacitor. With the injection of the input current I(t), the membrane capacitor is thus charged and the current leaks through the channels. This increase in the membrane voltage in turn causes the sodium ions to flow into the cell, increasing the voltage even further until an action potential is generated and followed by a negative overshoot that slowly decays to zero and constitutes the spike generation process.

### 4.1.1.2 Integrate and Fire Neuron Model

This model is a simple abstraction of the neuronal activity into an accumulation of charges until a threshold is reached, after which a spike is generated. The integrate-and-fire (I&F) model circuit consists of a capacitor C and resistor R in parallel and
are driven by an input current $I(t)$ \cite{128}. The concept of charge conservation dictates the following equation for the model:

$$ I(t) = \frac{du}{R} + C\frac{du}{dt} \quad (4.2) $$

where $u$ is the voltage across the capacitor or the membrane potential. Rewriting this equation in the standard form with the time constant $\tau_m = RC$ results in

$$ \tau_m \frac{du}{dt} = -u(t) + RI(t) \quad (4.3) $$

An additional equation is required to account for the threshold condition, where the firing time is specified as $t^{(f)}$, the threshold crossing is $u(t^{(f)}) = \vartheta$, and the reset potential value is $u_r$

$$ \lim_{\delta \to 0}(u(t^{(f)} + \delta)) = u_r \quad (4.4) $$

Once the threshold condition is reached, a spike is generated and the membrane potential is reset back to its original value $u_r$. The spikes are generated in a deterministic manner: after a refractory period in which the neuron does not provide any post-firing action potential, the integrating and firing actions are initiated again with the input excitation and threshold satisfaction.

### 4.1.1.3 Neuronal Stochasticity

Noise is an omnipresent feature in the biological medium due to the non-zero temperature and underlying ionic interactions \cite{121}. This noises impact on the neural system operation includes synaptic transmission failure and variable behaviour of the neuron in response to the same input current excitation. Moreover, despite this added randomness, recent studies in neuroscience support the benefits and enhanced efficiency of injecting noise in the learning and information processing \cite{129}. Techniques
used to induce stochasticity into the network either exploit the network-generated variability [130] or inject noise per neuron [131, 132]. Relying on the network-based variability is an approximation of the real case and is constrained to a certain set stability range [133]. On the other hand, with the injected noise per neuron technique, direct control of the stochastic properties of every neuron is attained. However, an added expense of area- and power-consuming hardware is incurred, which limits the networks scalability and efficiency (as depicted in Figure 4.4a).

Alternatively, we propose using the stochastic memristor as an inherent source of variability in the neuron that allows it to produce spikes stochastically. The memristor is thereby put in parallel with the original neuron circuit with its variable threshold capitalized upon to randomize the neurons firing threshold and consequently its spiking behaviour, as shown in Figure 4.4b. From the circuit perspective, with its variable threshold the memristor acts as a stochastic comparator and allows for a more area- and power-efficient implementation of the neuron. The memristor solely replaces both the random number generator for the injected noise along with the operational amplifier in earlier designs.

To verify the operation of the proposed memristor-based neuron, we build a model for the circuit and simulate it in python. A constant current of $I_{syn} = 3.1A$ is used as an input into the neuron membrane, with a simulation time step of 0.1ms; the
output current through the memristor device is recorded, as shown in Figure 4.5a. The variation in the peak values corresponds to the cycle-to-cycle variation of the resistance states of $R_{ON}$ and $R_{OFF}$ respectively. The spike generation occurs stochastically while following exponentially distributed spike-time intervals, as depicted in Figure 4.5b. The neuron circuit’s operation hence follows a Poisson distribution and demonstrates a close emulation of its biological counterparts.

### 4.1.2 Synaptic Model

Communication with the brain is based on electrical impulses and the weighted release of incoming spikes through the synapses. In hardware emulations, two modes of operation are available. The first involves an analog synapse with continuous states and change from a high to a low weight of 1 or 0, respectively [134]; the states correspond to a complete release, with a gradual change until an entire block of the spikes received. The second mode corresponds to a binary synapse, where the states are confined to either 0 or 1 [115]. The hardware implementation of neuromorphic building blocks is constituted of synapses mainly based on integrated CMOS circuitry [135].
Nonetheless, with the level of integration and power limitations imposed by the required degree of computation, novel alternatives are being considered to overcome the current bulky and energy-consuming circuit counterparts \cite{136}. One prominent device is the non-volatile redox-based resistive random access memories (RRAM), which is also known as the memristor.

4.1.2.1 Memristor-Based Synapses

The distinctive features of memory and resistance change within the memristor allow for its integration into neuromorphic platforms. In its continuous progression operation and threshold-less variants, it is shown to have the close resemblance to the behavior of a continuous synaptic element \cite{137, 138, 139}. In this mode of operation, the weight of the synaptic connection, which is coded through the memristors resistance change, holds any value between its upper and lower boundaries. In contrast, binary synapses are bounded by two states, namely ‘0’ and ‘1’, to induce either a strong or weak post-synaptic response (depending on the state of the memristor). This two-state operation of the synapses is also emulated by the threshold-based memristors. These devices are confined to a range under which no change in the resistance is observed. Once the input threshold is exceeded, the device switches to the boundary values of $R_{\text{ON}}$ or $R_{\text{OFF}}$ which represents the ‘1’ or ‘0’ value of the synapse.

