Design of a Device-aware Memristive Neuromorphic System with Spike Timing Dependent Learning

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Md Musabbir Adnan

December 2021

Acknowledgments

I would like to thank my advisor Dr. Garrett Rose for his constant support and cooperation throughout the time I have spent pursuing my PhD at the University of Tennessee, Knoxville. I would also like to thank my doctoral committee members Dr. James Plank, Dr. Ahmedullah Aziz and Dr. Andy Sarles for serving on my committee and spending their valuable time.

I would also like to thank my labmates Sagarvarma Sayyaparaju, Sherif Amer, Ryan Weiss, Nicholas Skuda, Samuel Brown, John Murray, Md Badruddoja Majumder, Mesbah Uddin and Gangotree Chakma for their constant help and interesting discussions throughout my graduate studies.

I would also like to take this opportunity to express my gratitude to my wife, family and friends for their unequivocal support in the pursuit of my PhD in a foreign country. Lastly, I want to thank my mentor Mr. Prodip Kumar for inspiring me throughout my academic life to achieve this goal.

Abstract

Spiking neural networks (SNN) present a biologically plausible energy efficient learning platform by encoding data into sparse spiking events. The discovery of memristor, a nanoelectronic device, has accelerated the development of SNNs, as the inherent plasticity of the device makes it a suitable candidate to imitate a biological synapse. In spite of offering the advantages of nanoscale sizing, non-volatility, compatibility with the fabrication process, and energy efficient operation, there remain significant challenges to form and program these devices to be used in a CMOS/memristor hybrid neuromorphic system. This work introduces a forming and programming circuit along with a scan-in scheme to program both CMOS and multi-bit memristor devices. Using the formed and programmed devices, a twin memristor synapse is constructed that exhibits both excitatory and inhibitory behavior. The function of the proposed synapse as an interface between neurons is described, implementing spike-timing-dependent plasticity (STDP) based learning to update the synaptic weight. The sparse recurrent SNN built using the designed twin memristor synapse is found to have high accuracy for different data classification applications. The twin memristor synapse design is enhanced further to support crossbar based single layer spiking neuromorphic system with unsupervised learning to counter the memristive device imperfections, leveraging pulse width modulated STDP in the synapses and homeostatic plasticity in the neurons. The performance of the proposed spiking neuromorphic system was evaluated by using the network to recognize a handwritten-digits dataset, quantifying the impact of different network parameters such as the number of output neurons, training epochs, capacitors used in the design etc. Moreover, the effect of memristive device imperfections such as device parameter asymmetry, endurance degradation, array failure, fabrication process immaturity on the performance of the network was considered, demonstrating a high degree of robustness.

As an application, the designed neuromorphic system was combined with a SPAD based vision sensor with Address Event Representation (AER) readout to enable efficient on-chip processing of spatio-temporal data.

Table of Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Research Goal . . . 3

1.3 Research Contribution . . . 3

1.4 Dissertation Overview . . . 4

2 Background **6**

2.1 Memristors . . . 6

2.1.1 Memristor Model . . . 7

2.1.2 Memristor Forming and Programming . . . 8

2.2 Spike-Timing-Dependent Plasticity . . . 10

2.3 Related Work on Spiking Neuromorphic Systems . . . 11

3 Forming and Programming of Memristors **15**

3.1 Forming Circuit . . . 15

3.2 CMOS/Memristor Programming . . . 17

3.2.1 Programming Logic . . . 17

3.2.2 Application : MrDANNA Core Programming . . . 21

3.2.3 Performance Analysis . . . 23

3.3 Multi Level Memristor Programming . . . 25

3.3.1 Programming Scheme . . . 25

3.3.2 Reading Scheme . . . 30

3.3.3 Erasing Scheme . . . 30

3.3.4 Simulations . . . 32
3.3.5 Performance Analysis . . . 37

4 Twin Memristor Synapse **40**
4.1 Structure and Operation . . . 40
4.2 STDP in Twin Memristor Synapse . . . 43
4.3 Design Parameters and STDP Performance . . . 47
4.4 Sparse Spiking Neural Network . . . 49
4.5 Energy Estimates . . . 53

5 Memristive Crossbar Based Spiking Neuromorphic System **55**
5.1 System Architecture . . . 56
5.1.1 Input Encoding . . . 56
5.1.2 Input Neuron and Synapse . . . 56
5.1.3 Output Neuron . . . 66
5.2 Network Architecture . . . 73
5.3 Training . . . 77
5.4 Testing . . . 79
5.5 Performance Analysis . . . 81
5.5.1 Effect of changing Number of Neurons and Epochs . . . 82
5.5.2 Effect of Increasing Capacitors . . . 84
5.5.3 Effect of Switching Mismatch of the Memristors . . . 86
5.5.4 Effect of Memristor Aging . . . 88
5.5.5 Effect of Memristor Device Failure . . . 88
5.5.6 Effect of Process Variation . . . 90
5.5.7 Area Overhead and Energy Consumption . . . 92
5.5.8 Other Design Considerations . . . 95

6 Application of Memristive Spiking Neuromorphic System: SPAD-based Vision Sensor **98**
6.1 SPAD based Vision Sensor . . . 98

6.2 Proposed SPAD based Vision Sensor with Memristive Neuromorphic System 99
6.2.1 SPAD based Vision Sensor with AER Readout 99
6.2.2 SPAD Pulse to Temporally Coded Spike Decoder 102
6.2.3 Memristor Crossbar based Spiking Neuromorphic System 104
6.3 Results 108

7 Conclusions and Future Work **114**
7.1 Conclusions 114
7.2 Future Work 115

Bibliography **117**

Appendices **130**
A Verilog Codes 131
A.1 Input Neuron 131
A.2 Output Neuron Control Block 134
A.3 SPAD to TCS Decoder 137
B Testing Results 142

Vita **146**

List of Tables

3.1 Truth table for forming and programming operation. 18
3.2 Energy consumption of the proposed multi-bit memristor programming and reading scheme 39
4.1 Energy estimates of the proposed synapse 54
5.1 Area overhead and energy consumption of relevant memristor based spiking neuromorphic systems. 94
6.1 The delay required for each pixel intensity to generate TCS from SPAD events. 105
6.2 Performance Comparison of Proposed Memristive Spiking Neuromorphic Processing with Other Event based Processing 112

List of Figures

2.1 (a) Circuit symbol of the memristor (b) I-V characteristics of the memristor device model with parameters adjusted to match the experimental data. [11] 9

2.2 The STDP behavior defined by the anti-symmetric piece-wise exponential curve. 12

3.1 The design of the forming circuit incorporating the programming operation. 16

3.2 Control circuitry for memristor forming and simultaneous programming of CMOS/Memristor memory. 18

3.3 Timing diagram showing the waveforms for programming the D Flip-Flops and selectively programming memristors after initialization. 20

3.4 Change of memristance of the 4 memristors, starting from pre-formed stage, then being formed to LRS, then reset to HRS, and finally selectively programmed to LRS. The memristors M0, M2, M3 are set to LRS and M1 remains at HRS. 22

3.5 Memristive Dynamic Adaptive Neural Network Array (mrDANNA) architecture with zoomed in view of each core, including forming and programming circuits. 22

3.6 Average power dissipation for forming and programming circuit with varying memristor LRS/HRS values with supply voltage $V_{DD} = 3.3V$. 24

3.7 Average power dissipation for forming and programming circuit with varying supply voltages for LRS/HRS of 30K/300K. 26

3.8 Change of memristance as a voltage is applied across the memristor terminals. At first, the memristor is in LRS. When the applied voltage exceeds the switching threshold, the memristance starts to rise. Before reaching HRS, it starts to plateau and is finally set to HRS. 27
3.9 Block diagram of the proposed programming scheme. 29
3.10 Block diagram of the proposed reading scheme. 31
3.11 During the programming operation, the memristor is driven by the decoder and directly connected to ground. During read operation, the memristor is still driven by the decoder, but it is connected in voltage divider configuration, in series with a resistor. For the erase operation, voltage is driven in opposite direction to the programming operation. 31
3.12 Waveforms that are applied during the programming scheme along with the states of the scan registers to program the memristors. 33
3.13 Change of memristance of the 8 memristors with time, connected to the decoders A and B. The same colored plots indicate the memristance of the memristors that are programmed simultaneously at the same programming cycle. 35
3.14 Waveforms that are applied during the reading scheme along with the states of the scan registers to read the data from memristors. 36

4.1 The proposed twin memristor synapse along with the block diagram of the synaptic control block, which a part of the pre-neuron. The synaptic control block houses shift registers for storing timing information of the firing events, a pulse generator to apply appropriate pulse width modulated waveforms, and an output control unit to disable outputs when not required. 41

4.2 Waveforms showing LTP/LTD with different cycle gaps. F_{Pre} and F_{Post} signals are the firing events from pre and post-neuron, respectively, V_{PSN} is the voltage applied at the postsynaptic node by the post-neuron, V_p and V_n are the voltages at memristors M_p and M_n, applied by the synaptic control block, V_{diff} is the voltage across the memristors during the learning phase only. The twin memristors are shorted at both ends during the learning phase, and hence V_{diff} is across both the memristors. 45

4.3 Change of STDP curve with clock frequency of the twin memristor synapse (a) Clock frequency of 25MHz (b) Clock frequency of 100MHz 48

4.4 Effect of device parameter mismatch on STDP curve of the twin memristor synapse. (a) Switching threshold voltage mismatch, keeping V_{tn} constant at 1V and changing V_{tp}. (b) Switching time mismatch, keeping t_{swn} fixed at $1\mu s$ and changing t_{swp}. 50

4.5 Performance of the best scoring networks for Iris, Wisconsin Breast Cancer and PIMA Indian Diabetes dataset, with different STDP cycle tracking abilities. 52

5.1 The input image is pre-processed to reduce the dynamic range of the pixel values (a) Original image with pixel values from 0 to 16 (b) Converted image with pixel values from -4 to +4 57

5.2 Proposed input encoding scheme to transform the input pixel from values into temporally coded spikes (TCS). A time frame of 8 timeslots is considered with output neuron spiking at the middle of the time frame shown with a vertical red line. The input spikes are placed such that positive and negative polarity pixels are placed before and after the output neuron spike, respectively, with higher magnitude pixels being placed closer to the time frame. 57

5.3 The input and output neurons being connected by the twin memristor synapse design. The vertical bar on the memristor symbol of M_p indicates the negative polarity terminal of the device. The bar is not shown for M_n, as the synapse design does not require its memristance to change and hence, it could be placed either way, disregarding the polarity. 59

5.4 TCS and resulting pulses produced by the input neuron to be applied to the twin memristor synapse for the three operating phases. Timeslots are also shown which could be related to Fig. 5.2 for clarity. (a) TCS and pulses during accumulation phase, which occurs from timeslots 1 to 4 for positive intensity pixels. Negative intensity pixels are not accumulated and ignored (b) TCS and pulses during depression phase, occurring from timeslots 5 to 8 for negative intensity pixels when the associated output neuron spikes due to the accumulation from other connections to it. (c) TCS and pulses during potentiation phase, occurring from timeslots 9 to 12 for positive intensity pixels when the associated output neuron spikes. The applied pulse from the input neuron is the same as the one applied during the accumulation phase, only delayed by 8 clock cycles. The memristor M_n is left floating during depression and potentiation, and hence the memristance does not change. It only contributes during the accumulation phase. 59

5.5 (a) The pulse duration over the memristive device with respect to the difference in spike timing of input and out neuron in terms of number of clock cycle for different clock frequencies. (b) The resulting change of conductance for different clock frequencies. Here, $\Delta T = t_{post} - t_{pre}$, depicting the clock cycle difference between post and pre-neuron spike. 62

5.6 Schematic diagram of the designed input neuron and the highlighted active blocks during different phases of operation. TCS is the encoded spike signal, CLK is the system clock, $CLK8$ is derived from the system clock to have 1/8th of the frequency, POL is the signal indicating the polarity of the input pixel, and $VoMp$ and $VoMn$ outputs are the pulse width modulated signals generated by the input neuron that are applied to the memristors M_p and M_n, respectively. (a) Input neuron during accumulation phase for an input pixel intensity of +2 (b) Input neuron during depression phase for an input pixel intensity of -3 (c) Input neuron during potentiation phase for an input pixel intensity of +2. 64

5.7 Cadence Spectre simulation of the designed input neuron to generate pulse width modulated waveform from the TCS for different phases of operation. A 100 MHz system clock is represented by CLK, which is divided by 8 to generate $CLK8$ signal. The encoded spike signal provided to the input neuron is represented by TCS signal. The system reset indicating the start of an input image is represented by $RESET$. The outputs from the input neuron that drive the memristors M_p and M_n are represented by V_M_p and V_M_n, respectively, while $Vdiff\_M_p$ and $Vdiff\_M_n$ represents the voltage across them, respectively. V_Post represents the postsynaptic node voltage, supplied by the output neuron. (a) Example waveform for an input pixel intensity of +2 showing accumulation phase from 10ns to 50ns and potentiation phase from 90ns to 130ns. (b) Example waveform for an input pixel intensity of -3 showing depression phase from 50ns to 90ns. 65

5.8 Input neuron layout designed with IBM 65nm CMOS10LPe PDK with an area of $34\mu m \times 25\mu m$. 67

5.9 The schematic diagram of the proposed output neuron with dynamically adjusted capacitance and the Winner-Takes-All bus connection. A control block generates the necessary signalling for the WTA interface and capacitance switching. 69

5.10 Simulation of the proposed output neuron using Cadence Spectre to demonstrate the homeostasis mechanism by showing reduced accumulation with each spike. The system clock is represented by the CLK signal. The applied input voltage for producing the input current (i_{in}) to the integrator is shown by the *Input Spike* signal. V_Post signal shows the voltage driven at the postsynaptic node by the neuron. The accumulated voltage at the integrator is represented by the V_{accum} signal along with the threshold of the neuron, indicated by a dashed blue line, upon reaching which, a spike is produced that is shown by *Spike* plot. 71

5.11 Schematic diagram of the control block of the proposed output neuron. The control block provides all the signals needed to carry out all the operating phases, in addition to interfacing with the WTA bus. 72
5.12 Output neuron layout designed with IBM 65nm CMOS10LPe PDK with an area of $122\mu m \times 38\mu m$. 74
5.13 The structure of the proposed single-layer SNN built with M rows of input neurons, N columns of output neurons, and a crossbar of $M \times N$ twin memristive synapses. 76
5.14 (a) The synaptic weights associated with the best performing neurons learning each digit. The conductance in μS is shown with a colorbar. (b) Four distinct output neurons learn four different forms of the digit '3'. (c) Testing result represented by a confusion matrix with the network trained with 50 columns and 10 epochs after applying the testing dataset. 80
5.15 Change of accuracy of the designed spiking neuromorphic system with the network parameters. (a) Change of accuracy with increasing number of output neurons with 20 training epochs (b) Change of accuracy with increasing number of training epochs with 200 output neurons. 83
5.16 (a) The accuracy of the network increases as the number of output neurons and training epochs increase, showing more output neurons and training epochs yielding higher accuracy, but the rate of change slows down. (b) The accuracy of the network increases as the number of capacitors and the maximum capacitance increase for a network with 50 output neurons and 10 training epochs. The increase with number of capacitors is higher than the increase with maximum capacitance. 85

5.17 (a) Impact of switching time mismatch on the performance of the network and recovery of the performance using a pulsing scheme with duty cycle modulation (b) Impact of switching voltage mismatch on the performance of the network and recovery of the performance using a pulsing scheme with duty cycle modulation and clock frequency reduction. All the networks here has 200 output neurons and uses 10 epochs. (c) Change of pulsing scheme for duty cycle modulation technique. The pulse is applied to the memristor M_p only during potentiation. V_M_p represents the pulse applied to M_p, V_Post represents the postsynaptic node voltage and $Vdiff\_M_p$ represents the voltage difference across the memristor. The dashed blue line represents the memristor switching threshold. 87

5.18 (a) Impact of resistance degradation, due to memristor aging, on the performance of the network (b) Impact of memristor device failure on the performance of the network, when the memristive devices appear to be stuck and not switch. All the networks here has 200 output neurons and uses 10 epochs. 89

5.19 (a) Impact on learning speed of the network considering the effect of process variation with 200 output neurons (b) Performance of the network with process variation affecting all the memristive device parameters with 200 output neurons and 20 training epochs. 91

5.20 Effect of process variation on switching window and devic failure (a) HRS and LRS distribution of a 64×200 network containing 25600 memristors with 30% RSD. (b) Increase of memristive device failure with respect to increasing RSD. 93

5.21 Effect of interconnect resistance on the network performance with 200 output neurons. 97

6.1 Conventional SPAD based vision sensor [81] 100

6.2 Proposed SPAD based vision sensor with AER readout connected to spiking neuromorphic system for on-chip processing via an interfacing unit to convert SPAD pulses into Temporally Coded Spikes (TCS). 101

6.3 Block diagram of SPAD to TCS decoder. The DeMux provides the asynchronous SPAD spike to the appropriate rows, which is first synchronized and then converted to the TCS by Pulse Synchronizer and Delay Block, respectively. 103
6.4 Synchronous SPAD event to TCS conversion. Looking at the position of spikes at SPAD timeslots, the delay required for conversion can be calculated. . . . 105
6.5 The Delay Block to delay the synchronized SPAD event by the required amount. The counter tracks the timeslot of the SPAD event. When the SPAD event occurs, the delay decoder converts the count value into the required delay value. The binary to thermometer decoder generates the select signals for the delay line using the delay value provided by the delay decoder to produce the TCS. 106
6.6 SPAD to TCS decoder layout designed with IBM 65nm CMOS10LPe PDK with an area of $43\mu m \times 30\mu m$. 107
6.7 Change of accuracy with increasing number of output neurons of the spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor with 10 epochs. 109
6.8 Change of accuracy with increasing number of epochs of the spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor with 200 output neurons. 109
6.9 Effect of increasing number of neurons and epochs on the accuracy of the spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor 111
6.10 Effect of increasing number of capacitors and maximum capacitor on the accuracy of spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor. 111

B.1 Test setup for testing the input neuron with a logic analyzer. 143

B.2 Input neuron testing for TCS representing positive inputs. *resetn* is the global reset signal, *clk* is the clock signal, *scan_enable* enable the shifting of data, *scan_in* is the data to be scanned in, *scan_out* is the data that is shifted out. (a) Waveforms for TCS representing the input '+1' that produces a one cycle wide positive pulse during accumulation and potentiation. (b) Waveforms for TCS representing the input '+3' that produces a three cycle wide positive pulse during accumulation and potentiation. 144

B.3 Input neuron testing for TCS representing negative inputs. *resetn* is the global reset signal, *clk* is the clock signal, *scan_enable* enable the shifting of data, *scan_in* is the data to be scanned in, *scan_out* is the data that is shifted out. (a) Waveforms for TCS representing the input '-1' that produces a one cycle wide negative pulse during depression. (b) Waveforms for TCS representing the input '-3' that produces a three cycle wide negative pulse during depression. 145

Chapter 1

Introduction

1.1 Motivation

The saturation of Moore's law and collapse of Dennard scaling has ushered in a new era in computing, setting up a paradigm shift from the traditional von Neumann architectures. The performance limitation caused by having separate memory and processing units is referred to as the von Neumann bottleneck, which results in the inefficiency of these machines for cognitive tasks performed naturally by human brains [46]. Owing to the computational efficiency and parallelism of the human brain, researchers have concentrated on exploring brain-inspired computational algorithms, making significant advancement in pattern recognition tasks [39]. The most common algorithm developed is the gradient descent algorithm, which calculates the error function gradient and propagates it backward through a multi-layer artificial neural network (ANN) [74] to reach optimal synaptic weights. However, the developed algorithms still run on von Neumann machines using extensive resources and expending energy [68].

In order to achieve energy efficiency and parallelism similar to the human brain, hardware implementations of spiking neural networks (SNN) have been proposed, which apply biologically plausible learning mechanism, wherein the data is communicated and processed across synapses and neurons by encoding them as spike information [18]. Unlike the supervised back-propagation of ANNs, SNNs usually use unsupervised learning rules such as spike-timing-dependent plasticity (STDP) [85]. In the weight update rule based

on STDP, the synaptic weight is adjusted by considering the timing and causality of the neurons' spikes, without the need to know the corresponding output associated.

In 2008, a nanoscale two terminal called device called the memristor (shortened term for "memory resistor") was discovered at HP Labs [86]. It was proven to have intermediate states between a Low Resistance State (LRS) and High Resistance State (HRS), making it a prime candidate to implement electronic plastic synapses amenable to STDP based learning [84]. Proposed by Leon O. Chua in 1971 [29], the memristor offers nanoscale footprint, high write endurance, non-volatility, full compatibility with CMOS fabrication methodology, and energy efficiency [12]. As a result, memristors have found usage as memory devices as well as synapses. However, it needs to go through a dielectric breakdown process known as electroforming [9], before it can be used as an analog memory device. Furthermore, memristors may need to be programmed to a certain resistance state depending on the application. Hence, a system level approach for forming and programming memristor arrays is needed.

Due to the aforementioned advantages provided by memristor, it has garnered widespread attention from researchers to be used as synapses in a CMOS-memristor hybrid neural network setup [93, 46, 24, 37]. However, most the the methods proposed either use additional control circuits or provide precisely shaped analog pulses to generate overlapping forward and backward spikes to change the memristance. Both of the implementations increase circuit overhead and constructing a large memristive crossbar becomes difficult. In regards to the training mechanism, a large number of designs use supervised learning [95, 82, 75], although, unsupervised learning mechanism is perceived to be more biologically plausible [71, 34, 37]. They employ synaptic update rules like STDP in the synapses as well as lateral inhibition and homeostatic plasticity in the neurons. Although a few of the works concentrate on developing a process variation immune method to tackle memristive imperfections [70, 62], the homeostasis is performed in software simulations, failing to harness the energy benefit brought about by a full hardware implementation.

To overcome the limitations of the previously proposed approaches, this work presents a complete circuit to system level design, starting with forming and programming of memristors and moving on to the implementation of memristive crossbar based SNN that is able to

account for the device level imperfections such as endurance degradation, device failure rate, switching asymmetry and process variations.

1.2 Research Goal

The main objective of this work is to design a device-aware and robust spiking neuromorphic system, leveraging the potential of memristors. Before delving into system level applications, some circuit level issues needs to be addressed, beginning with the electroforming of memristors that initiates defects in oxide material to make it amenable to resistive switching. In addition to that, a protocol must be developed to systematically program a memristor array, along with CMOS memory elements for reconfigurability. Programming of memristors could be binary (only HRS and LRS) as well as multi-level depending on the demand of the application. For the neuromorphic system, the building blocks i.e., neurons and synapses must be tailored to the switching characteristics of the memristors and need to be energy efficient to justify the use of memristors in the design. Finally, the robustness provided by the memristor based neuromorphic system against device non-idealities must be analyzed on the application level.

1.3 Research Contribution

The primary contributions of this research work are listed as follows:

- A forming and programming circuit and a system level approach for programming CMOS memory in conjunctin with memristors in any CMOS/memristor based reconfigurable systems.
- An access scheme to read and write distributed multi-bit memristor memory sequentially, supporting different reading and writing approaches
- A twin memristor synapse able to implement both positive and negative weights with a synaptic control block to implement pulse-width modulated synchronous STDP.

- A STDP enabled sparse neuromorphic system utilizing digital pulses to overcome the need for precise analog spike shaping, offering control over the amount of weight change of the synapse for on-chip learning and noise immunity to communicate signals across the chip.
- A temporal input encoding methodology to provide input spikes and an input neuron design to transform the spikes into pulse width modulated signals, realizing synchronous STDP based learning
- An improved twin memristor synapse that to implement both excitatory and inhibitory behavior without requiring extra circuit overhead in the synapse, enabling easier construction of crossbars.
- An output neuron with realizing inherent homeostatic plasticity by leveraging dynamically adjusted integration capacitance to modulate accumulate rate, which supports lateral inhibition according to Winner-Takes-All logic to implement competitive learning.
- A crossbar-based single-layer spiking neuromorphic system with the proposed homeostatic neurons and twin memristor synapses that executes an unsupervised classification task by taking advantage of neuronal homeostasis and synaptic STDP.
- Study of the resilience of the proposed SNN to tackle the memristive device level imperfections such as switching mismatch, device failure rate, endurance degradation, and process variations.
- Application of the proposed memristive crossbar based SNN in a SPAD based vision sensor with Address Event Representation (AER) readout for event based data classification.