4.1.2.2 Stochastic Synapses

An alternative approach to having stochasticity induced into the neural network is using stochastic synapses. In such synapses, the communication channels between the neurons do not behave in a deterministic manner. The incorporation of stochastic synapses is based on stochastic memristors in a crossbar structure and builds on the random switching between its two binary values. The synaptic weight is thus variably set to either an ON or OFF resistance state according to the corresponding value of
Figure 4.6: Memristor-based stochastic synaptic crossbar. The inset image shows the variability of switching of the memristor depending on the applied voltage level.

the memristor. Figure 4.6 shows the memristor crossbar with stochastic memristive elements. This mode of stochastic synapses is induced using novel RAM technologies operating in a low-current, low-voltage regime. Devices reported in the literature include conductive-bridge RAMs (CBRAMs) [61], Spin Transfer Torque RAMs (STT-RAM) [140], and RRAMs as well [91]. Nonetheless, the synaptic behaviour simulation is approached either from a system-level perspective with externally supplied weights and updates or as a sum of parallel binary stochastic memristors to account for the analog synaptic weight update [90]. Furthermore, simulation models are physically specific to the dynamics of a particular fabricated device.

In [49] we have proposed a simple generic stochastic model [34] for the memristor that is applied along with SPICE-based simulations for synaptic memristor crossbar interfacing integrate-and-fire neurons. Transient simulations allow for the application of temporal stochasticity and a realistic emulation of the synaptic behaviour. The result is a general neuromorphic platform that accounts for circuit variables and tests the corresponding synaptic plasticity and learning attained with stochastic memristor crossbar arrays.
4.2 Memristive Neural Networks

In its deterministic form, a memristor integrated within the neural network platforms has shown similar performance to conventional CMOS architectures, with additional area and power savings. Adding the stochastic feature provides a further venue for design saving and resilience to the inevitable hardware variations. Stochastic memristive networks are investigated using two distinct approaches: neuronal stochasticity with deterministic analog synapses and synaptic stochasticity with integrate-and-fire neurons.

4.2.1 Winner-Take-All Network

The winner-take-all (WTA) network is mainly composed of three layers: the input-encoding neurons, the WTA layer for the output neurons, and an inhibitory layer that is fully connected to all of the output neurons. For the binary image recognition application, every pixel is encoded with two stochastically firing neurons in the input layer. The possible values (1 and 0) of the pixel $X_i$ are mapped onto high- or low-spiking rate of the input neuron while the remaining neurons in the input layer remain silent. The output layer, which corresponds to the middle layer in Figure 4.7, consists

![Figure 4.7: Neural Network with I&F input neurons for encoding the images, and competing I&F output neurons for the winning-take-all operation](image-url)
Figure 4.8: (a) The initial state of the synapses and neurons within the network prior to any learning. (b) The learning is shown with the synaptic adaptation to the input patterns, and the neurons’ exclusive reception to the set fed [33].

of competing neurons; once any neuron spikes, the complete layer is inhibited through the third inhibiting layer. This last layer corresponds to a single neuron with a threshold low enough to send signals back to the WTA layer to inhibit the spiking of the non-winning neurons. The membrane potential of neurons $Z_k$ from the WTA layer can be written as a sum of the synaptic input:

$$u_k(t) = u_{k_0} + \sum_{i=1}^{N} w_{k_i}y_i(t)$$

(4.5)

4.2.2 WTA with Stochastic Memristive Neurons

The simulation platform is built using Brain Simulator, with input datasets taken from MNIST data for handwritten digits from 0 to 4. Each pattern is encoded for a period of 40ms. A total of approximately 1000 samples per class are used to train the network in an unsupervised manner. Once the training is concluded, the network’s learning and recognition rate is tested. Patterns are fed to the network in a random manner, with the network’s ability to distinguish the correct class being noted. The overall accuracy of the classification is calculated as follows:
\[ \text{Accuracy} = \frac{|\{x \in \text{test}\}{\text{classified}}\{x\} = \text{Trained}\{x\}|}{N_{\text{test}}} \] (4.6)

where \(N_{\text{test}}\) corresponds to the number of test images used for the classification. Prior to any learning, the synaptic connections and neurons are randomly set; the neurons do not spike exclusively in any specific pattern, as shown in Figure 4.8a. However, after training the synapses adapt to the input pattern and the corresponding neurons become exclusively receptive to the particular pattern fed, as depicted in Figure 4.8.

### 4.2.3 WTA with Stochastic Synapses

Using the memristor within a crossbar structure while incorporating the temporal stochasticity provides an interconnected array of input and output neurons. The interactions between the pre- and post-synaptic neurons impose voltage levels across the memristors, whose states are updated in a non-deterministic manner. Adding the stochastic feature to the binary synapses makes them behave in a probabilistic way by allowing the neuronal spikes to pass or induce a weak response as per the memristor state. This emulation of the noisy environment within the brain enhances the learning process for the neural network [141, 142] at a lower modelling complexity expense and higher power efficiency. In the following section, a SPICE-based network simulator is illustrated for a recognition application. Input encoding, training, and detection results are also elaborated.