1.4 Dissertation Overview

The rest of the dissertation is organized as follows: chapter 2 provides a background on memristors and models used in this work as well as an overview of existing neuromorphic

systems. Chapter 3 describes the forming of memristors and establishes a programming protocol for CMOS/memristor reconfigurable systems as well as multi-level memristive memories. Chapter 4 introduces the design of twin memristor synapse and STDP based learning of a sparse SNN. Chapter 5 presents the memristive crossbar based SNN, evaluating the performance of the system with respect to design parameters and memristor device non-idealities and highlights the advantages in terms of robustness, energy consumption, and scalability. Chapter 6 explores the application of the developed SNN as an on-chip processor for a SPAD based vision sensor. Chapter 7 draws conclusion of the contributions and discusses the plan for future works.

Chapter 2

Background

2.1 Memristors

In 1971, Leon O. Chua proposed symmetry arguments in his paper [29], stating that the four fundamental electrical circuit variables, namely *voltage(V)*, *current(I)*, *charge(Q)* and *flux(ϕ)* must be connected to each other by six relations. Two of the relations (current & charge and voltage & flux) are given by the fundamentals definitions, as charge is the time integral of current while flux linkage is the time integral of voltage. The basic circuit components, namely resistor, capacitor and inductor define the relationship between voltage & current, charge & voltage and current & flux. Chua postulated that there must be another fundamental device that establishes the missing link between charge & flux, which cannot be built from a combination of only resistors, capacitors and inductors. This missing two terminal device was named memristor, short for "memory resistor", because of its characteristics of functioning like a resistor having memory. Recent findings at HP labs [86] have substantiated the existence of passive devices, the attributes of which are similar to those projected by Chua. Following that, devices exhibiting similar memristive properties have been fabricated using several materials such as oxides [43, 89, 59, 98], chalcogenides [54, 65], silicon [21, 60], organic materials [16], ferroelectric materials [26, 67], carbon nanotubes [49], etc.

2.1.1 Memristor Model

From a circuit and system level perspective, memristors work like variable resistors, the resistance (or memristance) of which can be altered by applying voltage above the switching threshold of the devices. The device memristance is typically bound between a upper and lower resistance limit, known as High Resistance State (HRS) and Low Resistance State (LRS), respectively. Memristors commonly find their usage in neuromorphic circuits, as synaptic weights can be encoded as memristance values. Analog conductance change between LRS and HRS also proves vital for learning and hence, accurate and compact device models are needed to capture and simulate the switching behaviour of memristors.

There are two main types of existing memristor models: device physics-based and empirical models. The models based on device physics have certain shortcomings, as the understanding of switching mechanism of memristors is not complete. Alternatively, the empirical models are based on the experimental results providing good accuracy while achieving fast simulation convergence. The approaches to empirical modeling considers an internal state variable, that changes when voltage or current is applied. Then a dependency is formulated between the internal state variable and the memristor device resistance. However, the internal state variable is not readily measured by practical experimentation and extracting parameters become challenging. Hence, an empirical compact model was proposed in [11], which directly uses the device memristance as the state variable, which can be readily obtained from the I-V characteristics of the device. The model captures the switching mechanism of the memristor by (1) parameterizing the switching threshold voltage, (2) establishing a polynomial relation between applied voltage and change in resistance due to it, and (3) fixing an upper and lower bound on the resistance. This model is given by equations 2.1 and 2.2.

$$\frac{dM}{dt} = \begin{cases} -C_{LRS}(\frac{V(t)-V_{tp}}{V_{tp}})^{P_{LRS}} f_{LRS}(M(t)), & V(t) > V_{tp} \\ C_{HRS}(\frac{V(t)-V_{tn}}{V_{tn}})^{P_{HRS}} f_{HRS}(M(t)), & V(t) < V_{tn} \\ 0, & \text{otherwise} \end{cases}, \tag{2.1}$$

where M is the memristance, $\frac{dM}{dt}$ is the rate of change of memristance, C_{LRS} and C_{HRS} are fitting coefficients, containing switching time, V(t) is the voltage bias applied to the memristor, V_{tp} and V_{tn} are the device switching thresholds, P_{LRS} and P_{HRS} are the polynomial coefficients, that control the non-linearity of the model, $f_{LRS}(M(t))$ and $f_{HRS}(M(t))$ are the window functions that account for the resistance bounds and are given by equation 2.2.

$$f(M(t)) = \begin{cases} \frac{1}{1+exp\left(\frac{\theta_{LRS}LRS-M(t)}{\beta_{LRS}\Delta r}\right)}, V(t) > V_{tp} \\ \frac{1}{1+exp\left(\frac{M(t)-\theta_{HRS}HRS}{\beta_{HRS}\Delta r}\right)}, V(t) < V_{tn} \end{cases}, \tag{2.2}$$

where β and θ are fitting parameters, Δr is the difference between HRS and LRS. Fig. 2.1a shows the circuit symbol used for the memristor. The current vs voltage characteristics of Fig. 2.1b shows the pinched hysteresis of the memristor device model with parameters adjusted to match the experimental data.

2.1.2 Memristor Forming and Programming

With the discovery of the memristor, multiple projects have been undertaken to integrate the memristor in logic design or use the device as a storage device by itself. Most common applications are found in programmable logic design, analog computing, dot product engines (DPE) and brain-inspired computing [61]. However, the memristor in its original state is essentially an insulator and does not develop any resistance switching characteristics before going through a one-time forming process. The switching mechanism of the memristor is controlled by the formation and breakage of conductive filaments. Electro-forming (also called forming) is a method that introduces defects in the form of oxygen vacancies in the oxide, which subsequently leads to a change in resistance. The forming voltages are typically reported to be higher than the switching threshold of the memristor, even being higher than the normal operating voltage (V_{DD}) of CMOS devices. To overcome this challenge, an in-field forming circuit was proposed in [10], isolating the peripheral circuitry around the high voltage tolerant circuit that initiates the forming procedure.

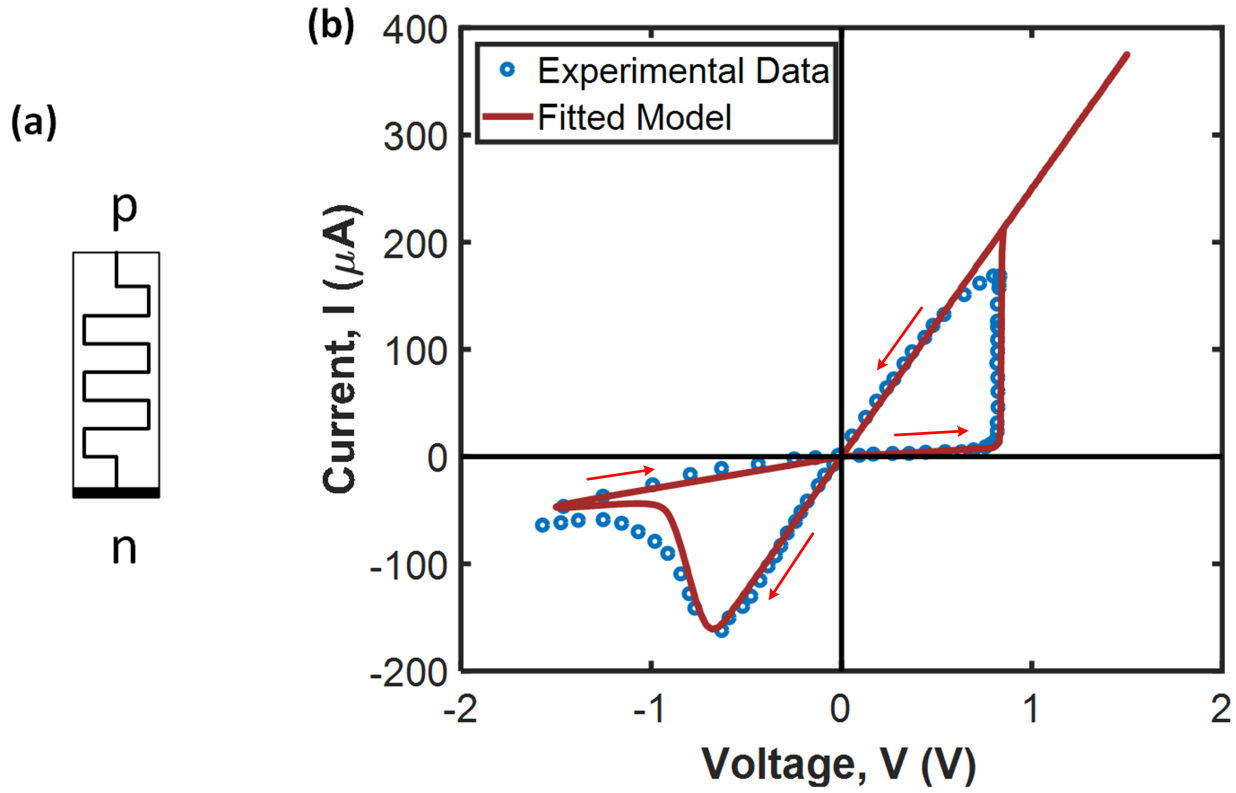


Figure 2.1: (a) Circuit symbol of the memristor (b) I-V characteristics of the memristor device model with parameters adjusted to match the experimental data. [11]

Depending on the requirement of the application, the formed memristors are programmed either to binary states or as a multi-bit memory device to provide discrete analog resistance. Moreover, memristors can also be integrated into a high density system of distributed memory elements [24, 30] as well as random access crossbar memory arrays [99]. As a result, different methods have been presented to read from or write to the memristors. For programming to analog states, a reference resistor array along with comparator were used in [50]. In another approach [14], a feedback op amp was used for programming analog states to the the memristor. However, both methods suffer from the circuit overhead required by the feedback, which complicates the design. A read monitored write procedure was proposed by the authors of [99], which overcomes the issues with feedback method, suggesting a multi-state read/write method for a crossbar which is tolerant to drift. However, many applications require the simultaneous programming of memristor and CMOS logic circuits, which calls for a system level approach to program both. The sparse neuromorphic circuits presented in [24] requires the programming of both memristor and CMOS memory as they are responsible for realizing synaptic weight and delay, respectively. In theory, they could be programmed by dedicated pins for each memory devices, but is impractical for a chip design. Hence, in s reconfigurable CMOS/memristor-based hybrid design, a system-level access protocol is required to program the CMOS memory and memristor together.

2.2 Spike-Timing-Dependent Plasticity

Spike-timing-dependent plasticity (STDP) is an unsupervised synaptic update mechanism commonly found in biological neurons [17]. The update rule requires the synaptic connection between neurons to be strengthened or weakened depending on the timing of the spikes from the neurons. The idea of STDP is derived from Hebbian learning [40], which is a simple rule to update synaptic weights. When a neuron continuously contribute to another neuron's spiking, the weight of the synapse connecting the neurons is increased. Here, a neuron preceding a synapse is defined to be the *pre-neuron*, while the succeeding one is defined to be the *post-neuron*. Pre-neuron is considered to have contributed to the post-neuron spike if causality is maintained, meaning pre-neuron must spike before the post-neuron. In that

case, long term potentiation (LTP) occurs, which means the strengthening of the synaptic connection. Conversely, if pre-neuron spikes after post-neuron, and the relationship is anti-causal, long term depression (LTD) takes place, which means the weakening of the synaptic connection. The amount of change in the synaptic weight is dependent on the temporal difference between the spikes. The STDP rule dictates that amount of change is higher when the time difference between the spike is smaller and vice versa [85]. The piece-wise exponential STDP behavior is explained with equation 2.3 and Fig. 2.2.

$$\Delta w(\Delta t) = \begin{cases} -A^- \exp(\Delta t/\tau^-) & \text{if } \Delta t < 0 \\ A^+ \exp(-\Delta t/\tau^+) & \text{if } \Delta t \geq 0 \end{cases}, \tag{2.3}$$

where, Δt is temporal difference between the post- and pre-neuron fire ($\Delta t = t_{post} - t_{pre}$), Δw is the synaptic weight change, normalized by the maximum weight (w_{max}). A^+ and A^- are the maximum possible synaptic modification. τ^+ and τ^- are defined to be the time frame over which strengthening or weakening may occur, respectively.

2.3 Related Work on Spiking Neuromorphic Systems

There have been numerous attempts to incorporate memristors in a neuromorphic system to mimic a plastic synapse amenable to on-chip learning. Hence, there exists a multitude of synaptic dynamics and architectures for the implementation of memristive neural networks. In [93], a 1T1R crossbar using a selector transistor along with a ferroelectric tunnel memristor (FTM) was built to work as a neuromorphic system. A binary memristor was also used in a 1T1R synapse configuration using HfO_2 in [68]. While both the methods presented enable on-chip STDP based learning, using a selector transistor along with a memristor dismisses the density advantage of memristors.

In [1], the authors proposed a memristor bridge synapse that can realized both positive and negative weights. However, each synapse requires 4 memristors, and a differential amplifier is needed to transform the synaptic weight given by the memristors into a current, again diminishing the advantage of nanoscale form factor of memristors. In [53], a

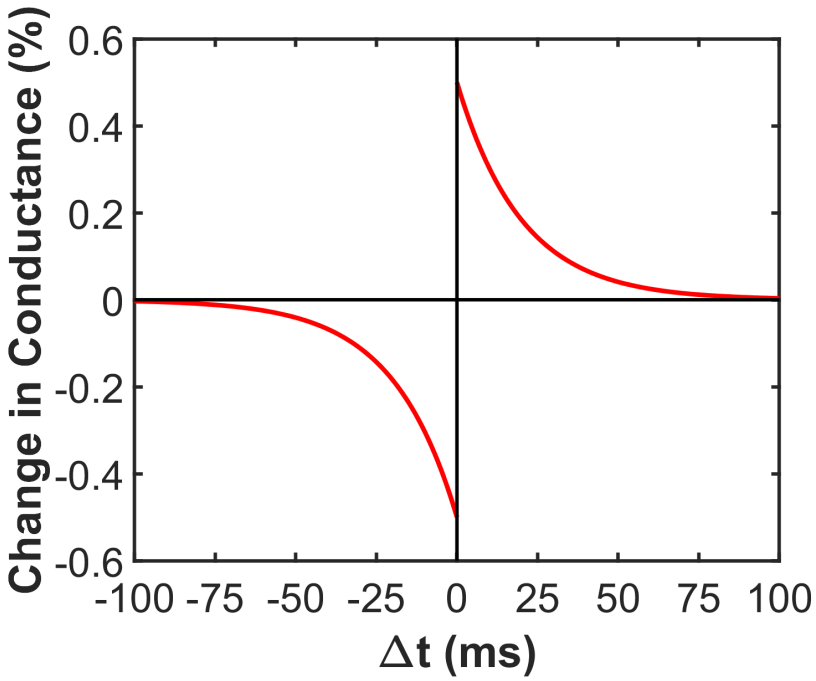


Figure 2.2: The STDP behavior defined by the anti-symmetric piece-wise exponential curve.

second generation current conveyor (CCII) approach was presented that reduces the circuit complexity for enabling crossbar operation. The CCII is housed inside the post-neuron, where a configuration bit defines the polarity of the memristor current injected into it. A similar method is followed in [56], to implement excitatory and inhibitory synapses by using different polarity voltages supplied by the presynaptic neuron for each case. Both the methods realize the polarity of the synapse as a function of neuron rather than the synapse itself.

For implementing STDP in memristive synapses, overlapping analog pulses were used in methods suggested in [56, 31, 37]. while the synapse design is simplified and the crossbar is readily constructed, the analog pulse shaping approach adds significant overhead in the neuron circuitry. Moreover, the analog pulses used are susceptible to noise and high fan-out loading effects which could lead to reduced performance of the designed network. To overcome the limitation of analog signalling, a digital binary STDP was proposed in [24]. The approach is more aligned with Hebbian learning than STDP, as the synaptic weight change only depended on the causality of the spikes. The amount of weight change was fixed and did not take into account the timing difference between the spikes. Furthermore, a synaptic control block was used that required the post-neuron spike to be routed back to the pre-neuron, making crossbar implementation difficult.

Another basic component of a neuromorphic system is the neuron, whose main function is the integrate the input current or charge supplied by the synapse and generate a spike upon crossing a preset threshold. Multiple neuron designs have been proposed in [45, 76, 8] using some form of the Integrate And Fire (IAF) or Leaky Integrate and Fire (LIF) [24, 95] concept. Although they are suitable for supervised learning algorithms, owing to the lack of homeostasis, they are not suitable for unsupervised learning algorithms. Homeostatic plasticity was implemented in a software simulation for a pattern classification application with unsupervised learning in [71, 34], but a description of a practical hardware implementation was not provided. In other approaches of [73, 57], the gain of the synapse was controlled to implement homeostasis, rather than implementing in the neuron. A memristor based homeostatic neuron was presented in [83]. However, the designed neuron was not used

in a classification application and hence the performance of the neuron in a patter recognition task could not be evaluated.

In terms of system-level applications, in [95, 82, 92], the authors have demonstrated supervised and unsupervised learning platforms applied to classification tasks. However, they do not consider the device level imperfections of memristors. Another supervised learning system was presented in [75] that consider memristor switching mismatch and endurance degradation, but does not deal with device failure or process variations. A process variation immune unsupervised memristive neuromorphic system was presented in [71] delivering promising results. However, the authors assume the memristor switching window (HRS/LRS) of 10000, which is impractical considering fabricated devices [52]. In [37], more practical memristive devices were considered showing a high degree of resilience against process variations, array failure and switching mismatch. However, only the synapse and the training part of the unsupervised learning mechanism was carried out in hardware, while the homeostatic plasticity and lateral inhibition induced competitive learning was executed by a software simulation.

Chapter 3

Forming and Programming of Memristors

Disclosure

This chapter is an adapted version of the manuscripts published in [2, 3] with minor modifications.

3.1 Forming Circuit

As discussed previously, memristors provide a multitude of advantages in terms of being used as a memory element. However, design of memristor based logic is complicated by the electro-forming which is a one-time process to enable resistive switching. The voltage required for forming is usually higher than the voltage required for normal operation of memristors such as SET or RESET. For HfO_2 memristors that were fabricated by SUNY Polytechnic Institute, a forming voltage of $2.1V$ is required, whereas the nominal operating voltage of the MOSFET devices is $1.2V$ [10]. To account for the higher forming voltage, thick oxide based MOSFET (DGXFET) devices from IBM 65nm CMOS 10LPe PDK that operates at $3.3V$, is used in this work. The design of the forming circuit is shown in Fig. 3.1, which is adapted from [10], using the memristor model described in section 2.1.1 for

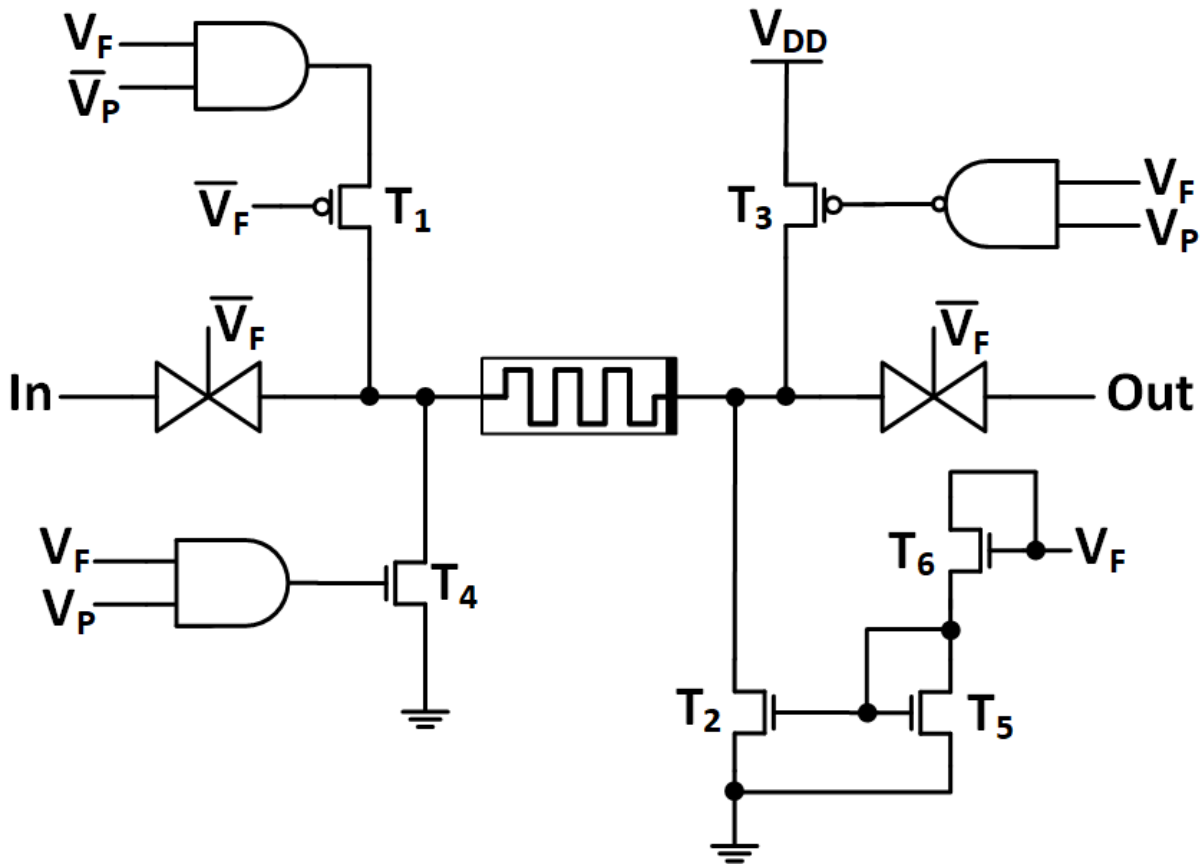


Figure 3.1: The design of the forming circuit incorporating the programming operation.

switching characteristics and [9] for capturing the electro-forming process. The figure omits the level shifter circuits that are required for converting the logic signals from $1.2V$ to $3.3V$.

Programming the memristive devices to an initial state is another crucial step in the specified system. Although analog state tuning is desirable, the large amount of circuit overhead required by this method may render it impractical. Therefore, in this work, we follow a different strategy, in which the memristor is programmed as LRS or HRS, and we rely on online learning to adjust the states at runtime. To avoid a downward shift in LRS during SET operation (going from HRS to LRS), current compliance is needed to establish a self-limiting process [44]. The current compliance is ensured by transistors $T2$, $T5$, and $T6$ in Fig. 3.1 and is shared during both forming and SET operations. This enables the integration of programming and forming steps, achieving a compact design. The control signals and operation of the circuit is best described by the truth table of table 3.1. The forming and programming circuit is turned off by setting $V_F = 0$, which allows the series transmission gates to connect the memristors to the learning circuit. For forming and SET operations, $V_F = 1, V_P = 0$, and the current compliance transistors are active for reasons explained above. For RESET operation, $V_F = 1, V_P = 1$ and the memristor goes from LRS to HRS.

3.2 CMOS/Memristor Programming

3.2.1 Programming Logic

In this work, a single input pin is used to program both the digital and memristive memory devices. A pin (*Bit_Stream*) is used as a serial input to program 'K' bits into the D Flip-Flops and 'N' bits into the memristors. The pin is connected to the input of a De-multiplexer, the selector of which determines which type of memory to program. This selector signal (*DONE_FF* in Fig. 3.2) is asserted when the programming into D Flip-Flops is completed. A digital counter counts the number of bits that are programmed into the D Flip-Flops, and when the count value reaches 'K', the De-mux selector signal *DONE_FF* goes from '0' to '1'. The change in level is converted into a pulse by a Moore-type finite state machine (FSM), that

Table 3.1: Truth table for forming and programming operation.

V_F	V_P	F
0	X	*learning*
1	0	$Forming/SET$
1	1	$RESET$

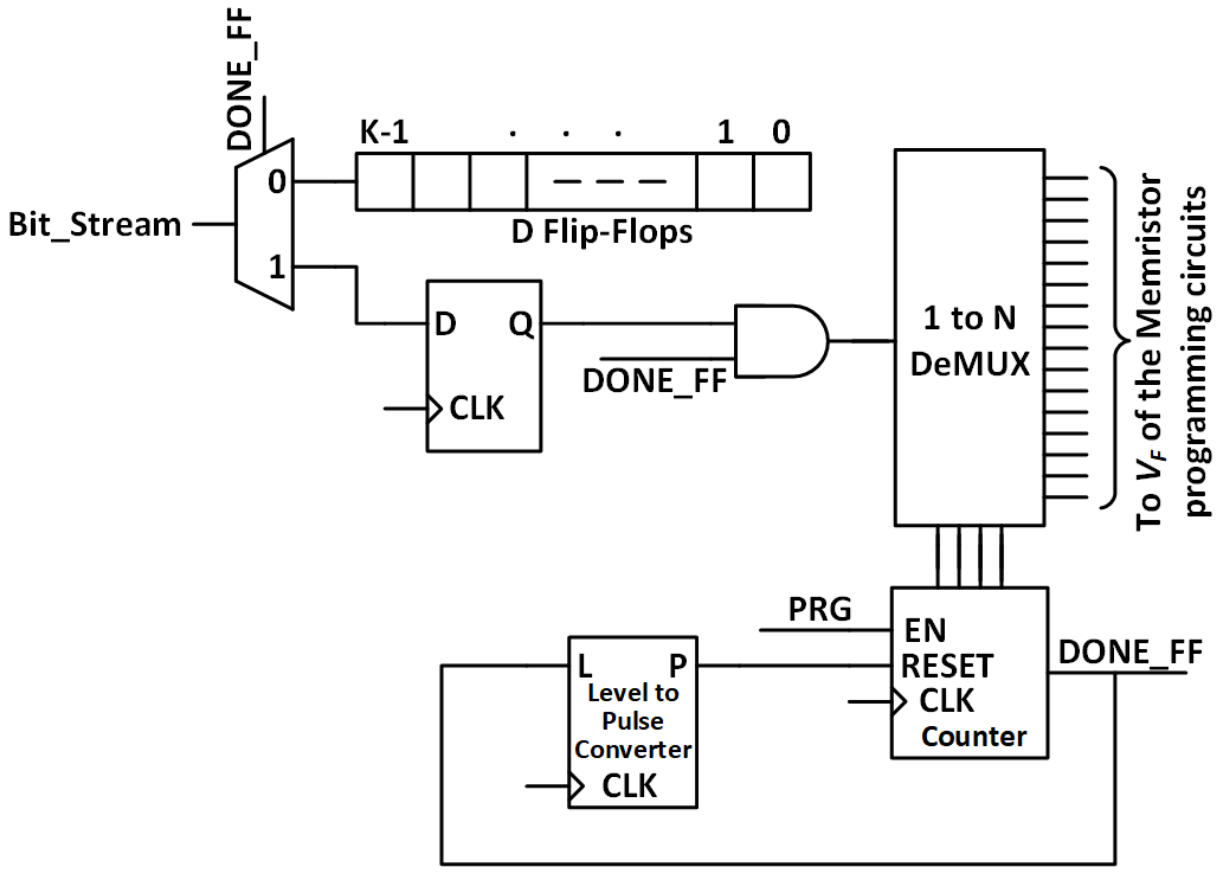


Figure 3.2: Control circuitry for memristor forming and simultaneous programming of CMOS/Memristor memory.

resets the counter to '0'. With the end of D Flip-Flop programming, memristor programming can begin, as the De-mux connects the *Bit_Stream* signal to the memristor programming path. A D Flip-Flop is used as a synchronizer between programming bits and counter values, as the input programming bits could have a phase difference. The synchronized programming bits are passed on to a 1-to-N De-mux via an AND gate that ensures that programming of memristors does not commence until digital D Flip-Flop programming is complete. The counter value is used as the selector of the 1-to-N De-mux, which allows the programming bit to be de-multiplexed on to the correct memristor. As the counter value increments, the programming bits of the *Bit_Stream* is passed on sequentially to all the connected memristors. If the programming bit is '1', the memristor is programmed from HRS to LRS. If the programming bit is '0', the memristor is kept at HRS.

Programming protocol begins with the forming of all memristors at the same time. A global forming pulse is provided to all the memristor programming circuits. V_F is $3.3V$ and V_P is kept at $0V$. Therefore, the forming path is activated and current flows through the transistor $T1$, the memristor and the transistor $T2$. Once the memristors are formed, they are at LRS. Then, a RESET pulse is applied to all the memristors. V_F remains at $3.3V$ and V_P is also asserted $3.3V$. As a result, the LRS to HRS programming path is now active and current flows through transistor $T3$, the memristor and transistor $T4$. Now all the memristors are are at HRS, so that they can be selectively reprogrammed to LRS using the control circuitry shown in Fig. 3.2. The SET direction (HRS to LRS) was chosen for selective reprogramming, as the transition time in SET direction is reported to be much lower than the RESET direction (LRS to HRS) [12]. If the memristors were formed first to LRS and then selectively reprogrammed in the RESET direction from LRS to HRS, longer pulses would be needed, which results in the reduction of clock frequency.

After completion of the initialization protocol, all the memristors are at HRS. Now, the scan-in process is commenced by applying a pre-configured bit sequence at the *Bit_Stream* pin. In this work, programming of 3 D Flip-Flops and 4 memristors are considered. The timing waveforms of all the relevant signals are shown in Fig. 3.3. The programming starts at time $t = 2.5\mu s$, with programming of the 3 D Flip-Flops. Then the 4 memristors M0, M1, M2, M4 are programmed according to the provided bit-stream. The bit-stream used is

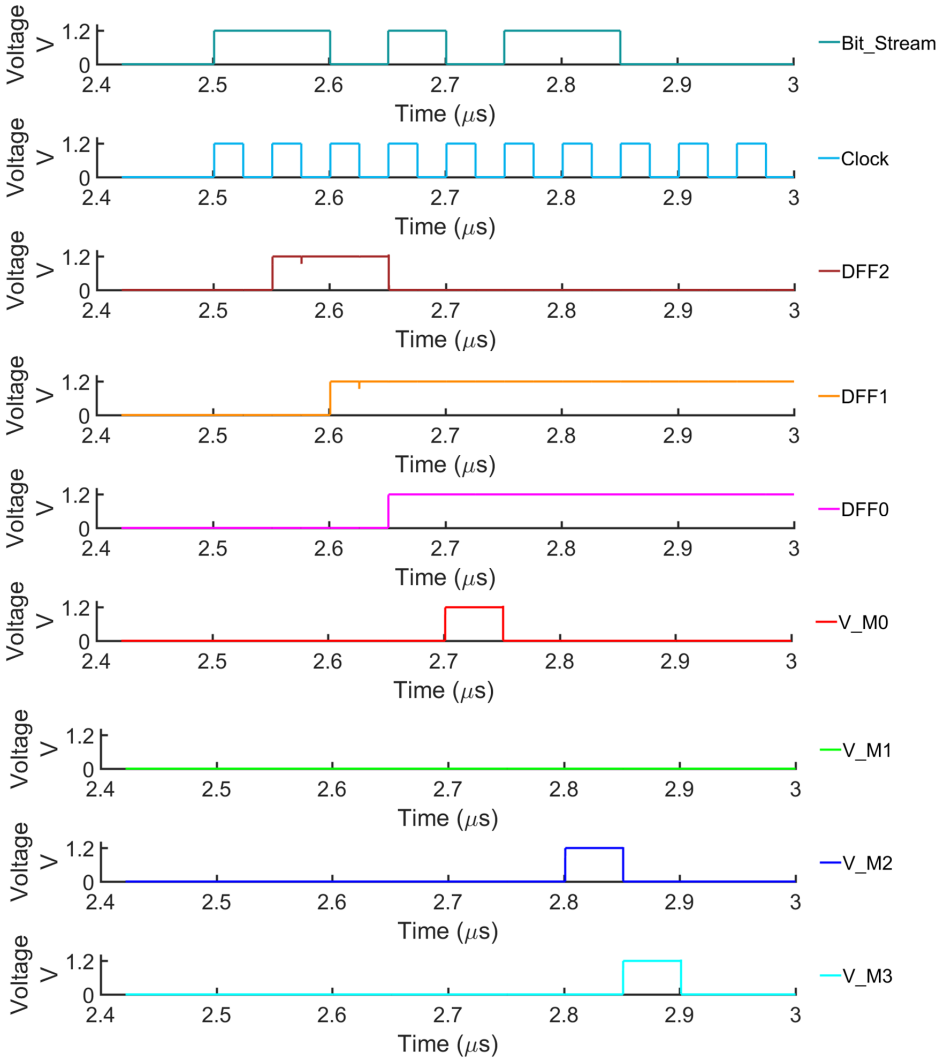


Figure 3.3: Timing diagram showing the waveforms for programming the D Flip-Flops and selectively programming memristors after initialization.

'1101011', meaning the D Flip-Flops are programmed to '011'. It is worth noting that the order is reversed as the first input bit shifts to the last Flip-Flop of the chain. The memristor programming bits '1011' is then programmed, as the memristors M0, M1, M2, M3 goes to LRS, HRS, LRS and LRS respectively. The states of the memristors are plotted in Fig. 3.4, showing the memristance values during pre-formed state, forming stage, initialization stage and selective reprogramming phase.

3.2.2 Application : MrDANNA Core Programming

A memristor based neuromorphic system called Memristive Dynamic Adaptive Neural Network Array (mrDANNA) was presented in [25], which was based on DANNA [32]. The mrDANNA was composed of spiking neuromorphic cores, acting as processing units. A mrDANNA core comprises a neuron at the heart of core and n synapses that are connected to it, as illustrated in Fig. 3.5. Each synapse has *delay* and *weight* attributes associated with it. D Flip-flops are used as synaptic delay elements in the synaptic buffer, whereas memristors store the weights of the synapses. A twin memristor configuration is used to store the weight, where the memristors are connected in opposite polarity, enabling the synapse to provide both positive and negative weight. By applying opposite polarity equal magnitude voltages at the memristors, synaptic buffers drive positive and negative currents through memristors M_p and M_n, respectively. As a result, a net current is produced, which could be positive or negative, depending on the memristance of the two memristors. The synaptic conductivity or weight is given by the following equation:

$$G_{eff} = \frac{1}{M_p} - \frac{1}{M_n} \tag{3.1}$$

MrDANNA system is based on a Spiking Neural Network (SNN) platform with both off-line and on-line learning using evolutionary optimization (EO) algorithm [77] and Spike-Timing-Dependent Plasticity (STDP) [84], respectively. The EO algorithm is used to generate an initial neural network with neurons and their connections (synapses). The synaptic delay and weights are also generated by EO. The on-line STDP algorithm then gradually pushes the synaptic weights to achieve the required tasks. The synaptic delay

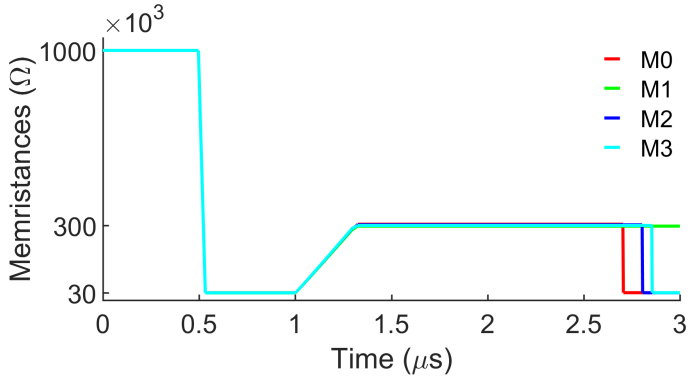


Figure 3.4: Change of memristance of the 4 memristors, starting from pre-formed stage, then being formed to LRS, then reset to HRS, and finally selectively programmed to LRS. The memristors M0, M2, M3 are set to LRS and M1 remains at HRS.

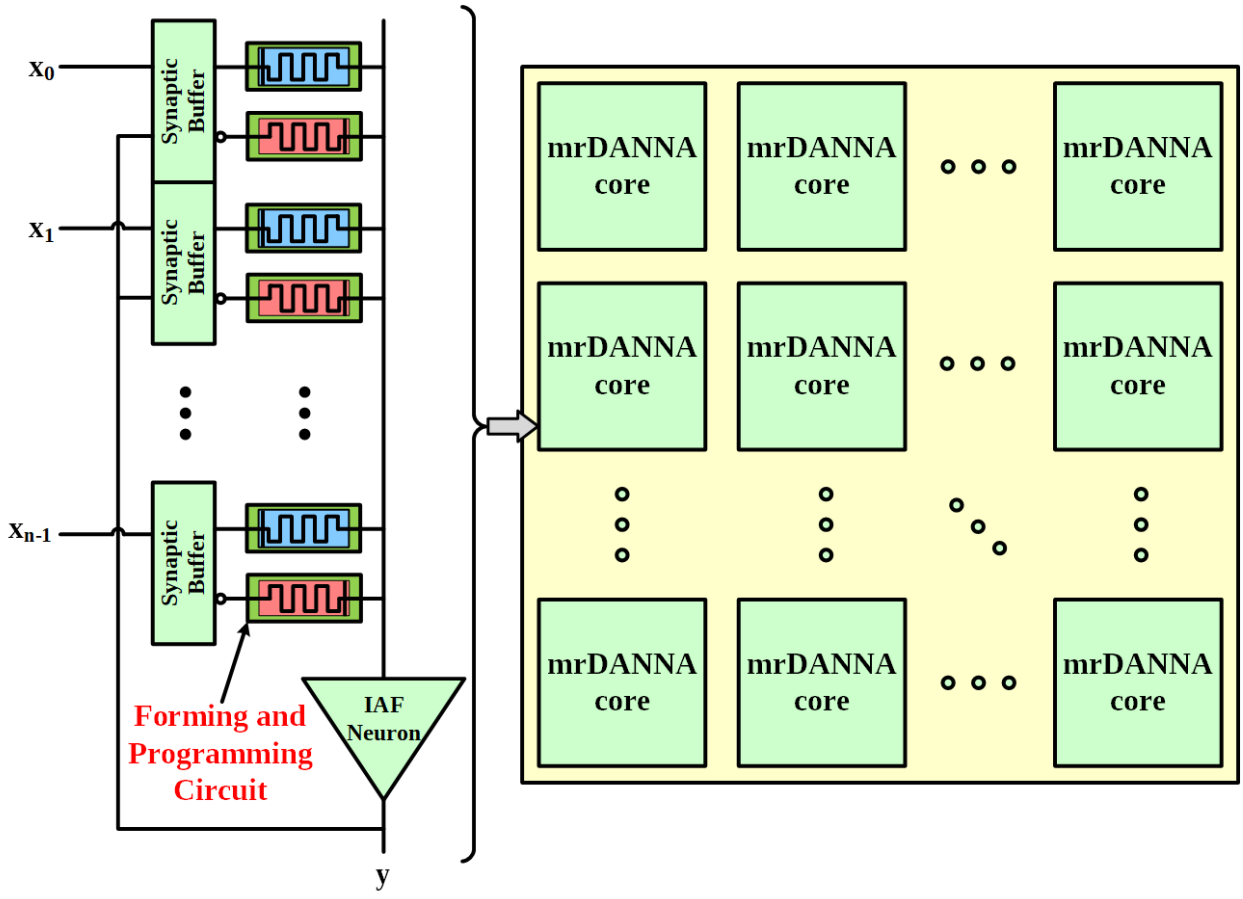


Figure 3.5: Memristive Dynamic Adaptive Neural Network Array (mrDANNA) architecture with zoomed in view of each core, including forming and programming circuits.

remains unchanged during online learning. To avoid analog memristance tuning during initialization of the network, the EO algorithm assigns one of the three synaptic weights, namely zero, minimum and maximum to each synapse. Zero weight is realized when both memristors M_p and M_n are programmed to LRS. Minimum weight is given by M_p and M_n being programmed to HRS and LRS, respectively. Conversely, maximum weight occurs when M_p and M_n are programmed to LRS and HRS, respectively.

In this work, we consider mrDANNA cores with a central neuron connected to 8 synapses. The maximum delay for each synapse is 7 cycles, which is set by the delay block. To configure the delay, 3 Flip-Flops are used per synapse. Hence, for each mrDANNA core, 24 D-Flip-Flops and 16 memristors are required and need to be programmed. To scan-in the required 24-bits for the Flip-Flops, a 6-bit counter is used. After programming the Flip-Flops, the counter is reset by the control circuit and memristor programming path is activated, so that the 16 memristors can be programmed according to the bit sequence provided. The programmed mrDANNA system with 3 sets of initial weight was implemented to classify the Iris dataset, reaching an accuracy of 95.7% with online learning [97].

3.2.3 Performance Analysis

To analyze the performance of the proposed forming and programming protocol, the power consumption of the designed circuit was measured, with the programming of 3 D Flip-Flops and 4 memristors, as discussed in section 3.2.1. The control circuitry shown in Fig. 3.2 is a digital module and hence dissipates very low amount of power, (only $1.46\mu W$ using a V_{DD} of $1.2V$). However, the forming circuitry operates at a higher V_{DD} of $3.3V$, due to the requirement of higher forming voltage. As a result, the power dissipation on the forming and programming circuitry is higher. Moreover, during forming and programming operation, the current driven through the memristor depends on the range of the HRS and LRS value of the memristor. Hence, the circuit simulation was performed using various LRS/HRS data of memristors reported in the literature [88]. The results are plotted in Fig. 3.6, showing the change in power dissipation for different memristor LRS/HRS values. With higher LRS/HRS value, the current through the memristor drops significantly, and accordingly there is a marked drop in power dissipation.

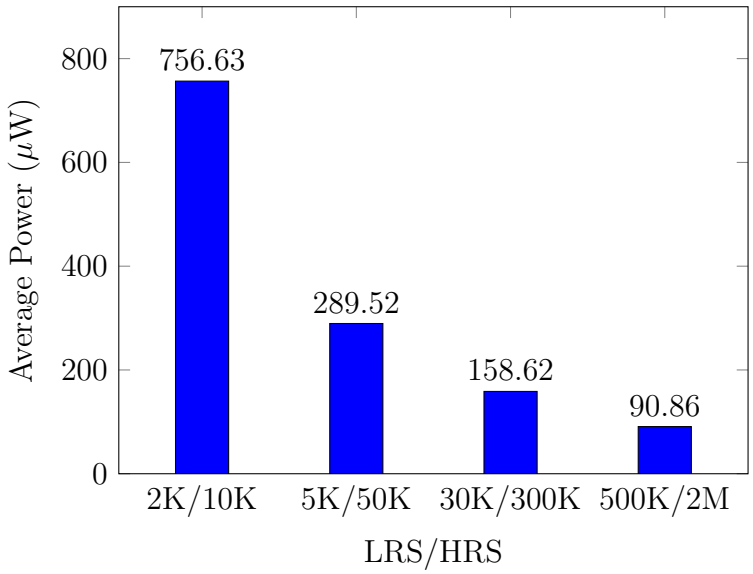


Figure 3.6: Average power dissipation for forming and programming circuit with varying memristor LRS/HRS values with supply voltage $V_{DD} = 3.3V$.

The high power dissipation during the forming operation is primarily due to the high forming voltage that must be provided for the memristors to form. The power supply voltage V_DD could be reduced below $3.3V$ if the forming voltages are lowered by process technology. The forming voltage parameter used in the memristor model was modified to measure the effect of decreasing forming voltage., by applying various supply voltages, the results of which is shown in Fig. 3.7. An order of magnitude reduction is observed when the power supply voltage is decreased from $3.3V$ to $1.2V$. These results underlines the need for a push towards improvement of memristor processing technology, aiming at increased device memristance and lower forming voltages.

3.3 Multi Level Memristor Programming

The forming and programming scheme described thus far is applicable for binary state programming. To enable multi-level programming of memristors, two interchangeable scan register is utilized to separate and parallelize the programming and scanning of data bits.

3.3.1 Programming Scheme

The approach taken here is demonstrated using a transition metal oxide (TMO) memristor, although it is equally applicable to other memristor based multi-level memory devices. A pulse width modulated waveform is applied to the memristor to program the memristor to a specific level. The programming direction is assumed to be the RESET direction from LRS to HRS, as the switching time for SET direction (HRS to LRS) [12] is small and hence attaining in-between states is challenging. According to the memristor model described in section 2.1.1, when the applied voltage across the memristor exceeds the switching threshold of the device, the memristance starts to rise. When it is close to HRS, the rate of change decreases, which is shown in Fig. 3.8.

Generally, the greater the width of the applied pulse, the greater the memristance change and vice versa. Using these properties, multiple memristance states can be obtained by applying pulses of varying widths. Here, we consider that each memristor has four discrete levels, and each device stores 2 bits. We modulate the width of the programming pulses so

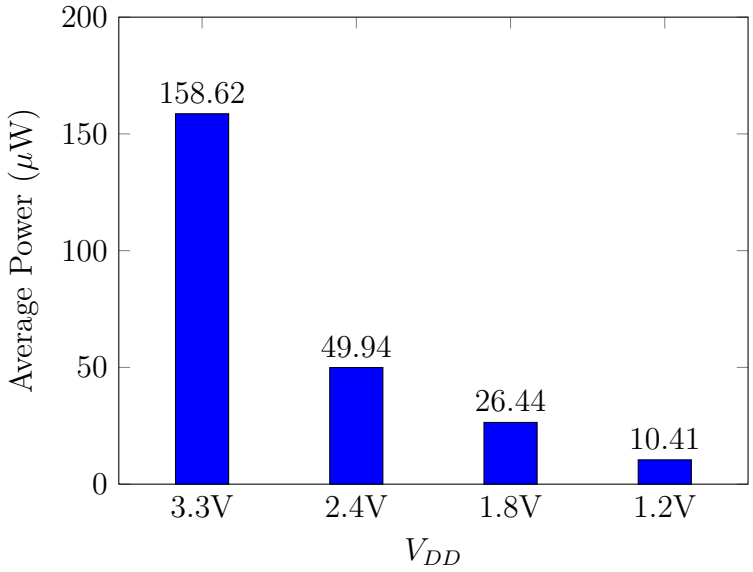


Figure 3.7: Average power dissipation for forming and programming circuit with varying supply voltages for LRS/HRS of 30K/300K.

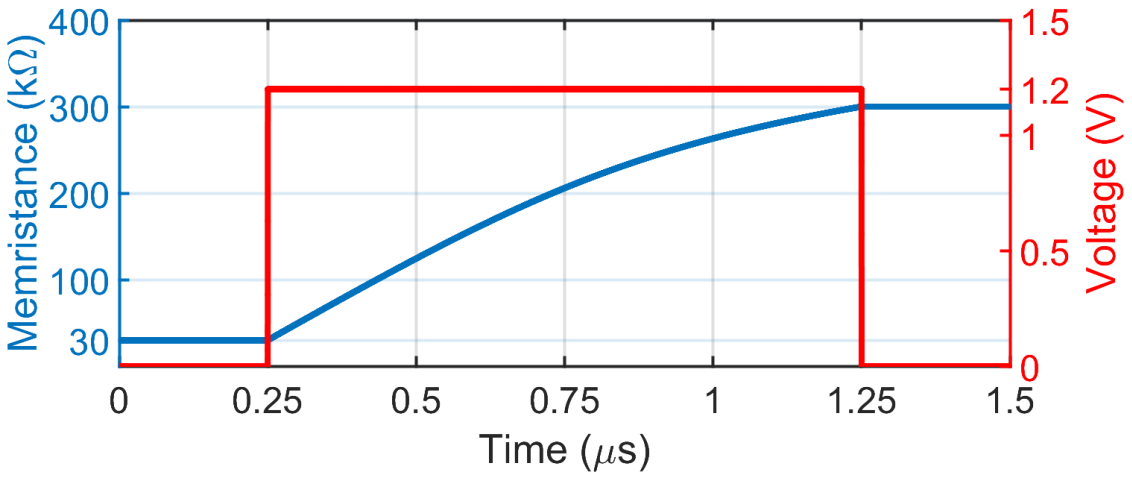


Figure 3.8: Change of memristance as a voltage is applied across the memristor terminals. At first, the memristor is in LRS. When the applied voltage exceeds the switching threshold, the memristance starts to rise. Before reaching HRS, it starts to plateau and is finally set to HRS.

that 0, 1, 2 and 3 unit-width pulses are equivalent to programming bits '00', '01', '10' and '11', respectively. The data bits are stored in two scan register. The motivation behind using two independent scan registers is to enable the decoupling of memristor programming and scanning in of programming bits. While one register scans bits of data, another scan register can program the memristor at that same time. When scanning the data bits into one register is completed, the programming from this scan register can start and the other scan register, which was holding the data bits to program the memristors, can now start scanning in new programming data bits. This can be achieved by programming the memristor within the time required to scan-in the data bits in scan registers. In this work, the scan register length is 4 bits and hence 4 clock period is required to scan-in the data. The maximum pulse width requited to switch memristor from LRS to HRS is 3 units, which corresponds to 3 clock periods, each unit width being equal to a clock period. Therefore, a programming/scan-in cycle is defined to be 4 clock periods long so that a scan register is able to scan-in the programming bits, while simultaneously, the other scan register (which was holding the program bits to program the memristor) can start to scan-in new programming data.