#### 4.2.3.1 Network Dynamics

The memristor crossbar mediates the excitatory synapses between the input and output layers. Figure 4.9 illustrates the neural network and its components. For a \(N \times N\) image, \(N^2\) input neurons are required to encode the pattern. Black and white pixels are encoded using constant voltage pulses of +0.6V and -0.6V respectively for a duration of 200\(\mu s\). The memristors are randomly initialized to either ON or
OFF state (10kΩ or 10MΩ). The output neurons sum the individual currents from the memristors along the column it is connected to \( V = \sum I_i w_i \). Once the potential across one of the neuron membrane reaches its firing threshold, the neuron fires and thereby generates a post-synaptic pulse. The pulse consists of a high voltage (-3.6V for 25µs) followed by an opposite +3.6V for another 25µs. The same neuron also generates a reset pulse through its inhibitory synapses to reset the other neurons in the output layer and suppress any firing.

The difference between the pre- and post-synaptic pulses puts the memristor under a weak programming condition with minimum and maximum voltage levels reaching +4.2V for an ON pixel and -4.2V for an OFF pixel. A probabilistic set/reset of the corresponding memristor synapse is thus allowed. A potential depression or potentiation of the respective synapse is a resultant reinforcement of the output neurons responsiveness to a particular pattern. Figure 4.10 shows the respective input and output pulses and the potential across the memristor due to the pre- and post-synaptic pulses.

Figure 4.9: A Winner-take-All (WTA) network with the synaptic memristors crossbar interfacing integrate-and-fire neurons with the RE-read enables and WE-write enables to switch between the reading and writing phases on the memristor. \( R_{\text{ON}} = 10k\Omega, R_{\text{OFF}} = 10M\Omega, \) crossbar resistance \( R_{\text{cb}} = 10\Omega \)
Figure 4.10: (a) The pulses (+/-0.6V input) applied at different stages of the training. (b) Post synaptic pulses. (c) A non-deterministic depression/potentiation reflects the stochastic binary learning due to the low voltage regime with max potential of +/-4.2V at the memristor terminals.

4.2.3.2 Iterative Training and Recognition

Building on unsupervised learning conditions, the network needs to be trained freely on the image set to be identified. To that end, the following training phase is applied: all images are supplied to the network one at a time, with the weights being adjusted according to the neuron that spikes and reinforced to the level that a pattern is associated and identified with a particular output neuron. The SPICE simulation of the training phase includes two parts: a read phase and a subsequent write phase. In the read phase, the read enable (RE) is activated, and a small reading pulse constituted mainly of the input patterns pixels is supplied. The output file is then
Figure 4.11: (a) Read Phase: The input pattern supplied to the network in the form of pulses to the memristor crossbar \((V_m = +/−0.6V, R_{OFF} = 10M\Omega, R_{ON} = 10k\Omega, R_{cb} = 10\Omega)\). The adaptation of a particular neuron to this specific image pattern results in its firing once its membrane potential exceeds the firing threshold \(V_t\). In this case, neuron 2 was the first one to fire and it resets all the remaining output neurons. (b) Write Phase: The temporal variability in the switching of the synaptic memristors under the excitation of the post synaptic pulses. A subset of the 100 memristors stochastically switch state at different times.

parsed for the spiking neuron. Within the crossbar, the synaptic memristors thus produce a current based on their internal resistance value and the net potential across them. The output neurons integrate their input current from the memristors along their column connection until one of them spikes and triggers the end of this phase. Figure 4.11a shows a sample output of the simulations corresponding to the neurons membrane potential and the resetting once the winning neuron reaches its threshold voltage \(V_t\).

Prior to the start of the write phase, the index of the fired neuron \(i\) is passed along with the memristor states to the next phase. The write phase is then initiated, with the \(i\)th column supplying a post-synaptic pulse while the other neurons are silent or set to 0. The row input in both the read and write phases is set to the pulse values \((+/−0.6V)\) of the particular image to be decoded. In the write phase, the memristor switches stochastically to either \(R_{ON}\) or \(R_{OFF}\) depending on the voltage across its terminal and the duration of application. As depicted in Figure 4.11b, a subset of the 100 memristors change state under the excitation of the post-synaptic pulse,
**Algorithm 2: Iterative SPICE Neural Network Simulator**

1. Initialize the memristors randomly with ON or OFF states
2. Training Phase;
   
   for *every image set* do
   a. Read phase is active with the input neurons generating pulses per the input image;
   b. The spice file is generated and simulated with the firing output neuron index, and memristor states recorded;
   c. The write phase is active with the corresponding configuration for the input and output neuron;
   d. The spice file is simulated with the updated memristor states recorded;
   e. The read phase is activated again to train the network on the same image;
   f. The loop is repeated again for several input images.

3. Recognition phase;
   
   for *every input image* do
   A read phase with the memristor states supplied from the last step of the training phase and the firing neuron index is recorded to test the learning of the network.

thereby adhering to the temporal variability required to enhance system performance. At the end of the transient simulation, the output file is parsed for the memristor resistance value and supplied as inputs to the read phase to re-start the training loop. The overall steps are summarized in the following algorithm.

For the simulation of the above algorithm for training and recognition, a python script is used to trigger the SPICE simulations and pass the states and input parameters on. A complete SPICE file (including the memristor crossbar and input and output neurons) is generated for each phase. Further circuit variables are also incorporated into the code, such as the crossbar resistance \( R_{cb} = 10\Omega \) within the memristor crossbar. This iterative setting allows for the simulation of the array completely at the circuit level (including the updating of the memristor state), which provides a realistic image of circuit behaviour. To test the network’s functionality, a simple set of MNIST images (each of 28x28) is used to train the network and mea-
Figure 4.12: The memristor resistance states adapt to the input patterns and learn based on the potentiation/depression that triggers the switching event stochastically.

4.2.4 Performance Analysis

To test network performance in different settings, a networks training phase involves randomly presenting digit patterns for the training dataset for digits 0 to 4. In neuronal stochasticity, the stochastic memristive neurons are set as the input and output layers. The network performance shows high accuracy ranges and robustness to several characteristic metrics variation [33]. The spike timing-dependent plasticity (STDP) learning rule is applied in the simulations, which allows the synapses to adapt to the input images in a more analog manner.

Alternatively, in synaptic stochasticity, a stochastic learning rule is applied with binary synaptic weights that are also confined and abstracted into two levels only. Nonetheless, the binary abstraction does not affect the accuracy of operation. On the
<table>
<thead>
<tr>
<th>Model</th>
<th>Output Neuron Layer</th>
<th>Synaptic Connections</th>
<th>Area</th>
<th>Learning Rule</th>
<th>Accuracy Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Layer deterministic network [143]</td>
<td>400</td>
<td>313,600</td>
<td>Medium</td>
<td>Regular STDP</td>
<td>93.5</td>
</tr>
<tr>
<td>Stochastic with injected noise [144]</td>
<td>128</td>
<td>200,704</td>
<td>Large</td>
<td>Regular STDP</td>
<td>80.1</td>
</tr>
<tr>
<td>Stochastic memristive neuron [33]</td>
<td>128</td>
<td>200,704</td>
<td>Medium</td>
<td>Regular STDP</td>
<td>78.4</td>
</tr>
<tr>
<td>Two layer network [143]</td>
<td>50</td>
<td>39,200</td>
<td>Small</td>
<td>Pulsed STDP</td>
<td>81.1</td>
</tr>
<tr>
<td>Two layer network with synaptic variability [144]</td>
<td>50</td>
<td>39,200</td>
<td>Medium</td>
<td>Pulsed STDP</td>
<td>68.7</td>
</tr>
<tr>
<td>Stochastic memristive synapses [50]</td>
<td>32</td>
<td>25,088</td>
<td>Small</td>
<td>Pulsed STDP</td>
<td>63.04</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison of the stochastic neural network applications with neuronal and synaptic stochasticity

contrary, in simulations for networks of 16 and 32 output neurons, the accuracy also ranged from 60% to 64% respectively for the MNIST dataset which is very similar to the behaviour of the neuronal model statistics. The close recognition rates attained with the neuronal and synaptic approaches demonstrate the interchangeability of stochasticity induction and adaptation to the supplied input patterns in terms of learning potential.

It is also noteworthy that the network’s performance is strongly dependent on the number of neurons and corresponding synaptic connections. Table 4.1 shows the comparison among different implementations of neural networks [143], [144]. Studies with 10 to 50 neurons and deterministic analog synapses could only achieve rates between 50% and 75% [145]. However, the potential to reach recognition rates above 90% [146] requires at least 300 neurons and 235,200 synapses. Furthermore, in terms of the stochasticity emulation, from the area and size perspective the stochastic synaptic crossbar is more area efficient as a single device can stand for a complete synaptic CMOS circuit; this offers orders of magnitude in area savings. Moreover, the inherent stochasticity results in the level of voltage applied being much lower than in the deterministic case and having many more synapses than the neuron achieves additional power savings.
4.3 Summary

This chapter investigated neuromorphic architectures mainly inspired by brain structure and unreliable noisy elements. The proposed approach capitalizes on the memristors variable switching and the threshold-based operation, which allow the memristor to be incorporated into a novel stochastic neuronal circuit and stochastic synaptic crossbar. The proposed design is akin to the firing principles of biologically stochastic neurons as well as the variability of the binary synapses. Further design verification is applied by simulating a probabilistic neural network using two approaches: 1) utilizing stochastic memristive neurons as the main building blocks for a digit recognition application, and 2) employing a stochastic neuronal network with non-deterministic binary synapses at the circuit and system levels. High learning rates and adaptation to the input patterns are achieved with further area and power consumption savings for the overall system.
Chapter 5