The design of the programming scheme is illustrated in Fig. 3.9. Here, each scan register contains 4 bits of data, with 2 bits to program each memristor. Hence, 2 memristors can be programmed at the same time. Once scan-in to a scan register (REG1) is done, it goes into programming mode and at the same time, the other scan register (REG0) begins scanning-in new data bits. The data bits of the scan register that is currently in programming mode (REG1), are provided to the pulse generator block (a Digital-to-Analog Converter) through a multiplexer which selects this specific scan register (REG1). The pulse generator converts the digital data bits into a pulse width modulated (PWM) waveform. In this example, the carryout signal produced by a 2-bit accumulator register is used as the PWM waveform. The two PWM signals generated by the two pulse generators are transmitted to the decoder, to which the memristors are connected. A counter that counts up in each programming cycle decides the memristor that will be programmed. Therefore, in each programming cycle, 2 memristors connected to 2 decoders are programmed simultaneously. After one programming cycle ends, another programming cycle begins immediately and the programming data bits

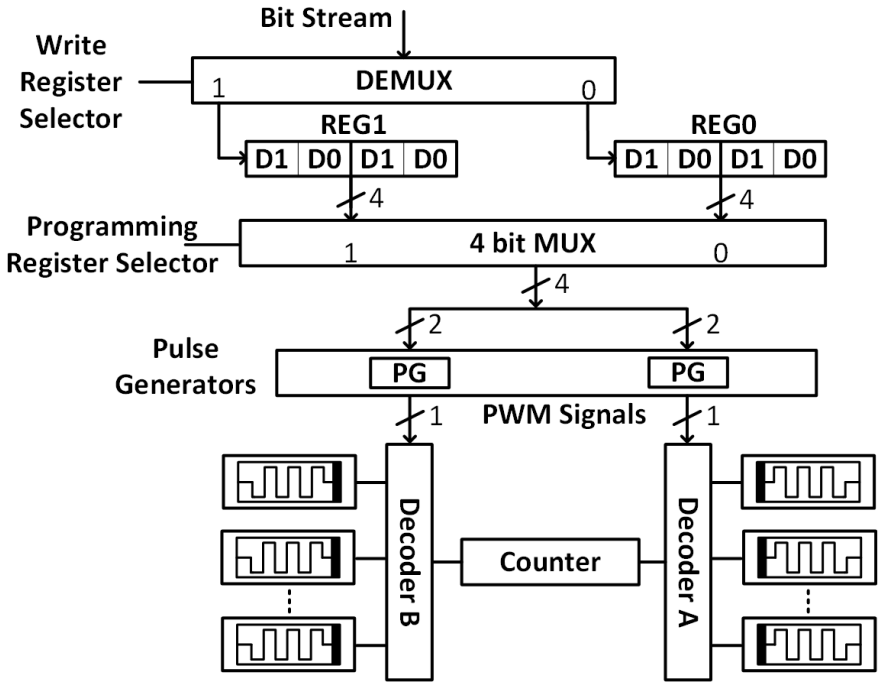


Figure 3.9: Block diagram of the proposed programming scheme.

are read from the other scan register (REG0). This process continues in a loop until all the memristors in the system have been programmed.

3.3.2 Reading Scheme

The design of the reading scheme is shown in Fig. 3.10. It reuses the scan registers to scan-out the data stored in the memristors. It follows a similar process to the programming scheme and loads the scan registers and scans-out the data bits simultaneously in one read cycle. The read cycle is again 4 clock periods, similar to the programming scheme. To read the data from memristors, they are connected in voltage divider configuration to the resistors, as shown in Fig. 3.11. The decoders previously used in the programming scheme, now apply positive rail voltage at the memristor terminal. The resistance value is selected so that the voltage across the memristor does not exceed the switching threshold. The output voltages from voltage dividers are sampled and provided to the Analog to Digital (ADC) converters, which generates 2-digital bits per memristor. Here, a flash ADC is used to produce the digital bits because the resistive ladder network used in the flash ADC can be configured to account for the non-linear memristance change shown in Fig. 3.8. The data bits produced by the ADCs are loaded into a scan register (REG0), and after the first read cycle is completed, these bits are scanned-out. Simultaneously, the other scan register (REG1) starts loading data bits from another pair of memristors in the same way. After another read cycle, the scan register REG1 (which is now loaded with data bits) begins scanning-out, while REG0 now loads the data bits from the ADCs. The counter used in the programming scheme is reused to go through all the memristors to read the data bits stored as memristance values.

3.3.3 Erasing Scheme

For erasing data from memristors, a gloabal ERASE signal (Fig. 3.11) is used. By pulsing the signal, all the memristors are set to LRS ('00' state) at the same time. Erase procedure is very fast, as the switching time from HRS to LRS is very low [12]. Erasing does not require accessing each memristor individually, and hence it is not discussed further in this work.

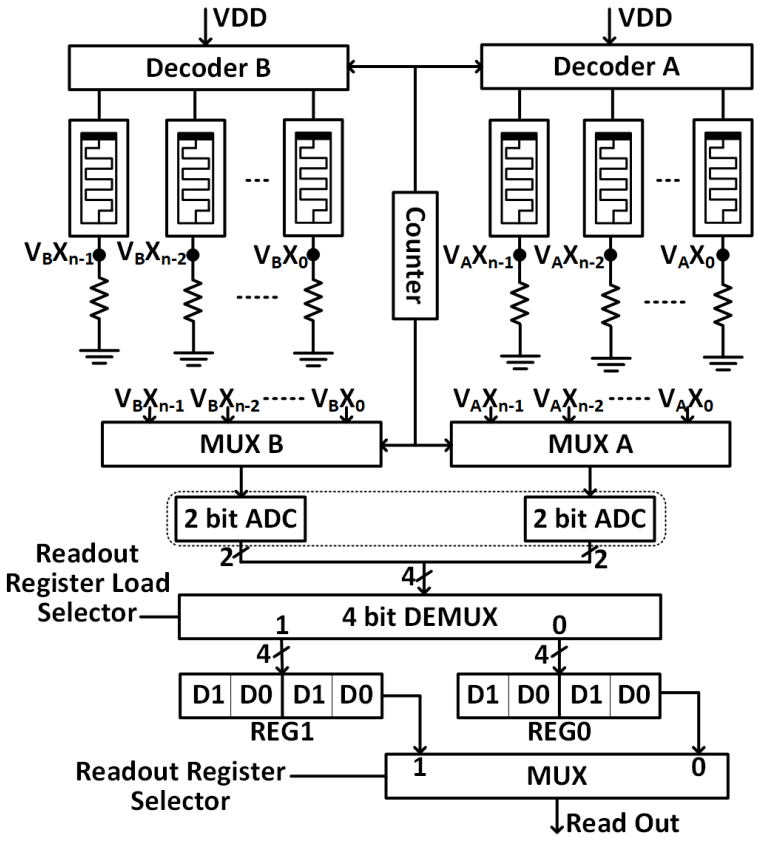


Figure 3.10: Block diagram of the proposed reading scheme.

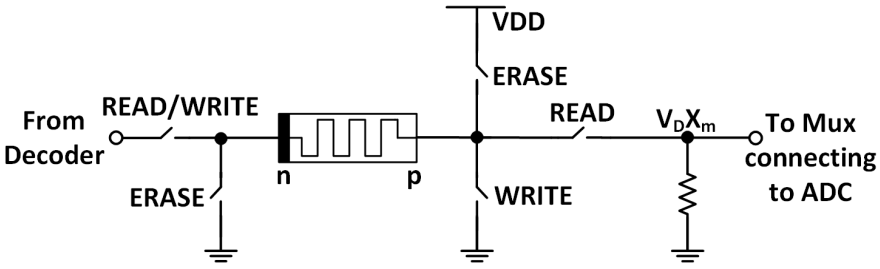


Figure 3.11: During the programming operation, the memristor is driven by the decoder and directly connected to ground. During read operation, the memristor is still driven by the decoder, but it is connected in voltage divider configuration, in series with a resistor. For the erase operation, voltage is driven in opposite direction to the programming operation.

It is worth noting that the access schemes presented here are not specific to the DAC or ADC used in this work. The scan register based access protocol can be used with any pulse based read/write schemes with different DACs or ADCs.

3.3.4 Simulations

The proposed access scheme of programming the memristors and and reading back the programmed data was substantiated with a Cadence Spectre transistor level simulation using IBM 65nm CMOS10LPe PDK. The HfO_2 memristor model described in section 2.1.1 was used for simulation, with the device parameters listed in [24].

Scan-In Programming

The programming simulation considers the programming for 8 memristors, each being programmed to 4 resistance states, storing 2 bits each. Hence, a 16-bit data '1011_1000_1100_0101' is taken as an example. Two memristors are simultaneously programmed, and hence each block of data represents the programming bits for those two memristors. As the bitstream is scanned-in serially, each block should be read from right to left to find out which state a memristor is being programmed to. The waveforms for the programming scheme is presented in Fig. 3.12, which should be read in conjunction with the block diagram of Fig. 3.9. The system clock signal is divided 8 times and the resulting signal is shown as $CLK/8$, which is used as *Write Register Selector*. The bitstream of the programming data is represented by the Bit_Stream signal. The register states of REG1 and REG0 are also shown. The dashed line on those waveforms means the register is being loaded at that time. The pulse width modulated signals $VMA0$, $VMA1$, $VMA2$, $VMA3$ and $VMB0$, $VMB1$, $VMB2$, $VMB3$ are the signals that are applied to the memristors by decoders A and B, respectively.

The loading of data bits into REG1 begins when the $CLK/8$ goes to '1' and it continues until it changes to '0'. The scan register REG1 now contains '1101', and these data bits can now be programmed into the memristors selected by the counter. The programming from REG1 occurs while the $CLK/8$ stay at '0'. As the counter value($COUNT$) is at '0', the

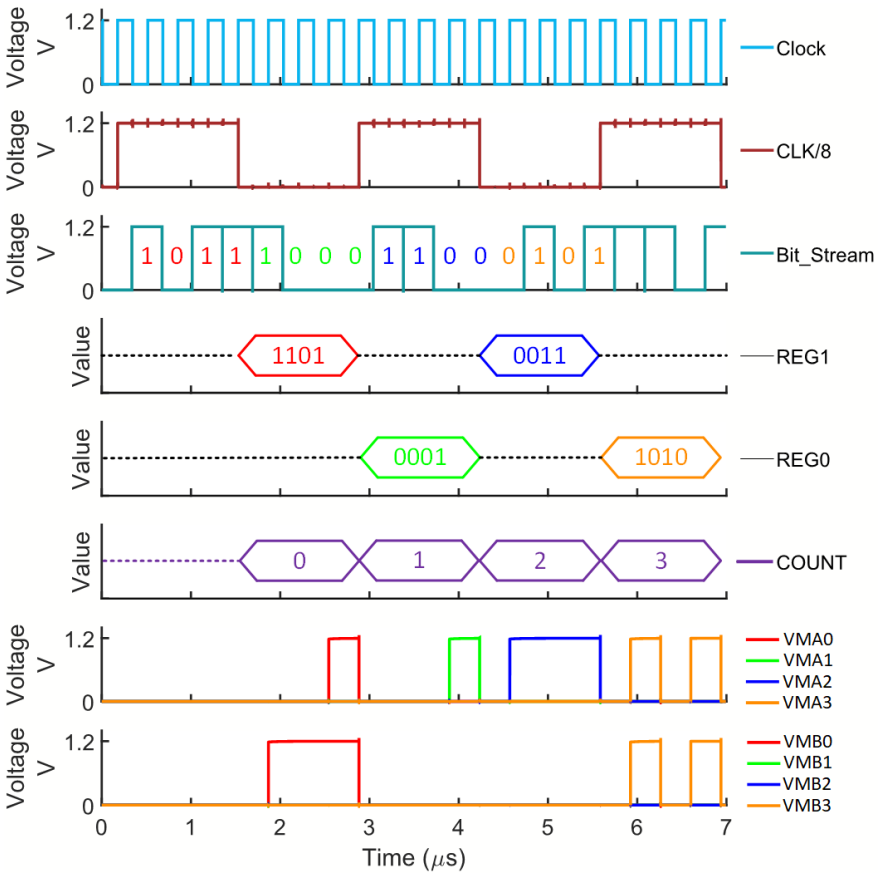


Figure 3.12: Waveforms that are applied during the programming scheme along with the states of the scan registers to program the memristors.

two memristors connected to the 0th pin of the decoders A and B are now programmed to the states indicated by the register value, which is '01' and '11'. As a result, a 1 cycle wide pulse and a 3 cycle wide is generated ($VMA0$ and $VMB0$, respectively) and applied to the memristors to program them to these bit states.

While REG0 contents were being programmed to the memristors, REG1 was scanning-in the next four data bits('0001') in parallel, while $CLK/8$ stays at '0'. When it goes to '1' programming begins from REG0 in a similar way. Simultaneously, REG1 scans-in new data bits. The cycle continues until all the memristors in the system have been programmed to the desired states. The memristance of the 8 programmed memristors connected to the two decoders are shown in Fig. 3.13.

Scan-Out Reading

The simulation of the reading scheme was performed with the memristors programmed to the states as shown in the programming scheme simulation. Waveforms of the signals used for reading the data bits from the memristors are shown in Fig. 3.14. CLK, $CLK/8$, $REG1$ and $REG0$ signals represent the same signals as described in programming scheme. However, the dashed lines on $REG1$ and $REG0$ signals now means that the data bits are being shifted out from the scan registers. For reading the data bits, V_{DD} is applied selectively via the decoders to the memristors which are in voltage divider configuration. The signals $VM0$, $VM1$, $VM2$, and $VM3$ represents those signals from the decoders, with the numbers corresponding to that pin of the decoders, selected by the counter value. The data stream *ReadOut* represents the previously programmed memristor data that is shifted out from REG1 and REG0.

The first read cycle occurs when $CLK/8$ goes to '1', with the counter value being '0'. The memristance value of the memristors connected to the 0th pin of the decoders A and B (i.e., MA0 and MB0) are read. From Fig. 3.13, we can see that MA0 and MB0 are at $150k\Omega$ and $300k\Omega$, respectively, which correspond to digital bits '01' and '11', respectively. In the read cycle, the scan register REG0 is loaded by the ADC and contains '1101', which is shifted out in the next read cycle. Simultaneously to this shifting out, the scan register

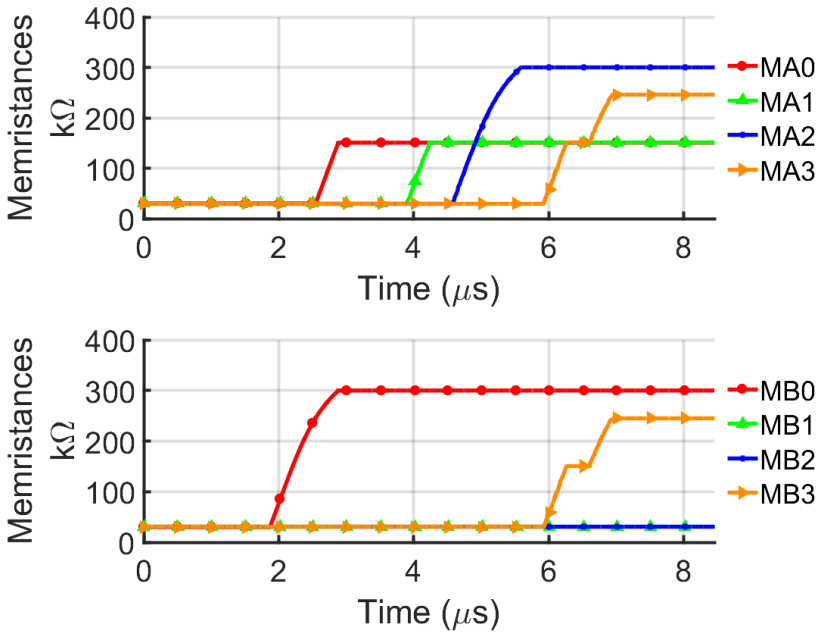


Figure 3.13: Change of memristance of the 8 memristors with time, connected to the decoders A and B. The same colored plots indicate the memristance of the memristors that are programmed simultaneously at the same programming cycle.

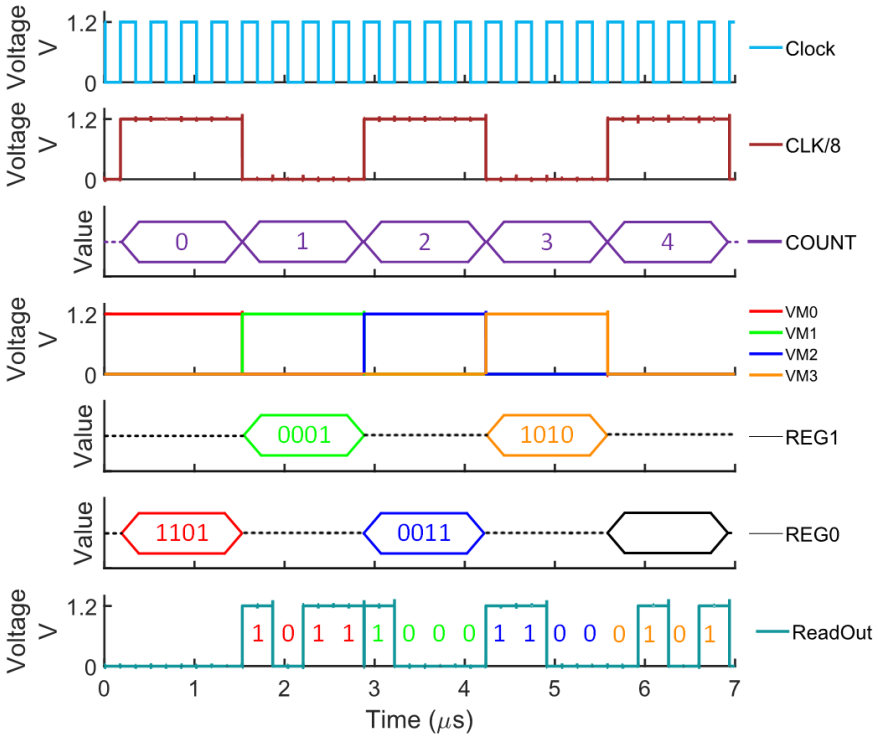


Figure 3.14: Waveforms that are applied during the reading scheme along with the states of the scan registers to read the data from memristors.

REG1 loads the next set of data from the memristors. This continues cyclically until all of the memristor data is read out.

$\overline{CLK/8}$ and $CLK/8$ signals are used as *Readout Register Selector* and *Readout Register Load Selector*, respectively. As inverted signals are used as selectors for the two different purposes, the parallelization of reading and scanning out of data bits is ensured. Looking at the *ReadOut* signal, it can be seen that it is the same data stream as the *Bit_Stream* used in programming, which confirms the functionality of both programming and reading schemes.

3.3.5 Performance Analysis

Scalability

The proposed scheme takes advantage of the parallelism of scanning in/out and programming/reading of data bits and optimizes the size of scan register. The scan register sizing is determined by the number of states that the memristors can be programmed to, not on the number of memristors to be programmed. The number of memristors to be programmed could be arbitrarily high, depending only on the size of the decoder. A scalability study of the proposed access scheme for enabling higher number of memristance states was performed and the number of required flip-flops and gates was found to be:

$$\text{No. of Flip-flops} = 2^{x+1} + (x+1)^2 + 4$$
$$\text{No. of eqv. Gates} = 2^{x-2}(14x + 18) + x(6x + 42) + 27$$
$$\text{where, } x = \log_2(\text{No. of levels})$$

Here, the equivalent gate count is taken as 1/4th the number of transistors, which can also be termed as equivalent 2-input NAND gate count. The scalability analysis was performed by calculating the number of registers and transistors needed for each blocks of Figs. 3.9 and 3.10 for different x values.

Energy Consumption

The energy requirement of the designed access scheme was measured, the result of which is listed in Table 3.2. Most of the energy is consumed by the ADC/DAC, which are not the focus here, with only 15% of the total energy being expended by the proposed programming and reading scheme. A multi-state programming scheme was presented in [91], which lists the write energy requirement to be $24.41 pJ/bit$, compared to which, the proposed design is found to be energy efficient.

Table 3.2: Energy consumption of the proposed multi-bit memristor programming and reading scheme

Operation	Access Energy (pJ/bit)	DAC/ADC Energy (pJ/bit)	Total Energy (pJ/bit)
Write Scheme	0.11	1.80	1.91
Read Scheme	0.40	1.17	1.57

Chapter 4

Twin Memristor Synapse

Disclosure

This chapter is an adapted version of the manuscript published in [5] with minor modifications.

4.1 Structure and Operation

A twin memristor synapse capable of realizing STDP learning is proposed here that is derived from [23]. It consists of two memristors that are connected to each other with their polarities reversed as shown in Fig. 4.1. The synapse comprising the twin memristor connects a pre-neuron to a post-neuron. The pair of terminals connected to the pre-neuron are driven separately by the neuron, whereas the other pair of terminals are shorted together to form a single *postsynaptic* node that connects to the post-neuron.

There are two operating phase of the synapse: *accumulation* and *learning*. The neuron includes a control circuitry which applies the required voltage levels to drive the memristors during each phase.

Accumulation

During accumulation, with a firing event at the pre-neuron, it drives the twin memristor nodes V_p and V_n to positive and negative rails. The postsynaptic node is held at a virtual

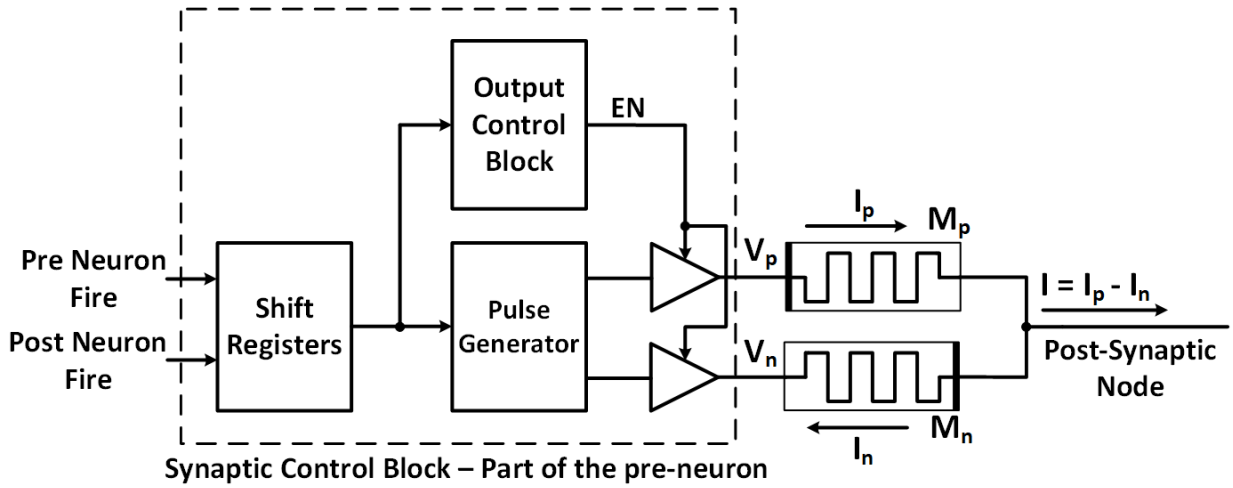


Figure 4.1: The proposed twin memristor synapse along with the block diagram of the synaptic control block, which a part of the pre-neuron. The synaptic control block houses shift registers for storing timing information of the firing events, a pulse generator to apply appropriate pulse width modulated waveforms, and an output control unit to disable outputs when not required.

ground by the integrator op-amp of the post-neuron [23]. A positive current is driven through the memristor M_p, while a negative current flows through M_n. The net current at the postsynaptic node can be positive, negative or zero, given the memristance states of the twin memristor. Thus, the conductance of the twin memristor synapse could be positive, negative or zero, without the need of extra circuit overhead. The effective conductance (or weight) of the synapse is given by the following equation:

$$G_{eff} = \frac{1}{M_p} - \frac{1}{M_n} \tag{4.1}$$

It is worth noting that the positive current passing through the synapse will cause the accumulation of charge in the post-neuron, while the negative current will cause charges to dissipate from it.