Conclusion & Future Work

5.1 Conclusion

In this work we investigated an alternative approach to electronic design with emerging memristive elements. We adopted the inherent variability in the memristor’s operation as a performance booster and feature to introduce novel computing systems. Moreover, we developed and verified a statistical model to capture the device characteristics and abstracted it to allow for its incorporation into circuit- and system-level architectures and applications. From the processing perspective, we addressed approximate computing with unreliable memristive components. We also established the probabilistic analysis and illustrated the impact of the corresponding arithmetic and system-level applications. High levels of both configurability and space for saving as well as further efficiency were attained with compromises among the different design metrics. On the other hand, with a complete change in the architecture and the means to attain the computations, we considered beyond von Neumann systems with neuromorphic structures. We introduced a novel stochastic neuron design that we incorporated into a probabilistic neural network to verify its operation and quantify its performance against deterministic versions within an MNIST image recognition application. Recognition rates attained were in accordance with alternative approaches but at lower area and design costs. An alternative technique for inducing the stochasticity in the network was also established by building on the variability of bi-stable synapses and its mapping onto a stochastic memristor crossbar. Circuit and system
level simulators were established to test the different models, structures, and data inputs. The simulator provided a realistic platform for neural network simulations with a simple stochastic memristor model capable of achieving STDP; it also showed levels of learning and neuronal adaptability at reduced voltages and higher power efficiency. Robustness to device variation and interchangeability between the neuronal and synaptic stochasticity were attained with improvements in the system design, which represents a step towards achieving human-level machine intelligence.

5.2 Future Work

In the future, we will further investigate alternative approaches that capitalize on the variability of the underlying electronic components and the design advantages provided. We will focus on two main techniques: one that addresses processing solely using the stochastic computing approach and one that employs the synaptic sampling paradigm to consider the complete architecture.

5.2.1 Stochastic Computing

The hardware used for computationally intensive and iterative applications (such as image and signal processing, communication coding, and informatics) is complex and consumes a large amount of energy. However, as the error tolerance feature is inherent in these operations, alternative approaches could be applied to cope with the current stringent design requirements. Adding emerging memory technologies such as ReRAM offers further innovative designs that build on the memristor’s intrinsic stochastic operation to allow for more efficient structures while abiding by all of the added constraints.
5.2.1.1 Operation Principles

In contrast to classical logic operations, the concept of stochastic computing builds on having random bit streams as inputs to digital logic. That is, the binary numbers are converted to probabilities through sequences of bits in which the number of ones over the total size of the sequence corresponds to the probability of the input. Figure 5.1 shows a block diagram of a multiplication operation using an AND gate and bit stream probabilities. As depicted, with an input sequence of eight bits and with four bits set to ‘1’, both inputs have a probability of $P = \frac{1}{2}$. However, the output sequence will only have two bits set to ‘1’ and thus correspond to an output probability of $P = \frac{1}{4}$. The longer the bits sequence, the better the precision of operation and closer the output probability to the expected value.

![Diagram of AND gate for multiplication](image)

**Figure 5.1:** AND gate to perform multiplication operation based on the input stream bits and corresponding probabilities.

With its variable switching and probabilistic behaviour, the memristor has been used in this mode of computation to generate the random bit sequence required for the native stochastic computing [147, 148]. Aside from the gains achieved in energy efficiency, randomness, and the decorrelation features needed to boost performance, using the memristor in its stochastic form abolishes the extra design efforts and correction schemes that are required when the memristors deterministic form is employed. Nonetheless, the stochasticity is only used in the bit sequence generation, whereas conventional and deterministic CMOS components are used to implement the underlying operation. As such, adding this logic variability feature within the context of stochastic computing allows for further configurability and convergence time savings.
with respect to bit sequence and operational frequency. The stochastic computing approach finds applications in diverse domains \cite{41} such as matrix operations \cite{149}, polynomial arithmetic \cite{150}, image processing \cite{151} and communication coding \cite{152}.

### 5.2.1.2 Low Density Parity Check Decoding

Low-density parity check (LDCP) codes are widely used in communication systems as a technique for sending information over noisy channels with a very high data rate \cite{153,154}. The decoding process of the received bits is based on an iterative algorithm that builds on the code-generating parity matrix. The hardware for this decoding process is complex due to the extensive computations (which are primarily repetitive product-sum operations). Utilizing conventional modes of computing requires large blocks to perform the operations. Alternatively, stochastic computing entails using very concise and basic logic gates to perform the same computations \cite{41,155}, as shown in Figure 5.2. Hence aside from using the stochastic memristor to generate stochastic bit sequences, the stochasticity is further exploited in the logic gates to allow for the configurable scaling of the applied voltage. Enhancements in the convergence and probability mapping process are thus expected to be attained in the future probabilistic designs.
5.2.2 Synaptic Sampling

Achieving human cognition levels, with the computing and communication efficiency as that of the brain, remains a challenging task. As an extension to our current research, we will consider the concept of synaptic sampling, in which a drop-out probability of the neuronal spikes is available in the system [156, 157]. The main idea lies in having synapses with analog values between ‘0’ and ‘1’, but with a probability at each instant of time to either allow the spikes to pass with the corresponding weight or completely block them. Thus in contrast to the bi-stable synaptic model and the stochasticity of the memristor, multi-state values are needed to establish this model of the synapse. From a learning and adaptation perspective, the synaptic sampling model has been shown to outperform alternative approaches for such probabilistic neural network and benchmark recognition tasks while providing more energy- and area-efficient hardware implementations.

For the hardware emulation of this model, the variability in the CMOS devices is investigated via the scaling of the transistor sizes; that is, the threshold voltage suffers from temporal variability [43, 47]. Similar to what is done in the stochastic modelling of the emerging devices, a stochastic transistor model is established based
on the threshold voltage variation. This variability combined with the memristors analog and deterministic operations are a suitable fit for the synaptic sampling model. Figure 5.3 shows a model proposed for the synapse with the dropout feature controlled by the probabilistic behaviour of the transistor switching.

Preliminary simulation results are shown in Figure 5.4, where the potentiation and depression behaviours are exhibited. The resistance increase in the depression phase suppresses the communication channel. On the other hand, the potentiation phase decreases the memristor’s resistance and increases the channel’s weight. With this proposed model, the synapses could easily be integrated within a stochastic gated crossbar structure that would allow for both high integration density and high efficiency in area and power metrics.
REFERENCES


108


APPENDICES

A Stochastic Memristor Codes

A.1 Verilog-A Model

// Stochastic Memristor Model

// Authors:
// Rawan Naous, Maruan Al-Shedivat, Khaled Nabil Salama
// King Abdullah University of Science and Technology
// Sensors Lab: sensors.kaust.edu.sa/tools
// Email: {rawan.naous, maruan.alshedivat, khaled.salama}@kaust.edu.sa

// Feel free to use/modify these codes as you see fit. Any publications (codes, papers, technical reports, ...) in which our codes (in their original or a modified format) have been used should cite the following references.

// References:

// Created: 9–Nov–2015
// run using run using spectre version 11.1.0.509.isr
// cadence version 2012.09

// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright(c)2015, R. Naous, M. Al-shedivat, & K. N Salama

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.

// Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
// * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
// * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
// * Neither the name of the King Abdullah University of Science and Technology (KAUST) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

// Copyright (c) 2015, R. Naous, M. Al–shedivat, & K. N Salama
// All rights reserved.
module memristor (tp, bn);
  inout tp, bn;
  electrical tp, bn;
  parameter integer model = 2;
  // (1: Yakopcic model - 2: Pickett model - 3: Biolek model)

  // parameters for Yakopcic memristor model
  parameter real Ap = 0.005;
  parameter real An = 0.08;
  parameter real xp = 0.2;
  parameter real xn = 0.5;
  parameter real alpha_p = 1.2;
  parameter real alpha_n = 3;
  parameter real a1 = 3.7e-7;
  parameter real a2 = 4.35e-7;
  parameter real b = 0.7;
  parameter real x0 = 0.1;
  parameter real time_step_y = 0.001;
// parameters for picket memristor model
parameter real f_off = 3.5e−6;
parameter real a_off = 1.2e−9;
parameter real b_p = 500e−6;
parameter real wc = 107e−12;
parameter real D = 3e−9;
parameter real w_init = 0.5;
parameter real Ron_p = 100;
parameter real Roff_p = 2e5;
parameter real f_on = 40e−6;
parameter real a_on = 2e−9;
parameter real time_step_p = 1e−8;

// General threshold based model (biolek)
parameter real beta = 10e13;
parameter real Roff = 10e3;
parameter real Ron = 1e3;
parameter real Rinit = 5e3;
parameter real time_step_b = 10e−10;  // time step
for the transient analysis

// local variables to use within the model
real I_t;
real Vm;
real vini;
real taw;
real lambda;
real prob_t;
real randm_p;
real x;
real x_i;
integer seed;

/// Yakopcic parameters /////
real V_p;
real V_n;
real f_x;
real g_v;
real w_p;
real w_n;
real dxy_dt;
real taw_p;
real lambda_p;
real taw_n;
real V_x;

/// Pickett parameters /////
real I_m;
real R;
real dw_dt;
real w;
real V_t;
real w_1;
real i_on;
real i_off;
real i_t;
real i_max;
/* parameters for bipolar threshold-based memristor model */
real X;
real dx_dt;
real Vt;
real x_1;

begin

// seed = V(in_seed); // to initialize the random number generator

begin

if (model == 1) begin

$bound_step(time_step_y);

@(initial_step)begin

x = x0;

V.x = 1.5;

Vp = 1.5;

Vn = 0.5;

seed = 584;

end

Vm = V(tp, bn);

// stochastic setting of the threshold voltage

taw = pow(10,(-2.67*abs(Vm)+5.43));