Learning

During the learning phase, the synaptic weight is adjusted according to the STDP learning rule. In this phase, the pre-neuron applies the same voltage on both the twin memristor terminals, making $V_p = V_n$. Now, a voltage is applied at the postsynaptic node by the post-neuron. The polarity of the applied voltage at these nodes depends on the timing of the firing events. When a pre-neuron fires before the post-neuron, the synapse should be potentiated i.e, the synaptic weight must increase. To achieve this, the pre-neuron drives the twin memristor terminals to the negative rail, while the post-neuron drive the postsynaptic node to the positive rail. As a result, the voltage difference across both the memristors exceeds the switching threshold. As the memristors are connected with opposite polarity, the memristance of M_p decreases, whereas the memristance of M_n increases. The end result is a net increase in the conductance of the synapse. The conductance after potentiation is given by the following equation:

$$G_{eff,pot} = \frac{1}{M_p - \Delta M_{p,pot}} - \frac{1}{M_n + \Delta M_{n,pot}} \tag{4.2}$$

When a pre-neuron fires after the post-neuron, depression occurs, where synaptic weight decreases. Now all the voltages are reversed compared to the potentiation condition. Hence,

the memristance of M_p increases, whereas the memristance of M_n decreases. Thus, the synaptic weight of the twin memristor synapse decreases. The conductance after depression is given by the following equation:

$$G_{eff,dep} = \frac{1}{M_p + \Delta M_{p,dep}} - \frac{1}{M_n - \Delta M_{n,dep}}. \tag{4.3}$$

The amount of change in memristance (ΔM) for both potentiation and depression depends on the temporal difference between the pre and post-neuron fires. The closer the fires are, the greater the change in memristance and vice versa.

4.2 STDP in Twin Memristor Synapse

The proposed mixed signal neuromorphic system combines analog implementation of synapses and neurons with digital firing events and inter-neuron communication. The twin memristor synapse itself is analog in nature, as the memristance state of the memristor is a continuous variable between HRS and LRS. The integrate and fire neuron [23] used in this work also accumulates analog voltage. However, the accumulated voltage, upon crossing the neuron threshold produces a digital firing event which is communicated to the following neuron(s). Moreover, the control circuitry that drive required voltages to the memristor terminals are also digital. The time for which the pulses are applied are modulated according to the STDP learning rule. If the time between pre and post-neuron fire is long, the voltage across the memristor is provided for a short period of time, causing weaker potentiation and depression. As the pre-neuron fire occurs closer to the post-neuron fire, voltage across the memristor is provided for longer, resulting in stronger potentiation or depression.

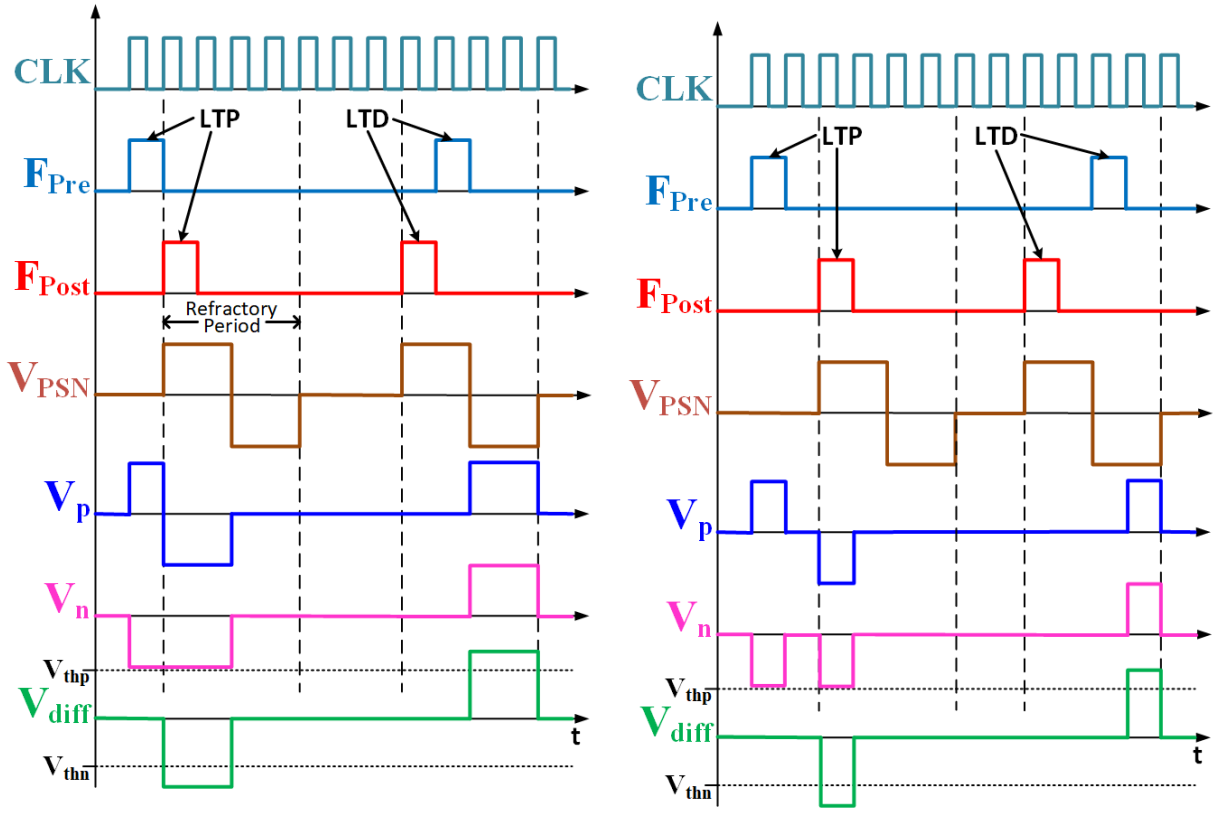
The synaptic control block of the neurons ensures the correct functioning of the circuit, the schematic diagram of which is shown in Fig. 4.1. To track the timing of the firing events of the pre and post-neuron, a shift register is used. It also provides the stored firing information to the pulse generator module. The pulse generator block is a combinational logic circuit that generates the required voltage pulses with appropriate width to be applied to the twin memristor terminals. The output control block generates output enable signal

for the neuron, deciding when the voltages should be applied to the memristors. When the synapse is neither in *accumulation* nor in *learning* phase, the voltage to the memristor is turned off by the output control block, which is considered the *idle* state of the synapse. During the *idle* state, the memristor terminals V_p and V_n are left floating so that there is no unintentional learning or accumulation.

In addition to accumulating and firing, the post-neuron also supplies its fire back to the pre-neurons and drive postsynaptic nodes during the refractory period, which coincides with the learning phase of the synapse. While in the refractory period, the path for post-neurons to accumulate is cut off, so no charge is accumulated. Instead, it applies a digital signal to the postsynaptic node, which remains positive for half of the refractory period and negative for the other half of the refractory period. The first half of the refractory period is used for potentiation, and the second half is used for depression. In this work, the designed circuits were simulated using Cadence Spectre, using the memristor model discussed in section 2.1.1 for the twin memristor synapse and the Integrate and Fire neuron shown in [23].

The operating principle of the synaptic control block is explained here with an example. We assume a system with the ability to perform STDP with 2-clock cycle tracking. In other words, in this system, the shift register of the synaptic control block can track the timing difference of pre and post-neuron fires of up to 2 clock cycles. Hence, there are 4 different scenarios of firing events: (1) The post-neuron fires just after pre-neuron, (2) The post-neuron fires after 1 clock cycle delay of pre-neuron fire, (3) The post-neuron fires just before the pre-neuron, and lastly (4) The post-neuron fires before the pre-neuron with 1 clock cycle gap between them. Scenarios 1 and 2 describe the potentiation condition, whereas 3 and 4 depict depression. However, for scenarios 1 and 3, both the potentiation and depression, respectively, will be stronger whereas for 2 and 4, they will be weaker, in accordance with the STDP rule. The scenarios of 1,3 and 2,4 are illustrated by figures 4.2a and 4.2b respectively.

When there is a pre-neuron fire, *accumulation* phase starts and V_p, V_n goes to postive and negative rail, respectively. This results in a net current flow into the postsynaptic node. As the postsynaptic node is held at a virtual ground, the voltage across each memristor does not go above the switching threshold and hence there is no change in memristance during this phase. The net current at the postsynaptic node is accumulated in the post-neuron



(a) Pre and post-neuron fires in consecutive cycles

(b) Pre and post-neuron fires with one cycle gap

Figure 4.2: Waveforms showing LTP/LTD with different cycle gaps. F_{Pre} and F_{Post} signals are the firing events from pre and post-neuron, respectively, V_{PSN} is the voltage applied at the postsynaptic node by the post-neuron, V_p and V_n are the voltages at memristors M_p and M_n, applied by the synaptic control block, V_{diff} is the voltage across the memristors during the learning phase only. The twin memristors are shorted at both ends during the learning phase, and hence V_{diff} is across both the memristors.

and if the accumulated voltage crosses the neuron threshold, it fires. With the fire, the *learning* phase starts and the post-neuron drives the postsynaptic node (V_{PSN}) to positive and negative rail for 2 cycles each. These 4 cycles also make up the refractory period, during which the accumulation path of the post-neuron is also cut off. On the pre-neuron side, the memristor terminals V_p and V_n are at equal voltages, which is shown in Fig. 4.2. For the scenarios 1 and 2, which are potentiation conditions, V_p and V_n are at negative rail with the voltage being applied for different period of time. The voltage across the memristor exceeds the threshold in such a way that the conductance of the synapse rises following equation 4.2. When the pre and post-neuron firing events are closer in time, the voltage across the memristor will exceed the threshold for a longer period of time, causing a higher reduction in memristance (scenario 2), and therefore higher increase in conductance (synaptic weight). The scenarios 3 and 4 are potentiation conditions and V_p and V_n are driven to positive rail in the later 2 cycles of the *learning* phase. Hence, the opposite scenario occurs and the conductance decreases. The digital logic circuit used in the pulse generator block for the 2-cycle STDP approach implements the following boolean functions for V_p and V_n,

$$V_p = \overline{F_{Post} \cdot (F_{Pre\_t1} + F_{Pre\_t2}) + F_{Post\_t1} \cdot F_{Pre\_t2}}, \tag{4.4}$$

$$V_n = F_{Post\_t2} \cdot F_{Pre\_t1} + F_{Post\_t3} \cdot (F_{Pre\_t1} + F_{Pre\_t2}), \tag{4.5}$$

where F_{Pre_t1}, F_{Post_t1}, F_{Pre_t2}, F_{Post_t2}, and F_{Pre_t3}, F_{Post_t3} represents the 1, 2, and 3 cycle delayed signals of F_{Pre} (pre-neuron fire) and F_{Post} (post-neuron fire), respectively. These delayed version of the timing signals are generated by the shift register block shown in Fig. 4.1.

The pulse generator output is turned off by the output control block during the *idle* state. It is implemented by another digital logic circuit that realizes the following function:

$$EN = \overline{V_p} + V_n + F_{Pre} \cdot \overline{Refrac}, \tag{4.6}$$

where $Refrac$ is the signal for refractory period.

By using similar method, different versions of synapses can be realized, with each version capable of tracking different number of cycles before and after a post-neuron fire. The example discussed in Fig. 4.2 is defined as 2-cycle STDP.

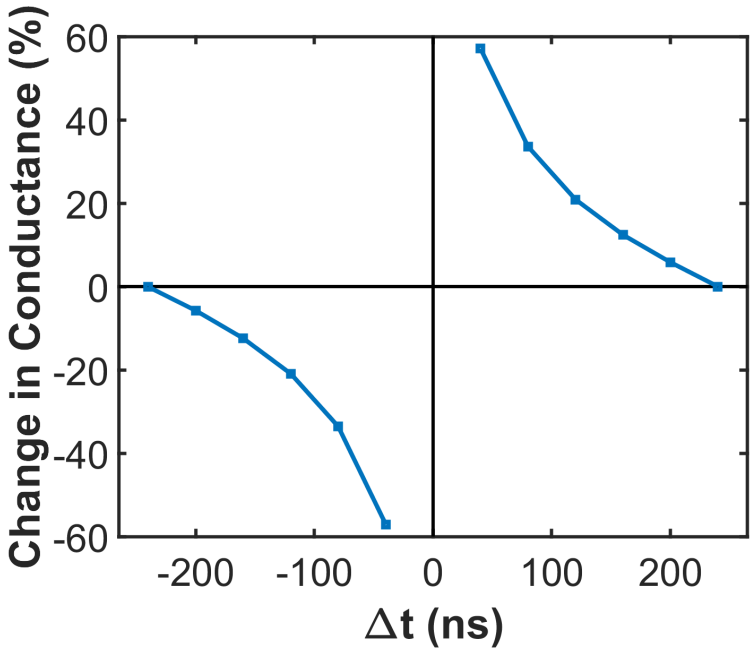
4.3 Design Parameters and STDP Performance

This section evaluates the effect of design parameters and memristor device properties on the synaptic weight change and STDP behavior of the twin memristor synapse. A 5-cycle tracking STDP architecture is considered for the analysis. The memristor model used for simulation was described in section 2.1.1, and the associated parameters are listed here as follows: $LRS = 5k\Omega$, $HRS = 50k\Omega$, $V_{tp} = 0.75$V, $V_{tn} = -0.75$V, $t_{swp} = t_{swn} = 1\mu s$. The synaptic conductance change was measured by normalizing against the maximum conductance of the synapse, which is, $G_{max} = \frac{1}{LRS} - \frac{1}{HRS}$. The twin memristor synapses was characterized with a clock frequency of 25 MHz and resulting conductance change in percentage of the maximum conductance is plotted in Fig. 4.3a. The curve maintains the anti-symmetric STDP shape explained in 2.2. The change in conductance increases non-linearly with lower temporal difference between the fires.

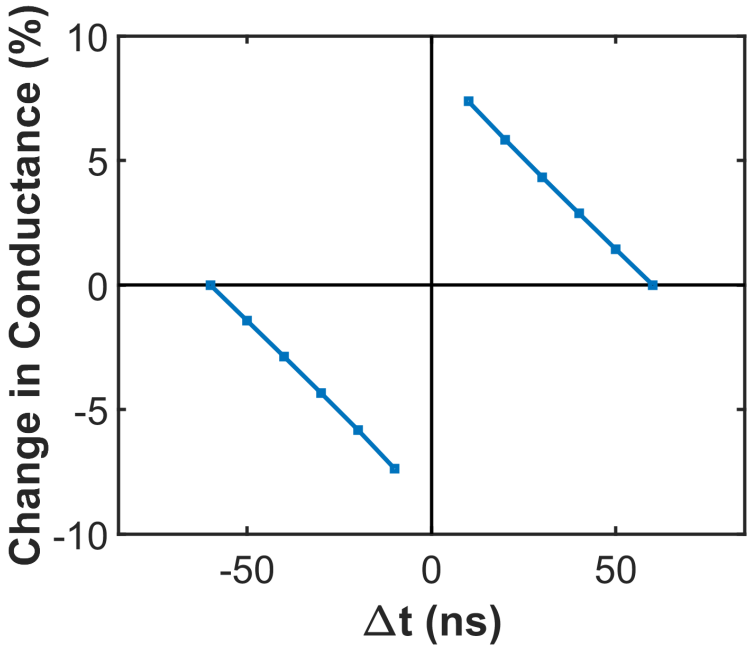
As the STDP is realized using pulse width modulated waves, the change in synaptic conductance is sensitive to the clock frequency of the system. With increased clock frequency, the change in memristance is lower, as the pulse width is narrower. Hence, the synaptic weight changes are smaller. The clock frequency was changed to 100 MHz and the STDP curve changes to a linear one, as can be seen from Fig. 4.3b. A closer look at the memristor model equations can describe the STDP behavior in further detail. For simplicity, let us assume that both memristors start out with the same memristance value, giving zero initial synaptic weight as $M_p = M_n = M$ and $G_{eff} = 0$. Assuming with applied voltage for learning, the memristance of both the memristor changes in equal amounts (ΔM), but at different directions. For potentiation, with $\Delta M < M_p, M_n$, equation 4.2 can be expanded using binomial theorem, and the synaptic conductance is given by:

$$G_{eff,pot} = \frac{2\Delta M}{M^2} \times [1 + (\frac{\Delta M}{M})^2 + (\frac{\Delta M}{M})^4 + ...] \tag{4.7}$$

As ΔM is small compared to M, the ratio $\frac{\Delta M}{M}$ is very small, with higher order terms becoming even smaller. Hence, effective conductance can be approximated as $\Delta G_{eff,pot} \simeq \frac{2\Delta M}{M^2}$, showing linear dependence on the change in memristance ΔM. Ignoring the window function for simplicity, the memristor model described in equation 2.1 points to linear



(a)



(b)

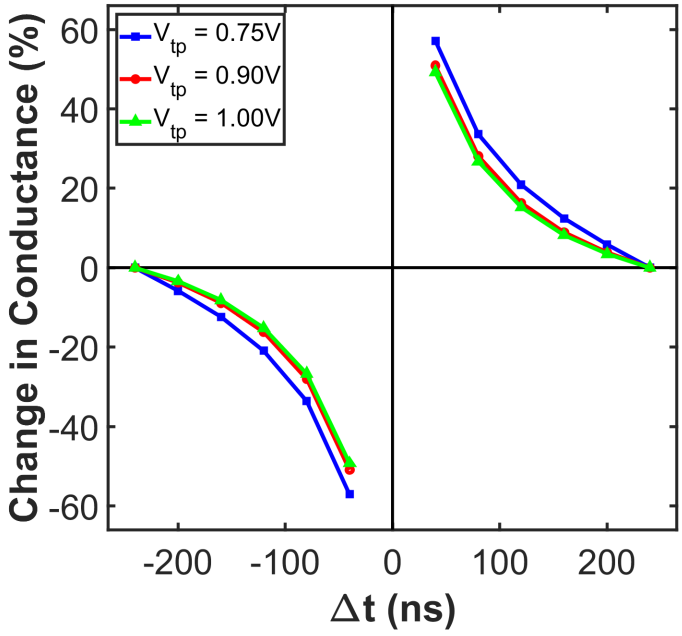
Figure 4.3: Change of STDP curve with clock frequency of the twin memristor synapse (a) Clock frequency of 25MHz (b) Clock frequency of 100MHz

dependence of change in memristance with the applied pulse width, when the applied voltage is greater than the switching threshold (V_{th}). This explains the linear change in memristance with higher clock frequency. When the clock frequency is lower, the pulse width is applied for longer periods of time, resulting in higher memristance change. As a result, the higher order terms of equation 4.7 cannot be ignored and the change is conductance is higher than linear relation. The simplified assumptions made here are sufficient to qualitatively explain the changes in STDP behavior under different clock frequencies.

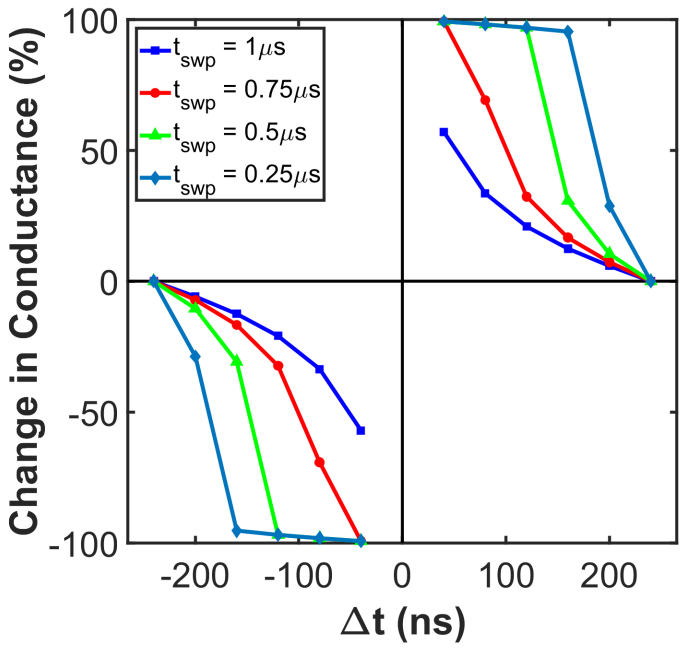
The memristor device parameters such as switching time and threshold voltages has been reported to be affected by process variations and mismatches [23, 12, 38]. There is mismatch in switching threshold voltages, as LRS to HRS switching threshold (V_{tn}) is higher than HRS to LRS switching threshold (V_{tp}). Furthermore, the switching time while going from HRS to LRS (t_{swp}) is reported to be orders of magnitude smaller than the switching time while going from LRS to HRS (t_{swn}). The effects of the parameter mismatches were analyzed and the results are plotted in Fig. 4.4. The STDP characteristics of the twin memristor synapses remains largely unaffected by the switching voltage mismatch, as the anti-symmetric curve is retained. However, with switching time mismatch, the change in conductance is highly affected because the switching in one direction is now significantly faster than the other direction. Due to the short switching time, the drastic conductance change of the device dominates the overall conductance change, so this faster switching device determines the (crippled) STDP behavior observed in this situation.

4.4 Sparse Spiking Neural Network

The performance of the designed twin memristor synapse was evaluated by integrating it in a sparse Spiking Neural Network (SNN) setup to carry out classification tasks on three datasets from UCI machine learning repository [33]. The Iris dataset has 150 entries, each of which has four attributes of Iris flowers and three output tags. The Wisconsin Breast Cancer dataset has 699 input data, and each having 9 features, and 2 output labels. The Pima Indian Diabetes dataset consists of 768 patient entries, each with 8 input features, and 2 output labels. To generate networks for the classification tasks, an Evolutionary Optimization (EO)



(a)



(b)

Figure 4.4: Effect of device parameter mismatch on STDP curve of the twin memristor synapse. (a) Switching threshold voltage mismatch, keeping V_{tn} constant at 1V and changing V_{tp}. (b) Switching time mismatch, keeping t_{swn} fixed at $1\mu s$ and changing t_{swp}.

algorithm was utilized [78]. In this method, a network was built with a constant number of input and output neurons, depending on the number of input features and output labels. However, the number of hidden neurons and connectivity(synapse) among input, hidden, and output neurons, as well as the synpatic weight and delay was not fixed, and was allowed to vary over evolutionary generations. In each generation, each network in the group is evaluated in a simulation, and each network is scored according to its performance in a specific application. The best scoring networks were used as parent networks for the next generation. The parent networks go over a probabilistic recombination through mutation and crossover to generate new sets of networks with different connections and synaptic properties, and thus the evolution process continues. The EO algorithm is implemented in a high-level software to generate off-chip networks with twin memristor synapses.

For the datasets mentioned above, EO was applied to find the best performing networks for each, with three different STDP cycle (1,3, and 5) tracking option. Half of the dataset was used for training purposes, while the other half was used for testing, the result of which is shown in 4.5. All of the networks' accuracy increases with more number of cycle tracking, although for the Iris dataset, there is only marginal improvement. As the Iris networks are small compared to the other ones, there is very little room for improvement even with multiple cycle tracking. The Iris network consists of 4 input and 3 output neurons, whereas the Breast Cancer network has 27 input and 2 output neurons and the Diabetes network has 24 input and 2 output networks. The accuracy is depended on different parameters of the network such as number of hidden neurons, synapses, their connections as well as synaptic weights and delay. However, from the results of Fig. 4.5 we can conclude that tracking for higher number of STDP cycles generally provides better accuracy for larger networks. Furthermore, the increase in accuracy with number of STDP cycles have diminishing returns. The increase in accuracy while going from 1-cycle to 3-cycle tracking is higher than that of 3-cycle to 5-cycle tracking. With higher cycle tracking, the circuit overhead also increases. Hence, considering the trade-off between the increase in accuracy and circuit complexity, circuits with the 3-cycle tracking ability can be a viable options for building spare neural networks with twin memristor synapses.