// calculation of the tau parameter for the poisson distribution

lambda = 1/taw;

prob_t = lambda*time_step_y;

randm_p = abs($rdist_uniform(seed,0,1));

if(Vm > 0) begin
if (prob_{t} >= randm_{p}) begin
  // stochastic induction on the switching operation
  if (V_m < 1.5) begin
    V_{x} = V_m - 0.1;
  end
  else if (V_m > 1.5) begin
    V_{x} = 1.5;
  end
end
else if (prob_{t} < randm_{p}) begin
  V_{x} = 1.5;
end
end

V_p = V_{x};

///// regular operation /////
if (V_m > 0) begin
  if (x >= x_p) begin
    w_p = ((x_p - x)/(1 - x_p)) + 1;
    f_{x} = (exp(-alpha_p*(x - x_p))) * w_p;
  end
  else if (x < x_p) begin
    f_{x} = 1;
  end
  // setting of g(V(t))
  if (V_m > V_p) begin
    g_{v} = A_p * (exp(V_m) - exp(V_p));
  end
  else if (V_m <= V_p) begin
    g_{v} = 0;
  end
end
end
else if (Vm <=0) begin
  if (x <= 1-xn) begin
    wn = x/(1-xn);
    f_x = (exp(alpha_n*(x+xn-1)))*wn;
  end
  else if (x> 1-xn) begin
    f_x = 1;
  end
if (Vm < - Vn) begin
  g_v = -A_n*(exp(-Vm) - exp(Vn));
end
else if (Vm > -Vn) begin
  g_v = 0;
end
dxyDt = g_v*f_x;
x = dxyDt*time_step_p + x_i;
if (Vm >= 0) begin
  I_t = a1*x*sinh(b*Vm);
end
else if (Vm < 0) begin
  I_t = a2*x*sinh(b*Vm);
end
I(bn,tp) <+ I_t;
x_i = x;
Vp = 1.5;
end
else if (model == 2) begin

$bound_step(time_step_p);

@ (initial_step) begin

w_1 = w_init*D;

vini = 0;

seed = 24883;

i_off = 115e−6;

i_on = 8.9e−6;

i_t = 115e−6;

i_max = −inf;

end

Im = I(tp, bn);

if (Im > i_max) begin

i_max = Im;

end

/// Stochastic behavior of the threshold currents

taw = pow(10, (−2.67∗R∗abs(Im)/5 + 5.43));

// calculation of the tau parameter for the poisson distribution

lambda = 1/taw;

prob_t = lambda∗time_step_p∗1e4;

randm_p = abs($rdist_uniform(seed, 0, 1));

if (prob_t >= randm_p) begin

// stochastic induction on the switching operation

if (Im < 0) begin

if (((abs(Im) >= 4.5e−6) && (abs(Im) <= 8.9e−6)) begin


\[ \text{i\_on} = \text{abs}(\text{Im}) - 1e^{-7}; \]

end

else begin
  \[ \text{i\_on} = 8.9e^{-6}; \]
end
end

else if (\text{Im} \geq 0) begin
  if \((\text{Im} > 50e^{-6}) \&\& (\text{Im} \leq 115e^{-6})\) begin
    \[ \text{i\_t} = \text{Im} - 1e^{-7}; \]
  end
  else begin
    \[ \text{i\_t} = 115e^{-6}; \]
  end
end
end
else begin
  \[ \text{i\_on} = 8.9e^{-6}; \]
  \[ \text{i\_t} = 115e^{-6}; \]
end

/// threshold period application ///
if \(\text{i\_t} \neq 115e^{-6}\) begin // if the threshold was changed
  \[ \text{i\_off} = \text{i\_t}; \]
end

@\((\text{cross(Im-i\_max,+1)})\)begin
  \[ \text{i\_off} = 115e^{-6}; \]
end

/// Operation of the memristor
if \((\text{Im} \geq 0)\) begin // Off switching case
  \[ \text{dw\_dt} = \text{f\_off} \times \text{sinh(Im/i\_off)} \times \]

\[
\exp(-\exp((w_1-a_{off})/wc - \text{abs}(\text{Im}/b_p)) - w_1/wc);
\]

end

else if (Im < 0) begin

\[
dw_{dt} = f_{on} \times \sinh(\text{Im} / i_{on}) \times 
\exp(-\exp((a_{on} - w_1)/wc - \text{abs}(\text{Im}/b_p)) - w_1/wc);
\]

end

w = dw_{dt} \times \text{time\_step\_p} + w_1;

if (w > D) begin

w = D;

dw_{dt} = 0;

end

if (w <= 0) begin

w = 0;

dw_{dt} = 0;

end

R = R_{on} \times (1 - w/D) + R_{off} \times w/D;

V_t = R \times \text{Im};

V(bn, tp) <= V_t;

w_1 = w;

end

@end

\\ /////// Bipolar threshold-based memristor model //////////

//

else if (model == 3) begin

\$bound\_step(time\_step\_b);

@(initial\_step) begin

\text{vini} = 0;

\text{Vt} = 4.6;