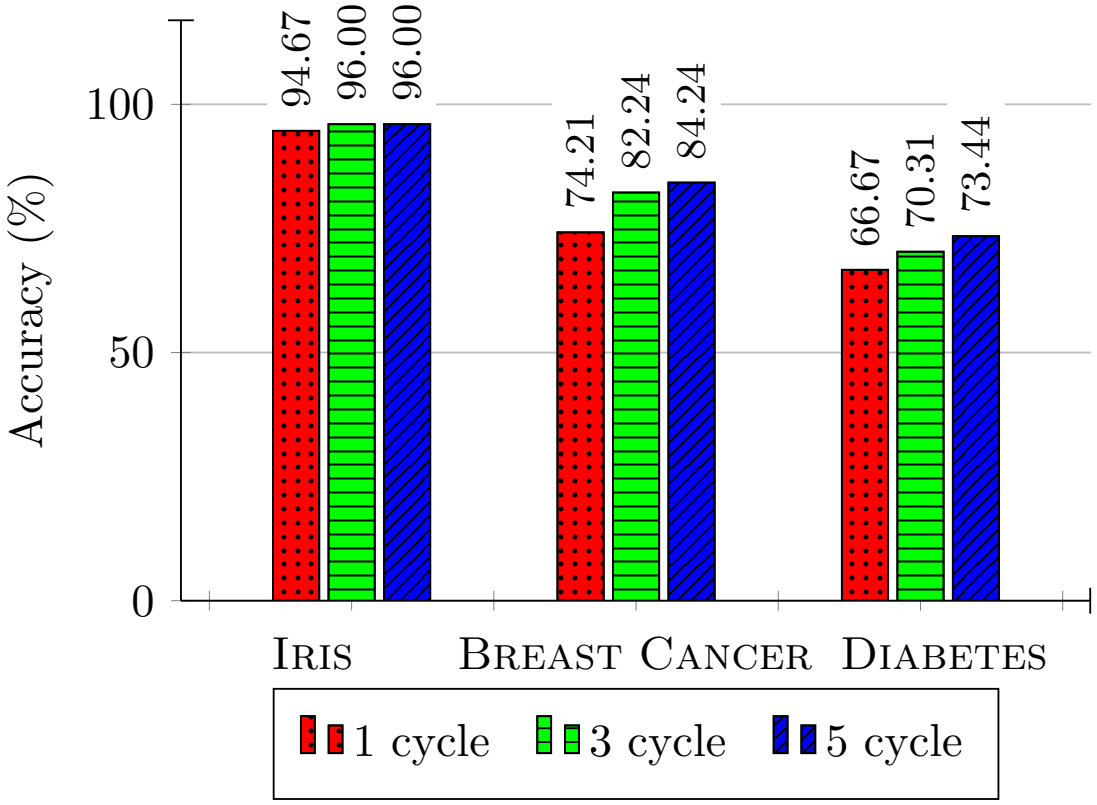


Figure 4.5: Performance of the best scoring networks for Iris, Wisconsin Breast Cancer and PIMA Indian Diabetes dataset, with different STDP cycle tracking abilities.

4.5 Energy Estimates

The energy consumption of the proposed twin memristor synapse was calculated and is provided in Table 4.1. An analog synapse was presented in [64] that consumes 1.8mW power for a collection of synapses, leading to 0.23nJ to 0.23mJ per synapse. In another approach [20], pulses were directly applied to the synapses, reporting energy values that range from 11 to 0.1pJ per spike for memristors with resistance range of 1kΩ to 1MΩ, respectively. In [41], the memristors used ranged from 70Ω to 670Ω and reports energy consumption of 36.7pJ per spike.

Tracking for more number of STDP cycles increases circuit overhead and hence the energy consumption is also higher. The energy consumption listed in Table 4.1 was based on the memristive device of [12], with LRS and HRS of $5k\Omega$ and $50k\Omega$, respectively. The energy requirements will be even lower with memristors with higher ranges such as $500k\Omega$ and $200M\Omega$ reported in [59].

Table 4.1: Energy estimates of the proposed synapse

Number of Cycles Tracked	Energy per Spike (pJ) Idle	Energy per Spike (pJ) Accumulation	Energy per Spike (pJ) Learning
1 cycle	0.07	1.45	2.58
3 cycle	0.20	1.68	3.41
5 cycle	0.32	1.76	3.76

Chapter 5

Memristive Crossbar Based Spiking Neuromorphic System

In this chapter, a spiking neuromorphic system is presented which uses a memristive crossbar to build a single-layer Spiking Neuron Network to classify handwritten digits. At first, the system architecture with the temporal input encoding scheme and the building blocks of the neuromorphic system such as input neurons, twin memristors synapses, and homeostasis enabled output neurons are proposed. Using the building blocks, a twin memristive crossbar is constructed to perform a pattern recognition task. The performance of the designed system was evaluated by varying the system parameters and the robustness against the memristive device imperfections were analyzed.

Disclosure

This chapter is an adapted version of the manuscript published in [4] with minor modifications.

5.1 System Architecture

5.1.1 Input Encoding

The input to a spiking neuromorphic system needs to be spiky in nature. To convert the input data into spikes, a temporal encoding scheme is presented here. The input considered here is an 8×8 image, with pixel intensities ranging from 0 to 16. Before applying the input encoding scheme, a pre-processing step is performed wherein the dynamic range of the pixel intensities are downsampled to -4 to +4, as illustrated in Fig. 5.1. Assigning polarities to the pixel values allows the temporal encoding scheme to be coherent with STDP rule. The input encoding scheme is shown in Fig. 5.2, where the top row of the pre-processed input image is converted into temporally coded spikes (TCS). Each pixel of the images produces a spike input and there are 64 different inputs going to 64 input neurons, corresponding to each pixel. A time frame of 8 timeslots are considered for each input, with higher pixel intensity magnitude generating a spike closer to the center of the time frame. Positive and negative intensity pixels are placed before and after the center of the time frame, respectively. The lower magnitude pixels are gradually placed further away from the center of the time frame. As the output neuron is forced to spike at the center of the time frame (timeslot 4 in Fig. 5.2), the encoding scheme allows convenient spike adaptation for performing STDP. As the most positive and negative magnitude pixels (timeslots 4 and 5, respectively) are placed closest to the output neuron spike, there would be maximum potentiation and depression for those inputs, respectively. Conversely, potentiation and depression will be weakest for the least positive and negative magnitude pixels (timeslots 1 and 8, respectively), respectively as they are placed farthest away from the output neuron spike.

5.1.2 Input Neuron and Synapse

The input encoding scheme presented generates the TCS that are provided to the input neurons. The input neuron is responsible for generating the required pulse wide modulated signals to drive the synapses. The generated signals differ according to the phase of operation,

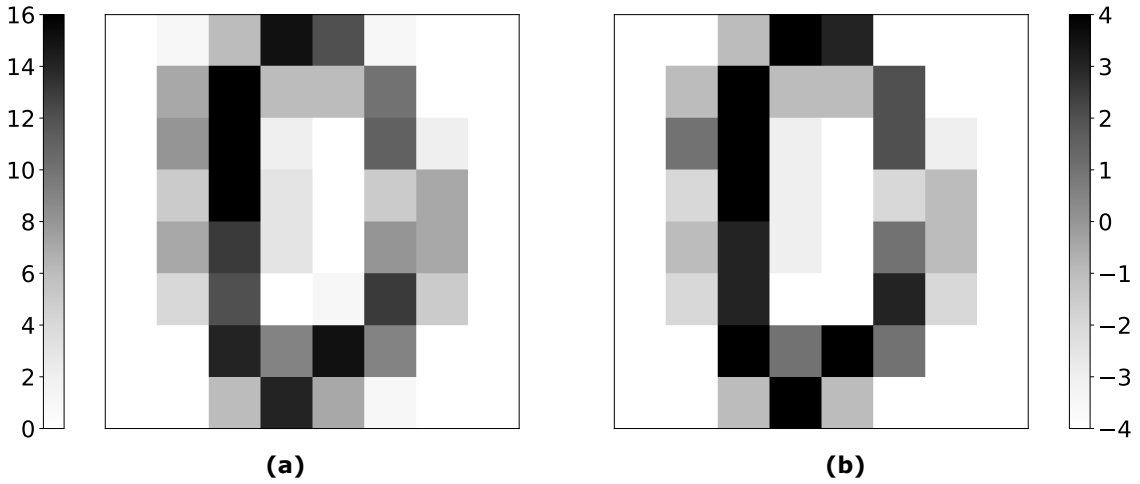


Figure 5.1: The input image is pre-processed to reduce the dynamic range of the pixel values (a) Original image with pixel values from 0 to 16 (b) Converted image with pixel values from -4 to +4

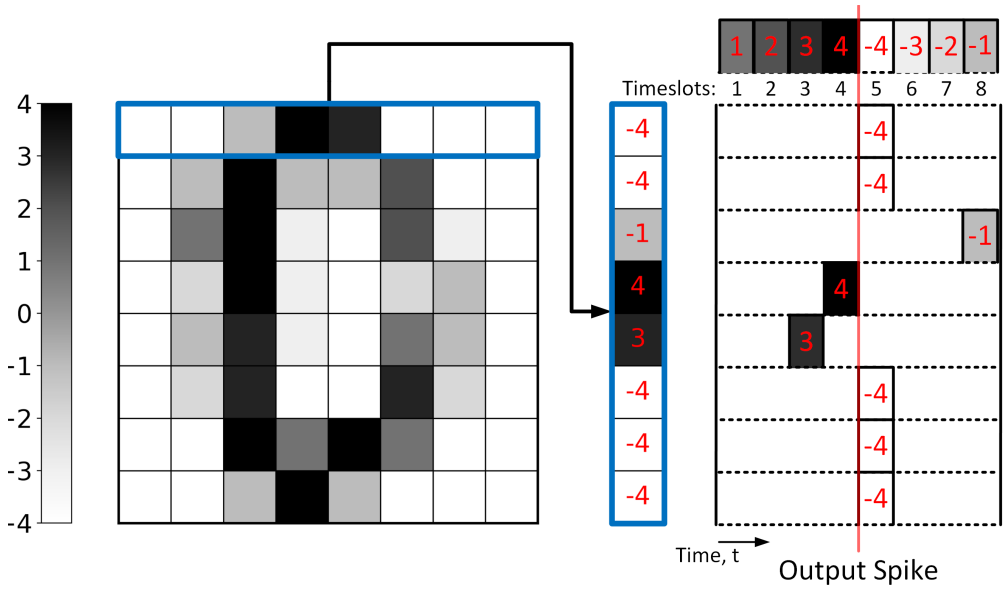


Figure 5.2: Proposed input encoding scheme to transform the input pixel from values into temporally coded spikes (TCS). A time frame of 8 timeslots is considered with output neuron spiking at the middle of the time frame shown with a vertical red line. The input spikes are placed such that positive and negative polarity pixels are placed before and after the output neuron spike, respectively, with higher magnitude pixels being placed closer to the time frame.

with different signals being generated for accumulation and STDP enabled learning. The operating phases of the input neuron is discussed along with the introduction of the synapse.

The design of the synapse is adapted from the one presented in chapter 4, which was derived from [24, 5]. The proposed design is demonstrated in Fig. 5.3, with two terminals of one end of the twin memristor driven by the proposed input neuron, while the other two terminals at the other end are combined together to form the postsynaptic node, which is driven by the output neuron. While the synaptic structure has the same architecture as the one mentioned before, the proposed synapse idea enables the crossbar based operation. In addition to the feed-forward spike, the previously proposed synaptic architecture of chapter 4 required the post-neuron fire to be fed back to the pre-neuron and the synaptic control block to perform STDP. This extra connection added to the circuit overhead and crossbar implementation was proven to be difficult. Hence, the improved synapse design is proposed, where no feedback connection from the output neuron is required and a twin memristive crossbar is easily realized.

The neurons and synapses operate on three different phases, namely accumulation, potentiation and depression. During the accumulation phase, only the spikes generated by the positive input pixels are accumulated at the output neuron. The learning of the designed system is divided into the potentiation and depression phases, and depends on the polarity of the input. Synapses potentiate only if the input was positive and depress only if the input was negative. When the output neuron spikes, the learning starts with the depression phase, followed by the potentiation phase, with only one phase being active for a particular synapse.

The input neuron drives the twin memristors M_p and M_n to different voltage polarities and for different duration, as required by the operating phases and the TCS. The applied voltages and the timings for the accumulation phase is shown in Fig. 5.4a, which occurs from timeslots 1 to 4. The width of the applied pulses is proportional to the input pixel intensity. The twin memristors M_p and M_n are pulled to the positive (VDD) and negative (VSS) rail voltages, respectively, while the postsynaptic node is pulled to the mid-rail (GND) voltage by the integrator op amp of the output neuron. Accordingly, there is positive and negative current flow through M_p and M_n, respectively, which results in a net current flow into the

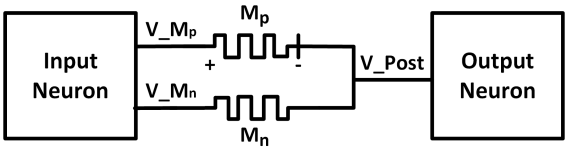


Figure 5.3: The input and output neurons being connected by the twin memristor synapse design. The vertical bar on the memristor symbol of M_p indicates the negative polarity terminal of the device. The bar is not shown for M_n, as the synapse design does not require its memristance to change and hence, it could be placed either way, disregarding the polarity.

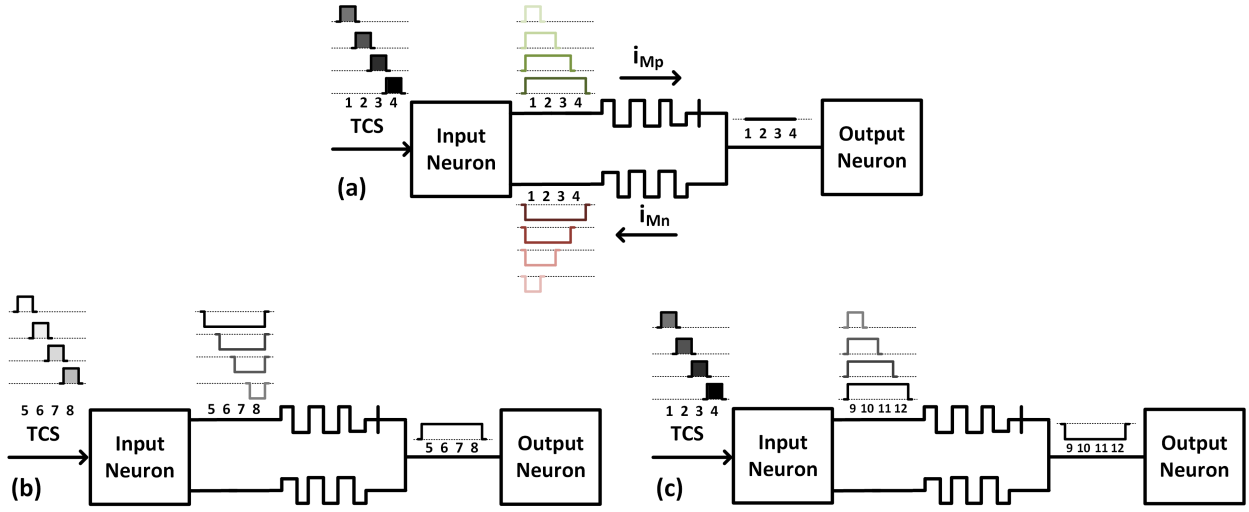


Figure 5.4: TCS and resulting pulses produced by the input neuron to be applied to the twin memristor synapse for the three operating phases. Timeslots are also shown which could be related to Fig. 5.2 for clarity. (a) TCS and pulses during accumulation phase, which occurs from timeslots 1 to 4 for positive intensity pixels. Negative intensity pixels are not accumulated and ignored (b) TCS and pulses during depression phase, occurring from timeslots 5 to 8 for negative intensity pixels when the associated output neuron spikes due to the accumulation from other connections to it. (c) TCS and pulses during potentiation phase, occurring from timeslots 9 to 12 for positive intensity pixels when the associated output neuron spikes. The applied pulse from the input neuron is the same as the one applied during the accumulation phase, only delayed by 8 clock cycles. The memristor M_n is left floating during depression and potentiation, and hence the memristance does not change. It only contributes during the accumulation phase.

output neuron, which is given by:

$$i_{net} = i_{M_p} - i_{M_n} = V \cdot (\frac{1}{M_p} - \frac{1}{M_n}) \tag{5.1}$$

where, V is the magnitude of the voltage that is applied across both the memristors. V is only half of the rail-to-rail voltage, as the postsynaptic node is at mid-rail, and hence, less than the switching threshold of the memristors. This is necessary to make sure that during the accumulation phase, there is no inadvertent change in memristance of any memristor. The effective conductance of the twin memristor synapse, derived from equation 5.1 is given by:

$$G_{eff} = \frac{1}{M_p} - \frac{1}{M_n} = G_p - G_n \tag{5.2}$$

The effective conductance of the twin memristor synapse is given by the conductance difference of the memristors. The net current provided by the conductance of the synapse goes into the postsynaptic node to be accumulated by the integrator of the output neuron. When the accumulated voltage exceeds the threshold of the output neuron, a synchronous output spike is generated along with a backward pulse that drives the postsynaptic node. The backward pulse pulls the postsynaptic node to positive rail from timeslots 5 to 8, indicating depression phase and negative rail from timeslots 9 to 12, indicating potentiation phase. The output neuron structure and operating principle to accumulate, spike and provide backward pulse is detailed in section 5.1.3. Learning at the synapses commences with the spike from the output neuron, which is always made to occur at timeslot 5, regardless of when the accumulation voltage crosses the threshold. For learning, input neurons applies voltage only to the memristor M_p, while the other memristor M_n is left floating. As the postsynaptic node voltage is changed during learning, only the memristor M_p changes it state, meaning learning only affects M_p. As M_n is not driven by the input neuron, the applied voltage across it is zero, and hence there is no change in memristance. The potentiation operation is shown in Fig. 5.4c, during which the input neuron drives the positive terminal of M_p to the positive rail and the output neuron drives the negative terminal to the negative rail. As a result, the voltage across M_p exceeds the switching threshold, leading to a fall (ΔM) in

its memristance, and increase in the overall effective conductance of the synapse, as can be derived here:

$$G_{eff,pot} = \frac{1}{M_p - \Delta M} - \frac{1}{M_n} \tag{5.3}$$

For the depression operation shown in Fig. 5.4b, the input neuron drives the positive terminal of M_p to the negative rail and the output neuron drives the negative terminal to the positive rail. As a result, the voltage across M_p exceeds the switching threshold in the other direction, leading to a rise (ΔM) in its memristance, and decrease in the overall effective conductance of the synapse, as can be derived here:

$$G_{eff,dep} = \frac{1}{M_p + \Delta M} - \frac{1}{M_n} \tag{5.4}$$

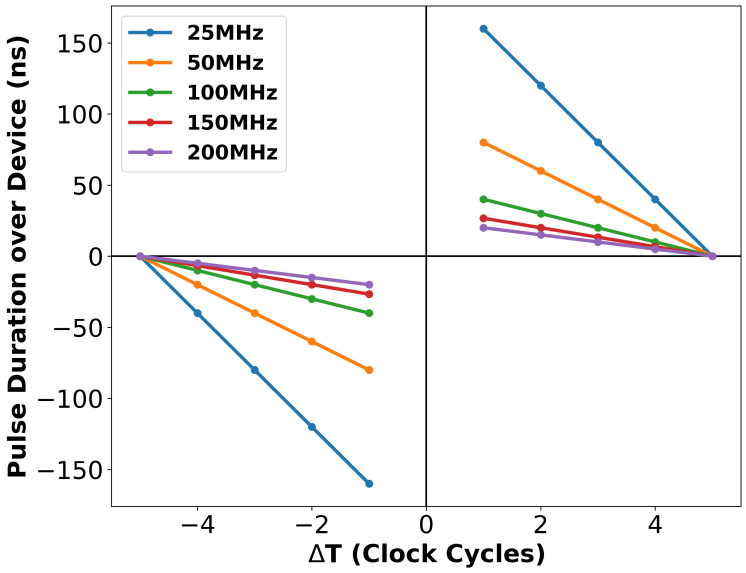
If we take the derivative of equations 5.3 and 5.4 with respect to the change in memristance ΔM, the change in the effective conductance of the synapse for both potentiation and depression can be found.

$$\frac{dG_{eff,pot}}{d\Delta M} = \frac{1}{(M_p - \Delta M)^2} \tag{5.5}$$

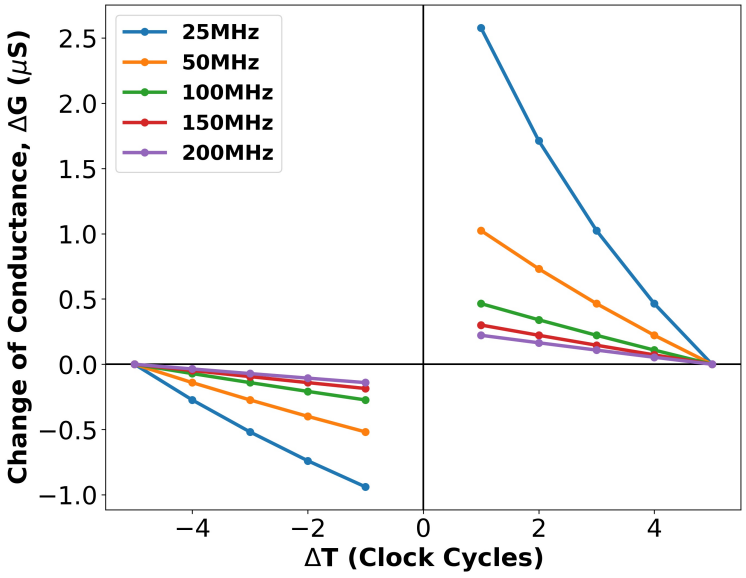
$$\frac{dG_{eff,dep}}{d\Delta M} = -\frac{1}{(M_p + \Delta M)^2} \tag{5.6}$$

As we can see from the equations 5.5 and 5.6, the change of effective conductance for potentiation is higher than that of depression, which has been reported in [17].

To enable STDP behaviour in the twin memristor synapse, the input neuron considers the temporal difference between the TCS and the output neuron spike and produces pulses with width that is inversely proportional to the time difference between the spikes. When the spikes are further apart, the applied pulse is shorter, resulting in a small change in the effective synaptic conductance and vice versa. The pulse duration and the resultant change of conductance with respect to the difference in spike timing using the memristor model of section 2.1.1 is shown in Fig. 5.5 for different clock frequencies. With the decrease of frequency, the applied pulses become wider, producing greater change in the effective conductance of the synapse.



(a)



(b)

Figure 5.5: (a) The pulse duration over the memristive device with respect to the difference in spike timing of input and out neuron in terms of number of clock cycle for different clock frequencies. (b) The resulting change of conductance for different clock frequencies. Here, $\Delta T = t_{post} - t_{pre}$, depicting the clock cycle difference between post and pre-neuron spike.

The schematic design of the input neuron is shown in Fig. 5.6, which provides the necessary pulses for each phase of the synapse. Alongside the system clock, there are two other inputs that are needed for the input, which are the TCS signal, the spike representing the pixel intensity and the POL signal, which represents the polarity of the pixel. For positive and negative pixels, the POL signal is '1' and '0', respectively. The operating phases are 4 cycle long each, constituting an operation cycle. The first operating phase is accumulation, which is shown in Fig. 5.6a, where a TCS representing a positive input generates a positive and negative rail voltage at memristor M_p and M_n respectively. The pulse width of the applied voltage is modulated according to the intensity of the pixel (i.e., timing of the TCS). In this example, a positive input pixel intensity of +2 is considered, for which the TCS is at timeslot 2 and POL is '1'. Whenever POL is high, the input neuron generates a positive and negative rail voltage at M_p and M_n respectively, and deactivates the output as soon as the TCS signal goes low by using a negative edge triggered D flip-flop. This approach generate the pulse required pulse widths proportional to the input pixel intensity, as higher pixel intensity input generates TCS later, and the voltage stays high for longer period of time. The generated waveforms from a Cadence Spectre simulation with IBM 65nm PDK of Fig. 5.7a demonstrates the accumulation phase for the +2 input, showing that the voltage is applied for 2 clock periods (from 10ns to 50ns) for accumulation.

The applied pulse width modulated waveform to the memristor M_p for accumulation is also stored in a 4-bit shift register to be reused for the potentiation phase, which is shown in Fig. 5.6c. In this phase, the PWM waveform is only applied to M_p, while M_n is left floating, as there is no learning for it, according to discussion above. The waveforms for potentiation phase are shown in Fig. 5.7a, which occurs from 90ns to 130ns. The previously generated and stored 2 cycle wide pulse during accumulation is simply replicated here by the shift register. As the output neuron drives the postsynaptic node to the negative rail during potentiation, the voltage across M_p is above the switching threshold and thus the memristance reduces, causing an increase in synaptic weight. As an input of +2 was used in this example, potentiation also happens for 2 clock periods. After that, the memristor terminal connected to the input neuron is not driven anymore and left floating, so that there

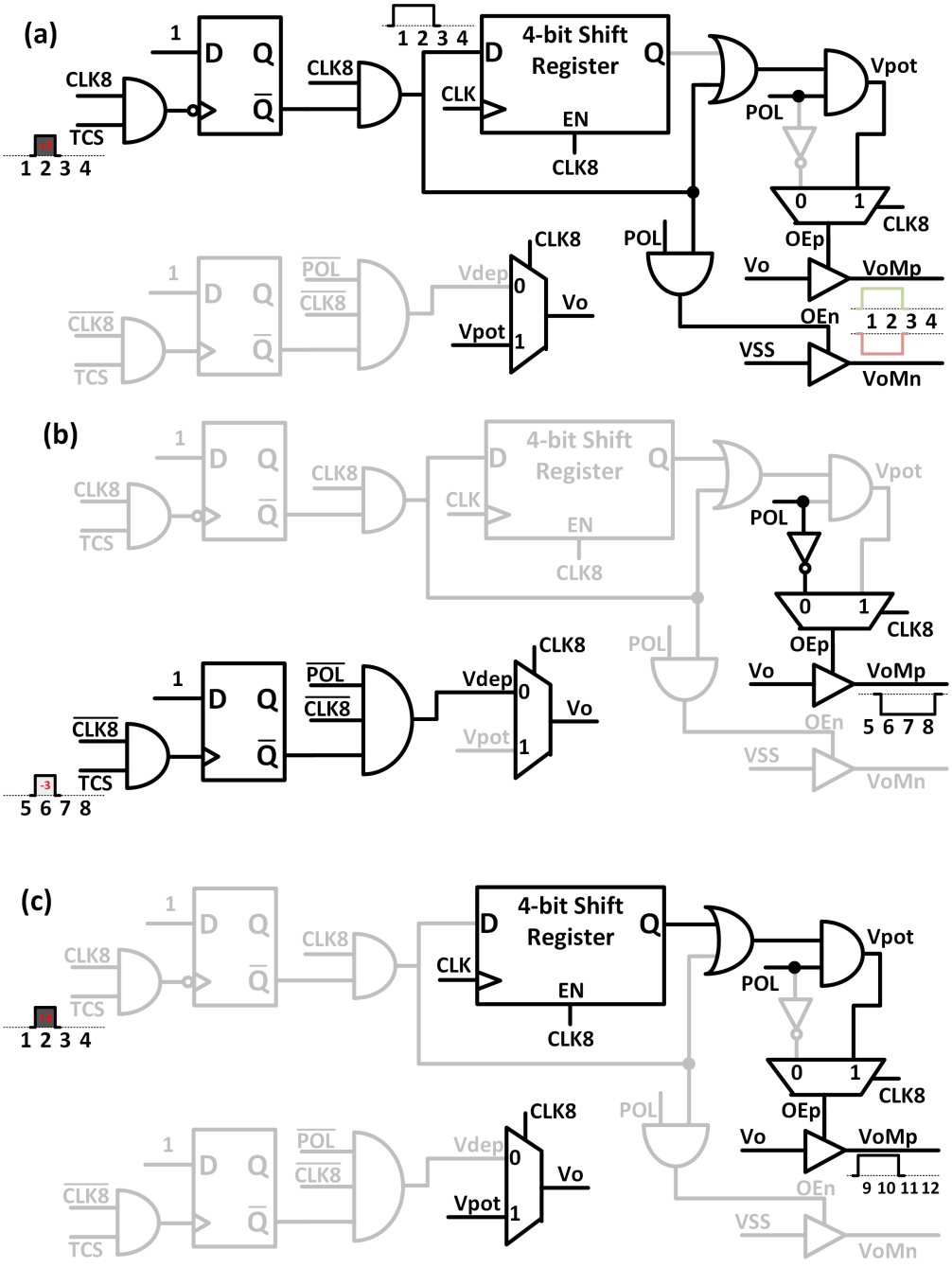


Figure 5.6: Schematic diagram of the designed input neuron and the highlighted active blocks during different phases of operation. TCS is the encoded spike signal, CLK is the system clock, $CLK8$ is derived from the system clock to have 1/8th of the frequency, POL is the signal indicating the polarity of the input pixel, and $VoMp$ and $VoMn$ outputs are the pulse width modulated signals generated by the input neuron that are applied to the memristors M_p and M_n, respectively. (a) Input neuron during accumulation phase for an input pixel intensity of +2 (b) Input neuron during depression phase for an input pixel intensity of -3 (c) Input neuron during potentiation phase for an input pixel intensity of +2.

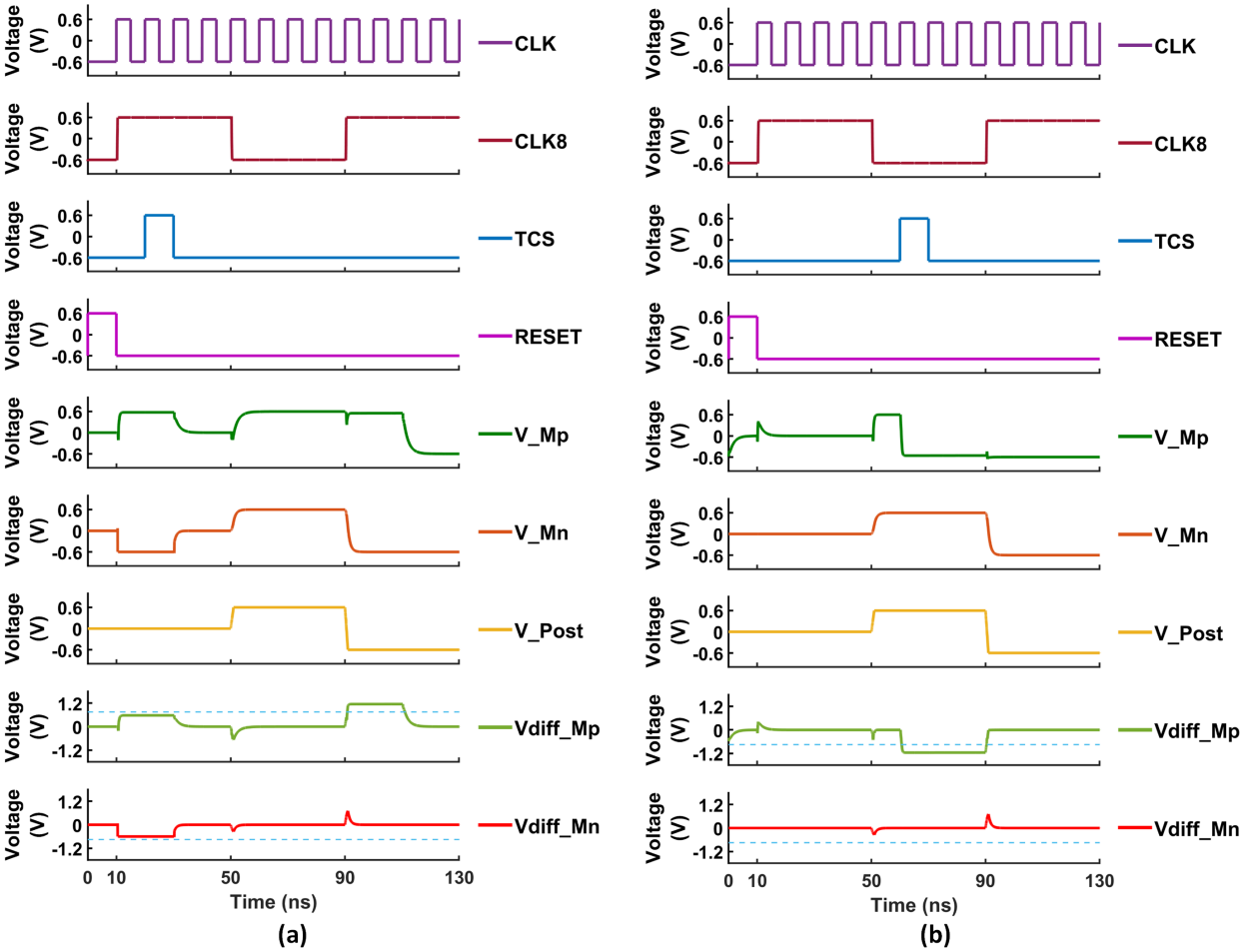


Figure 5.7: Cadence Spectre simulation of the designed input neuron to generate pulse width modulated waveform from the TCS for different phases of operation. A 100 MHz system clock is represented by CLK, which is divided by 8 to generate $CLK8$ signal. The encoded spike signal provided to the input neuron is represented by TCS signal. The system reset indicating the start of an input image is represented by $RESET$. The outputs from the input neuron that drive the memristors M_p and M_n are represented by V_M_p and V_M_n, respectively, while $Vdiff\_M_p$ and $Vdiff\_M_n$ represents the voltage across them, respectively. V_Post represents the postsynaptic node voltage, supplied by the output neuron. (a) Example waveform for an input pixel intensity of +2 showing accumulation phase from 10ns to 50ns and potentiation phase from 90ns to 130ns. (b) Example waveform for an input pixel intensity of -3 showing depression phase from 50ns to 90ns.

is no voltage across it even if the postsynaptic node is still held to the negative rail for the whole potentiation period.

There is no accumulation for negative intensity pixels, which is ensured by the *POL* signal being '0' for them. The design of the output neuron guarantees that the output neuron spike occurs at the start of timeslot 5, right after accumulation phase ends. The negative intensity pixels are encoded such that they generate TCS after the output neuron spikes, leading to depression in accordance with the STDP update rule. Furthermore, the higher magnitude negative intensity pixels' generated TCS are placed closer and lower intensity pixels' generated TCS are placed gradually further away. The input neuron schematic highlighting the active blocks during the depression phase is shown in Fig. 5.6b. During the depression phase of timeslots 5 to 8, the memristor M_p is left floating by default. If there is an output neuron spike, and the input to the input neuron represents a negative intensity pixels, the voltage applied M_p goes to the negative rail as soon as the TCS arrives. A positive edge triggered D flip-flop is used to achieve that, with the TCS triggering it. At the onset of the output neuron spike (and the depression phase), the postsynaptic node is pulled to the positive rail by the output neuron. Now, there is a voltage across M_p which is above the switching threshold, causing it to increase memristance. The increase in memristance depends on the width of the applied pulse, which in turn depends on the timing of the TCS and the input pixel intensity. In this example, an input of -3 is considered which results in a 3 cycle wide negative rail pulse from the input neuron, as shown in Fig. 5.7b. The depression phase occurs from 50ns to 90ns, but the negative rail voltage is only applied from 60ns to 90ns, while the memristor was left floating from 50ns to 60ns, and followed the voltage at the postsynaptic node. As discussed before, the memristor M_n does not go through any change, as it is left floating by the input neuron in depression phase. The layout of the designed input neuron with IBM 65nm CMOS10LPe PDK is shown in Fig. 5.8.

5.1.3 Output Neuron

To construct an unsupervised learning platform, Winner-Takes-All (WTA) strategy is applied for realizing competitive learning in output neurons. The neuron with the earliest spike wins,

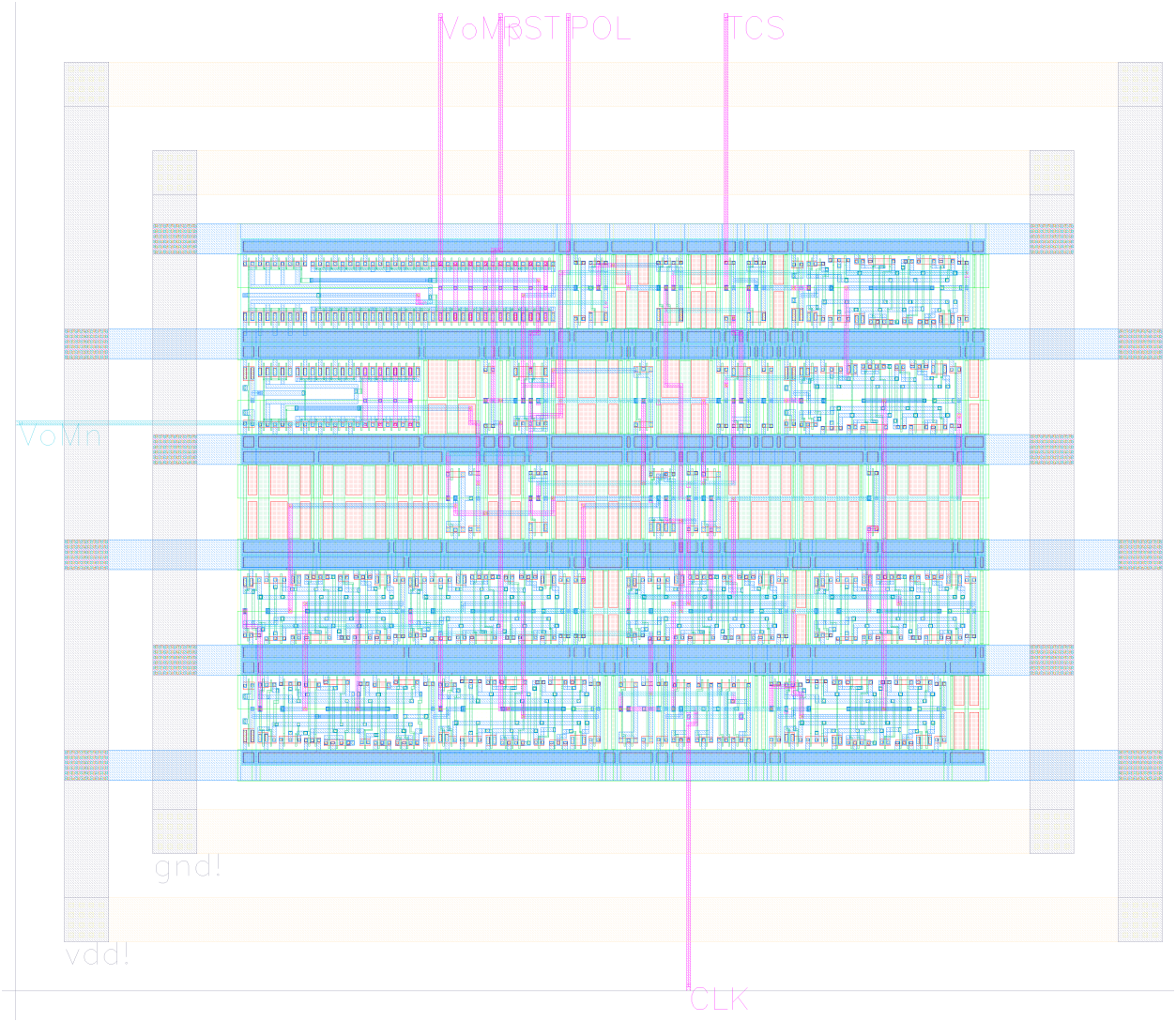


Figure 5.8: Input neuron layout designed with IBM 65nm CMOS10LPe PDK with an area of $34\mu m \times 25\mu m$.

and the synapses contributing to the winning neuron are strengthened in accordance with the STDP rule. This makes the neuron susceptible to spiking for the pattern that had been applied, in addition to other logically similar but physically different patterns. Hence, there is a need to discourage the winning neuron to spike for the patterns other than the one it had already won for. The mechanism to discourage the neuron to spike and control the spiking frequency is referred to as homeostasis. Typically, homeostasis in implemented by modifying the spiking threshold of the neuron. However, controlling the threshold of the neuron with a pin and analog reference voltage is difficult due to having large number of neurons. To overcome the challenge, this work proposes to modulate the rate of accumulation in the neuron instead of having an on-chip adaptive threshold. The rate of accumulation rate is varied by dynamically changing the integration capacitance of the integrator op amp, similar to the approach adopted in [19], where a stochastic neuron was built by randomly choosing the capacitance value. As an increase in integration capacitance decreases the rate of accumulation of non-leaky integrator, homeostasis can be implemented. Decreasing the rate of accumulation is equivalent to increasing the spiking threshold of the neuron, as the increased capacitance prolongs the time needed to get to the threshold of the integrate and fire neuron.

The schematic diagram of the proposed output neuron is shown in Fig. 5.9, which improves on the Integrate And Fire (IAF) neuron design of [24, 19]. The input current to the op amp is accumulated as a voltage, which is compared to a pre-set threshold by another op amp in open loop configuration. When the accumulated voltage at the integrator reaches the threshold, the comparator op amp generates a spike. The accumulated voltage at the integrator is:

$$V_{accum} = -\frac{1}{C_{fb}} \int_0^{\tau} i_{in}(t)dt \tag{5.7}$$

where, C_{fb} is the total capacitance, $i_{in}(t)$ is the input current to the integrator. The current is the net current produced by all the synapses contributing to the accumulation over the whole accumulation phase ('0' to τ), which is given by $i_{in} = G_{syn}V_{syn}$. Here, G_{syn} represents the equivalent synaptic conductance and V_{syn} represents the voltage difference across the synapses. The input pattern and the conductance of each synapse connected to the output

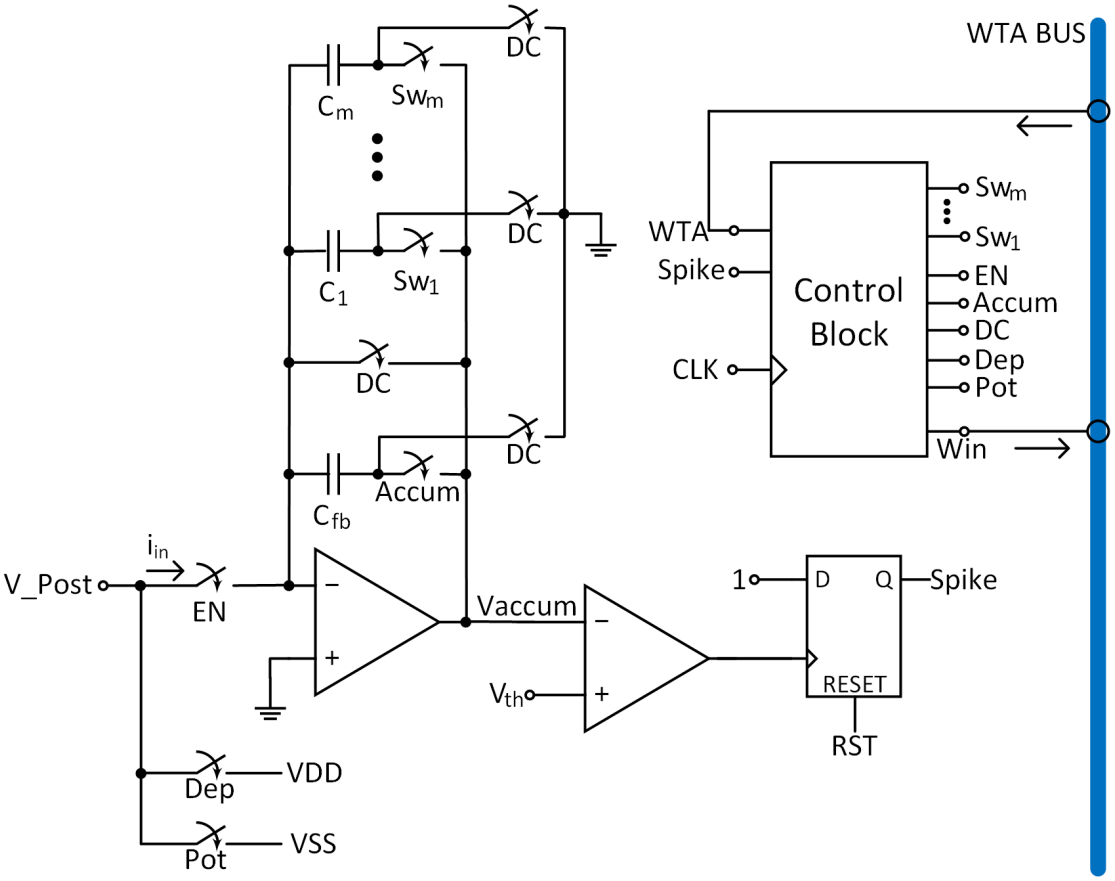


Figure 5.9: The schematic diagram of the proposed output neuron with dynamically adjusted capacitance and the Winner-Takes-All bus connection. A control block generates the necessary signalling for the WTA interface and capacitance switching.

neuron determines the number of spike needed for it to get to the threshold, V_{th} to produce a spike.

As we can see from the equation 5.7, increasing the feedback capacitance results in a reduced voltage accumulation on the integrator, reducing the spiking possibility of the neuron, even if the threshold remains the same. The effective feedback capacitance of the integrator is the sum of all the capacitors that are switched on, as they are connected in parallel (Fig. 5.9). The control block is responsible for generating the switching signals for the parallel capacitors as well as the necessary signalling for the WTA interface to enable lateral inhibition. The simulation of the output neuron is performed using Cadence Spectre and the the waveforms are shown in Fig. 5.10 to demonstrate the homeostasis mechanism. For simplicity, a constant input current of $10\mu A$ is considered with the equivalent synaptic conductance, G_{syn} of $1/60K\Omega$ and V_{syn} of $0.6V$. The neuronal threshold is assumed to be $-50mV$, which is indicated on the $Vaccum$ signal plot by a dashed blue line. The inverting integrator op amp integrates the current and when the accumulated voltage reaches the threshold, a spike is communicated to the WTA bus, which is represented by the *Spike* signal. This example scenario assumes that this particular neuron wins, making all other neurons inhibited, and is responsible for driving the postsynaptic node to the required voltage levels for learning. The backward learning pulse generated by the neuron is shown by V_Post, which can be related to the same signal shown in Fig. 5.7. In addition to that, the control block increases the effective feedback capacitance of the integrator by switching on additional parallel capacitor, lowering the rate of integration for subsequent accumulation phases. This is evident in the waveform for the next spike and accumulation phase, where it takes 31ns for the neuron to hit the threshold, whereas it had taken 24ns previously. Following the second spike, the control block enables more capacitors to establish the homeostasis, lowering the accumulation rate to the point that the neuron is unable to reach the threshold within the accumulation phase (from 240ns to 280ns), and as a result, the output neuron does not spike.