\text{seed} = 584;

end
\[ x_{\cdot 1} = R_{\text{init}}; \]
end
\[ V_m = V(tp, bn); \]
// stochastic setting of the threshold voltage
\[ \text{taw} = \text{pow}(10, (-2.67 \times \text{abs}(V_m) + 5.43)); \]
// calculation of the tau waiting parameter
\[ \lambda = \frac{1}{\text{taw}}; \]
\[ \text{prob}_t = \lambda \times \text{time\_step\_b}; \]
\[ \text{randm}_p = \text{abs}($\text{rdist\_uniform}(\text{seed}, 0, 1)); \]
if (\text{prob}_t >= \text{randm}_p) begin
// stochastic induction on the spiking operation
if ((\text{V}_m > 0) && (\text{V}_m <= 4.6)) begin
\[ \text{V}_t = \text{V}_m - 0.01; \]
end
else if ((\text{V}_m < 0) && (\text{abs}(\text{V}_m) <= 4.6)) begin
\[ \text{V}_t = \text{V}_m + 0.01; \]
end
end else if (\text{prob}_t < \text{randm}_p) begin
\[ \text{V}_t = 4.6; \]
end
\[ d\text{x}_{\cdot dt} = \beta \times (\text{V}_m - 0.5 \times (\text{abs(\text{V}_m+\text{V}_t)} - \text{abs(\text{V}_m-\text{V}_t)})); \]
if (\text{d}x_{\cdot dt} > 0) begin
if ((\text{X} > \text{R}_{\text{off}})) begin
\[ \text{d}x_{\cdot dt} = 0; \]
end
end
if (\text{d}x_{\cdot dt} < 0) begin
if ((\text{X} < \text{R}_{\text{on}})) begin
\[ \text{d}x_{\cdot dt} = 0; \]
end

$$ X = \frac{dx}{dt} \times \text{time\_step\_b} + x_1; $$

if (X<\text{Ron}) begin
    X = \text{Ron};
end

if (X>\text{Roff}) begin
    X = \text{Roff};
end

I(\text{bn}, \text{tp}) <+ \frac{\text{Vm}}{X};

x_1 = X;

V_t = 4.6;

end // end bipolar model

end // end analog

endmodule // end model

A.2 SPICE Model

*******************************************************************************
*** Stochastic Threshold Based Memristor
*******************************************************************************
simulator lang= spice

*******************************************************************************
**** Memristor Subcircuit
*******************************************************************************
.subckt memR_TH plus minus thresh $PARAMS:

.PARAM
+ Ron=1k Roff=10k Rinit=1k beta=1E13 Vt=4.6

*memristive port

Gpm plus minus value='\text{\textasciitilde}-V(plus, minus)/V(x)'
**************
**** Integrator model

Gx 0 x value='beta*(V(plus,minus)-0.5*(abs(V(plus,minus)+abs(V(thresh)))-abs(V(thresh)))#1p*(stpss(V(plus,minus))
+stpss(-V(plus,minus))*stpss(V(x)-Ron))'

Raux x 0 1T
Cx x 0 1p IC=Rinit

************
**** Smoothed functions

.param b1=10u b2=10u
.param stps(x,b)="-1/(1+exp(-x/b))"
.param abss(x,b)="x*(stpss(x,b)-stpss(-x,b))"
.param fs(v,b)="beta*(v-0.5*(abss(v+Vt,b)-abss(v-Vt,b)))"
.param ws(x,v,b1,b2)="stpss(v,b1)*stpss(Roff-x,b2)+stpss(-v,b1)*stpss(x-Ron,b2)"
.param stpss(x) = "0.5*(1+sgn(x))"
.ends memR_TH

************

**** Stochastic Threshold setting

************

.PARAM alpha=-2.76, epsilon=5.43, delta_t = 1e-8
.param stpss(x) = "0.5*(1+sgn(x))"
*for standardizing the gaussian distribution of
*the resistor noise (Y = (X-mean)/sigma)
.param stnd(x)="(x-0.48)/0.16"

***** Probability calculation

*** Input voltage Vm

Vin inp 0 sin 0 5 50meg
****Random noise source
R0 r_p 0 2G noiseon=yes
I0 0 r_p 250p
*transformation of the gaussian noise into a uniform with
*the approximation of the erf(x) function (error function)
E_rp erp 0 value='1-
  sqrt(1-exp(-((2*stnd(V(r_p))ˆ2)/3.14)))'

****Calculated Poisson Probability
Etau tau_p 0 value="1/10*(alpha*abs(V(inp)) + epsilon)"
** to add the condition for the input to be below the threshold
E_p Ep 0 value="delta_t*V(tau_p)*stpss(4.6-abs(V(inp)))"
E_dif dif_p 0 value="V(Ep)-V(erp)"

**** Original threshold voltage************
i1 0 1 4.6
r1 1 0 1
g4 4 0 value='−V(1)'
r4 4 0 1

*** voltage controlled resistor to act as a switch
G1 1 2 VCR PWL(1) dif_p 0 −0.0001v,10000G 0.0001v,1m
G2 0 2 value="V(inp)"
r3 2 0 1

Xmem inp 0 1 memR_TH

.tran 0.1n 1u noisefmax=1G
.probe
Appendix C

List of Publications

Journals


Journals Submitted and Under Preparation


Peer Reviewed Conferences


Talks