The control block of the output neuron, shown in Fig. 5.11, is primarily responsible for generating the proper signalling to indicate different phases of operation and in turn driving the postsynaptic node. To determine the operating phase, the control block communicates with the WTA bus to see if the particular neuron won or lost and accordingly generates the

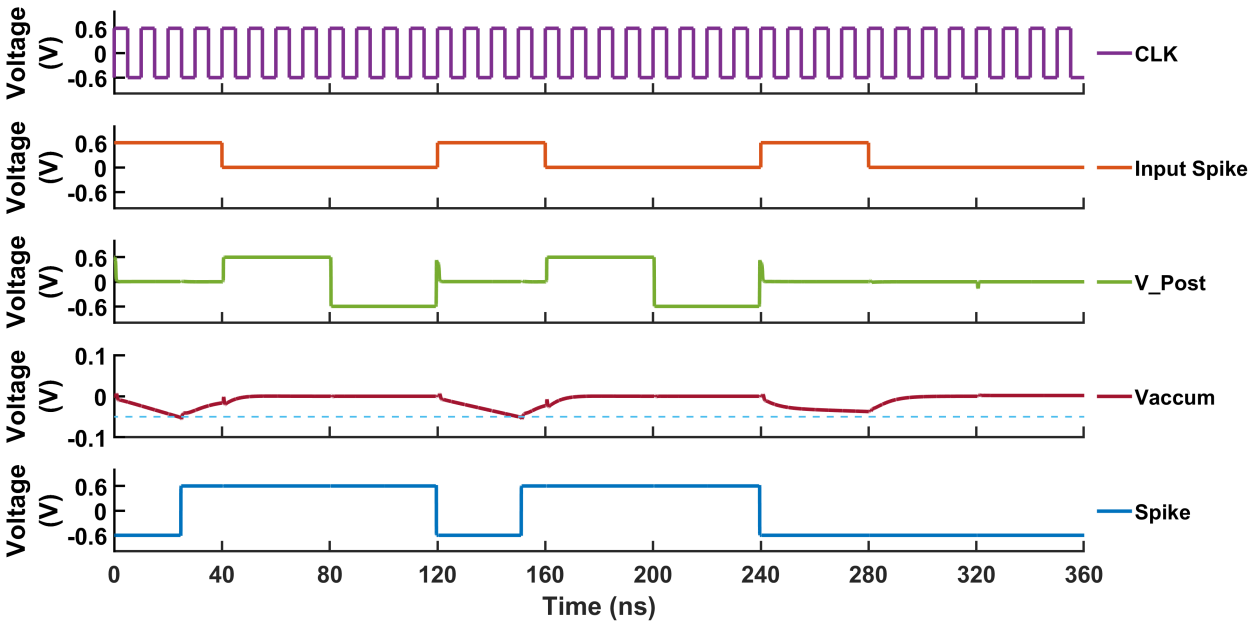


Figure 5.10: Simulation of the proposed output neuron using Cadence Spectre to demonstrate the homeostasis mechanism by showing reduced accumulation with each spike. The system clock is represented by the CLK signal. The applied input voltage for producing the input current (i_{in}) to the integrator is shown by the *Input Spike* signal. *V_Post* signal shows the voltage driven at the postsynaptic node by the neuron. The accumulated voltage at the integrator is represented by the V_{accum} signal along with the threshold of the neuron, indicated by a dashed blue line, upon reaching which, a spike is produced that is shown by *Spike* plot.

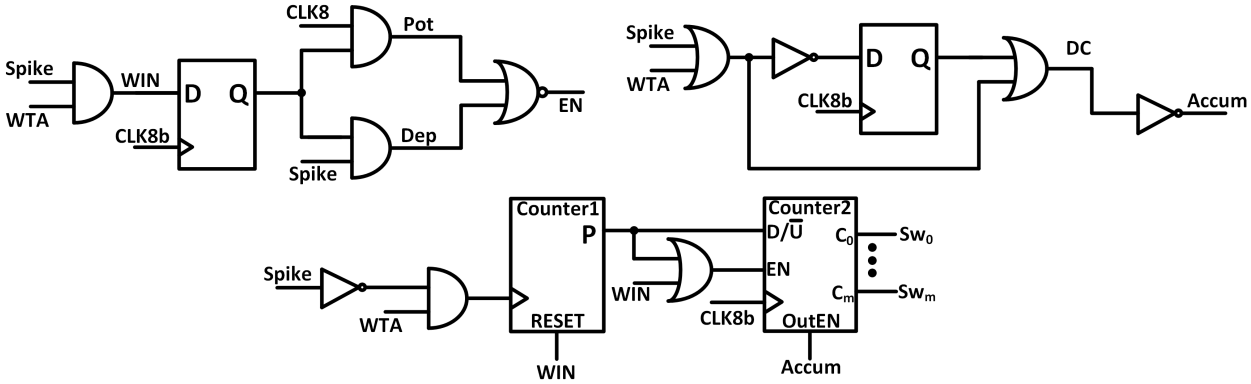


Figure 5.11: Schematic diagram of the control block of the proposed output neuron. The control block provides all the signals needed to carry out all the operating phases, in addition to interfacing with the WTA bus.

appropriate signals. The negative feedback of the op amp drives the postsynaptic node to virtual ground (mid-rail) during the accumulation operation. If there is no spike from any of the output neurons, the postsynaptic node is simply kept at virtual ground so that there no unintentional learning during the timeslots of depression and potentiation. For the winning neuron, the control block determines the depression and potentiation phase and accordingly turns of switches to provide the postsynaptic node with appropriate voltages, which is shown by the V_Post signal of Fig. 5.10.

Moreover, the control block control the switches of the parallel integration capacitor to adjust the effective feedback capacitance. This is done by a counter ($Counter2$), the count value of which is directly connected to those switches. With each spike, the count value increases, which switches the capacitors to increase the effective integration capacitance. The count values is updated as soon as the accumulation phase ends. If a neuron is unable to spike for a preset number of consecutive inputs, its count value is decreased to increase the accumulation rate, encouraging it to spike. This mechanism is here referred to as reverse homeostasis. Another counter ($Counter1$) is used to count the number of inputs the neuron has not fired for, and upon reaching a preset number, the counter generates a signal to indicate the $Counter2$ to count down, thus reducing the integration capacitance. The $Counter1$ size is equal to the number of bits needed to represent the preset value, whereas the $Counter2$ size is determined by the the number of parallel capacitors used in the design of the output neuron. The layout of the designed output neuron with IBM 65nm CMOS10LPe PDK is shown in Fig. 5.12.

5.2 Network Architecture

The performance of the proposed neurons and twin memristor synapses were evaluated by applying them in a crossbar based single-layer spiking neuromorphic system to classify handwritten digits dataset from the UC Irvine Machine Learning Repository [35].

Handwritten digits data was collected from 43 people as images, with 3823 image data from 30 people being used as the training set, while 1797 image data from the rest of the 13 people was used as the training set. The image resolution was 64 pixels, each image being

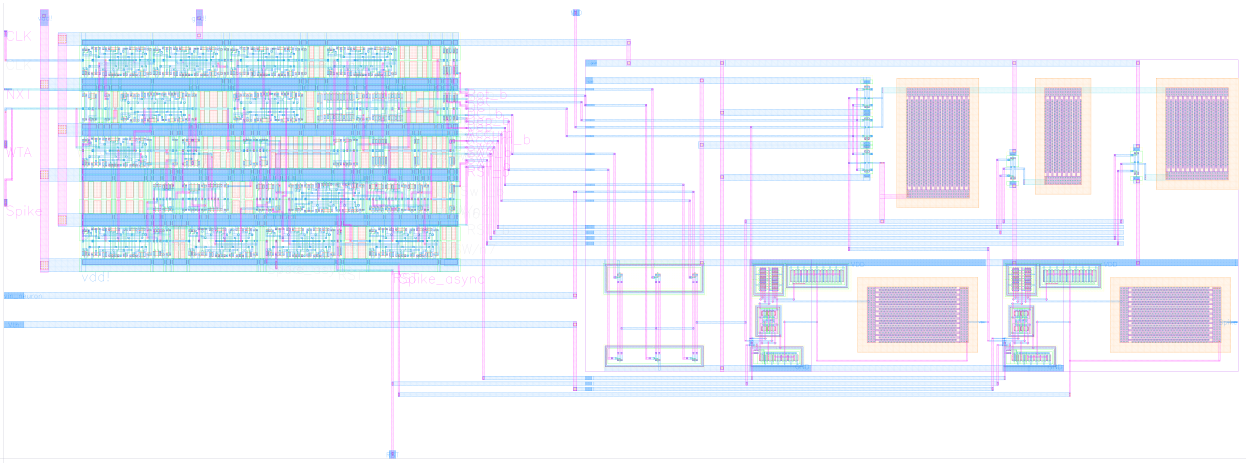


Figure 5.12: Output neuron layout designed with IBM 65nm CMOS10LPe PDK with an area of $122\mu m \times 38\mu m$.

8×8 matrix, with a dynamic range from 0 to 16. The pre-processing step flattens the 2D matrix to 1D array of pixels and applies the proposed down-sampling and encoding scheme of section 5.1.1 to generate TCS for each input neuron. The number of input neurons of the SNN was 64, each accounting for the TCS of each pixel of the image, and as a result the single layer SNN crossbar requires 64 rows. The number of columns is determined by the number of output neurons, which is a design variable of the unsupervised learning system, and was varied from 10 to 1000 in this work to see how it affects the performance of the network. A twin memristor synapse is at the crosspoint of each columns and rows. The single-layer SNN built with a crossbar of twin memristive synapses and lateral inhibition in the output neuron is demonstrated in Fig. 5.13. Each input neuron drives two rows of wires for the twin memristor, the green wire for the M_p memristor and the red wire for the M_n memristor. The shorted end of the memristors, i.e., the postsynaptic nodes constitute a column shown with blue vertical line, which connects to each output neuron. For accumulation, the output neurons pull them to mid-rail voltage. During the learning phases, the output neurons drive the backward pulse shown by V_Post signal of Fig. 5.7 to the columns.

To implement the competitive unsupervised learning with lateral inhibition [58], a WTA bus is interfaced to all the output neurons, which determined the neuron that won the competition. The asynchronous WTA bus and the logic has been adopted from [95], which ensures that as soon as an output neuron reaches the threshold, it is considered to be the winning neuron and at once, the all other neurons stop accumulating and does not go through learning phases for that input. Only the winning neuron applies the backward learning pulse to the corresponding column and synapses connected to that particular column goes through learning according to STDP. In addition to the synaptic update, the output neuron also goes through the homeostasis process by adding parallel capacitors through switches, resulting in a reduced accumulation rate for subsequent inputs, in accordance with our discussion of section 5.1.3.

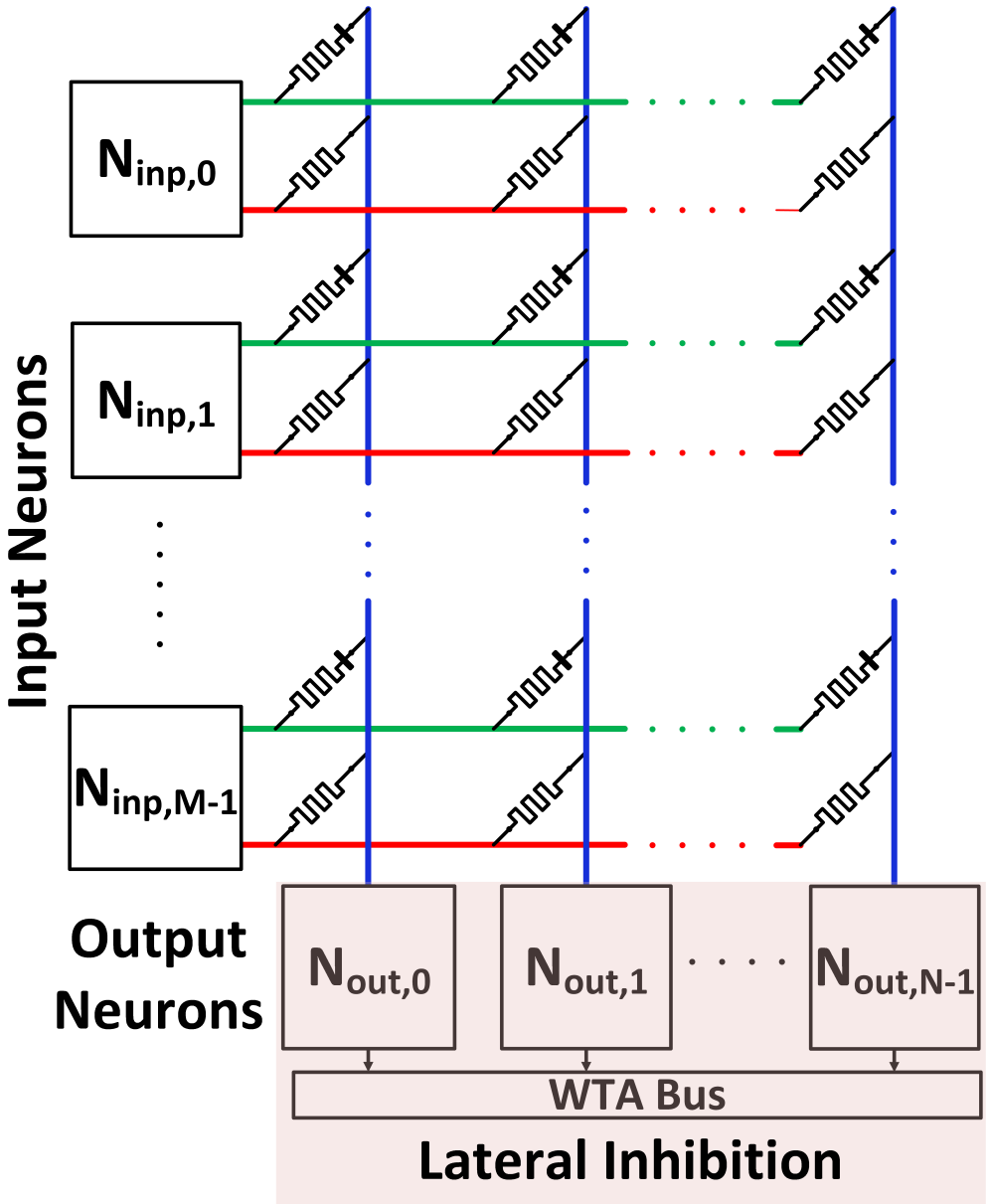


Figure 5.13: The structure of the proposed single-layer SNN built with M rows of input neurons, N columns of output neurons, and a crossbar of $M \times N$ twin memristive synapses.

5.3 Training

The twin memristive synapses used to build the single-layer SNN is made up of memristors that have their initial memristances normally distributed with a mean value. This gives rise to a random distribution of conductance of the the synapse. The average values of the column conductance is used to calculate an expected average accumulation in the output neuron, which could be used as a rough estimate of the neuron threshold, and is derived from equation 5.7:

$$Vth_{neuron} = -\overline{G_{col}} \cdot V_{diff} \cdot \frac{\Delta T}{C_{fb}} \cdot Pix_{avg} \tag{5.8}$$

where, average column conductance is represented by $\overline{G_{col}}$, the voltage difference across the synapse for accumulation is represented by V_{diff}, the system clock period is represented by ΔT, the feedback capacitor of the integrator is represented by C_{fb}, and the average pixel intensity over all the training images is represented by Pix_{avg}. As the homeostasis mechanism is realized by dynamically changing the capacitance of the integrator, and the accumulate rate, different adaptive threshold voltage for different output neurons is not necessary anymore, which enables us to use only a single pin to supply this common threshold across all the output neurons. The choice of the neuron threshold voltage by using equation 5.8 was justified by running simulations with changing the threshold by ±50% from the computed value, which produced results showing unaffected accuracy.

The training algorithm used here to train the randomly initialized single-layer crossbar based SNN to recognize handwritten digits is listed here:

1. $Img^{(k)}$ representing the k-th image of the dataset is considered.
2. The dynamic range of the pixel intensities is reduced to -4 to +4 from the original image
3. The input encoding scheme is applied to generate Temporally Coded Spikes (TCS) for each pixel, and they are provided to all of input neurons in parallel
4. **Accumulation:** Pulse width modulated waveforms proportional to the pixel intensity are applied to the synapses for positive (+1 to +4) intensity pixels only. The duration of the accumulation phase is 4 clock cycles.

5. If there is no spike from any of the output neurons, there is neither homeostasis in the neuron nor learning in the synapses, and the algorithm jumps to step (7)

6. If any of the output neuron generates a spike, WTA chooses the winner and learning commences for the synapses interfaced to the winner neuron or column and homeostasis occurs for the winner neuron. The mechanisms of learning are:

    - **Depression:** The TCS is generated by negative pixel intensity (-1 to -4) in this phase. The input and winner output neuron provide the required voltage shapes to depress the synapses linking the rows that had negative pixels and the winner column. The depression operation takes place after accumulation and is also 4 cycles long.
    - **Potentiation:** For pixels with positive intensity (+1 to +4), the input and winner output neuron provide the required voltage shapes to potentiate the synapses linking the rows that had positive pixels and the winner column. The potentiation and depression operation cannot occur simultaneously because different polarity backward pulse is needed for the two operations, even though the synapses going through the operations the different. The duration of the potentiation phase is also 4 clock cycles.
    - **Homeostasis:** The effective feedback capacitance of the integrator of the winner neuron is increased to reduce the rate of accumulation, discouraging the neuron to spike for dissimilar input. This step can occur in parallel to either the depression or potentiation phase.

7. The accumulated voltages at all of the output neurons are reset to make sure that they start from zero when the next input is applied.

8. Training ends for the input $Img^{(k)}$. The next image $Img^{(k+1)}$ is considered and the training cycle (steps (2) to (7)) repeats. This continues for all the images in the training dataset.

9. Reverse homeostasis mechanism is applied in parallel to the training cycle. A counter is used to count the number of consecutive inputs a particular neuron neuron has

not fired for. If the count reaches 20, the integration capacitance of that neuron is reduces so that the neuron is encouraged to spike, which is opposite to the homeostasis mechanism applied for the learning phase.

An epoch is defined as one iteration of going through the training cycle of all the data in the training dataset. An example network having 50 columns and 10 epochs was trained using a simulator written in Python, the result of which is shown in Fig. 5.14. As there are multiple slightly different versions of the same digit, the number of columns (output neurons) was taken to be higher than number of classification categories (10 digits). The 10 best performing neurons are chosen and the conductance of the 64 synapses connected to them are shown in Fig. 5.14a by rearranging them in 8×8 matrices. The synaptic conductance of the synapses mimics the digits that are learned, which means the connected output neurons are expected to spike for those digits more frequently. As there are more neurons than the number of classification categories, several output neurons are expected to assimilate to the same digit, which is evident from 5.14b, where four distinct output neurons learn four different forms of the digit '3'.

5.4 Testing

As the network trains in an unsupervised manner, even though the input data has been learned by the network, there is no label associated with them, and hence the accuracy of the network cannot be estimated. As a result, a labeling step is necessary before applying the training dataset to evaluate its performance. The labeling procedure closely follows the training algorithm, except for the learning operations. The output neurons accumulate in the similar way to the training approach and also produce spikes upon reaching the threshold, and the WTA along with the inhibition is still performed to find the winner neuron. However, no change is allowed in any of the network structure, meaning there is no depression, potentiation or homeostasis. An epoch of the training images are applied and the spikes from the winning output neurons are tallied to find which the number of times each neuron spikes for each digit. A neuron that spikes the most number of times for a digit is labeled to classify that digit. For the trained network of Fig. 5.14a, the tally of the output

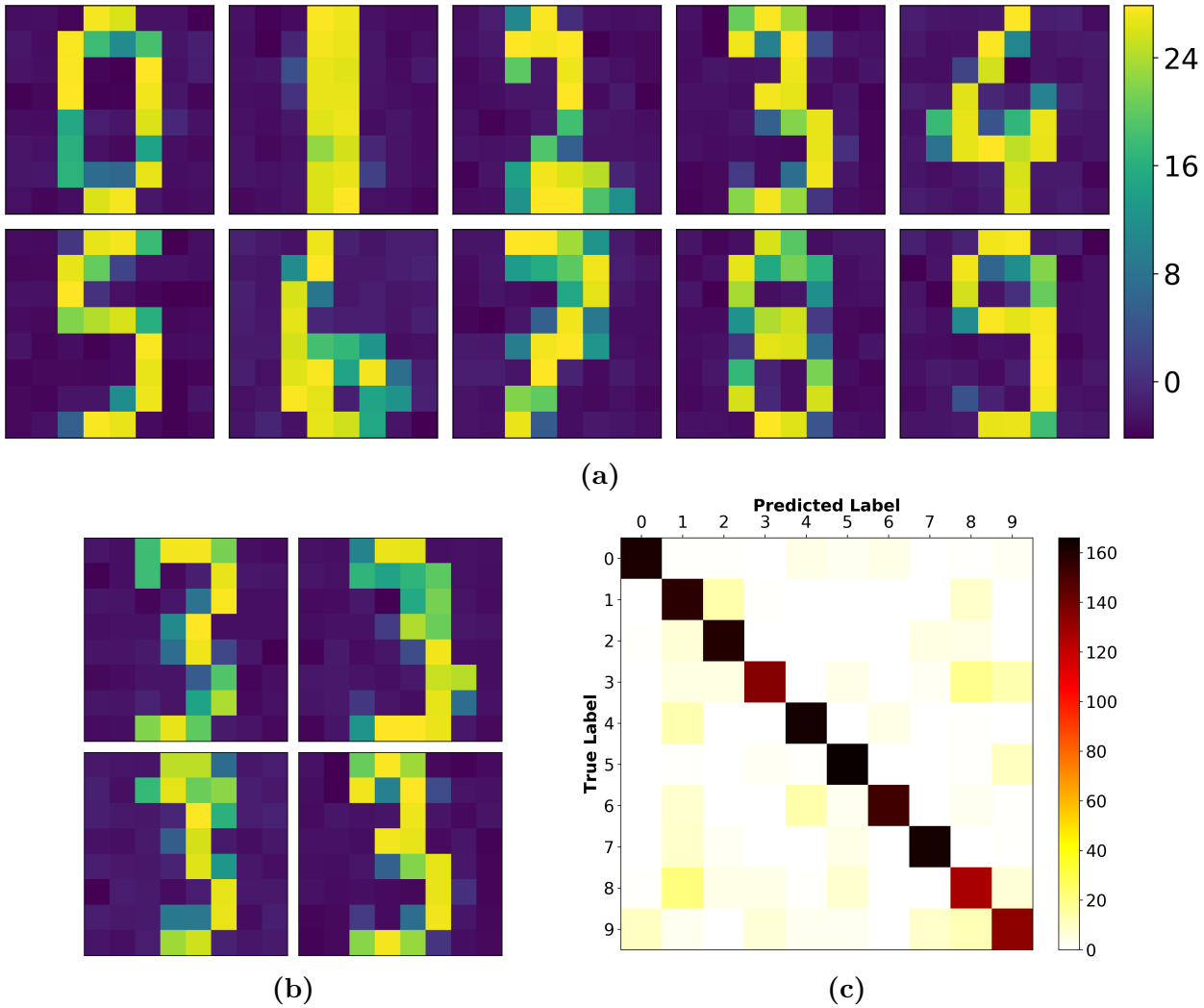


Figure 5.14: (a) The synaptic weights associated with the best performing neurons learning each digit. The conductance in μS is shown with a colorbar. (b) Four distinct output neurons learn four different forms of the digit '3'. (c) Testing result represented by a confusion matrix with the network trained with 50 columns and 10 epochs after applying the testing dataset.

neuron of column 49 was a total of 194 spikes, with 187 spikes for the digit '3', 4 spikes for the digit '5', and 3 spikes for the digit '9'. As the neuron spiked the most number of times for the digit '3', it was labeled to identify that. During the testing phase, if it spikes for input image of the digit '3', the result would be correct. If it produces a spike for any other digit, the result would be listed as being incorrect.

The testing methodology follows a similar approach using the testing dataset. The learning is again disabled, with the output neurons only able to accumulate and produce spikes. The WTA bus selects the winner and with the labels associated with the winning neuron, the performance can be evaluated. Once the whole testing dataset is applied, the overall testing accuracy of the network with 50 columns and 10 epochs was reported to be 84.52%. To analyze the result, a confusion matrix was created to see how the inferred label differed from the true labels, which is illustrated in Fig. 5.14c. Barring few exceptions, majority of the digits were classified with high reliability. The most commonly confused digits were the digit '3' with the digits '8' and '9', the digit '9' with the digits '0' and '8', the digit '8' with the digit '1'. This is due to having similar intensity patterns at the same pixel positions of the images of the dataset.

5.5 Performance Analysis

To analyze the performance of the proposed spiking neuromorphic system, network level parameters such as number of columns of the crossbar, the number of training epochs, output neuron capacitors were varied. In addition to that, the robustness study was performed by evaluation the performance with respect to memristor imperfections such as parameter mismatch, device failure, endurance, and memristor process variations. A clock frequency of 100MHz was used for simulation, with the memristor parameters listed as follows: $HRS = 300K\Omega, LRS = 30K\Omega, V_{thp} = 0.75V, V_{thn} = -0.75V, t_{swp} = 1\mu s, t_{swn} = 1\mu s, \theta_{HRS} = 0.85, \theta_{LRS} = 2.1, \beta_{HRS} = 0.2, \beta_{LRS} = 0.05, C_{HRS} = 6.75, C_{LRS} = 7.29, P_{HRS} = 3, P_{LRS} = 3$. The device level parameters used in this work follows [24] and the fitting parameters of the memristor model uses parameters from [11, 3]. To account for the randomness of memristors'

initial conductance, each analysis includes 25 runs of the same simulations and the reported results are the median of those runs.

5.5.1 Effect of changing Number of Neurons and Epochs

The network level parameters such as the number of columns of output neurons and the number of training epochs was varied to explore their impact on the performance of the network. The input image resolution determines the number of input neurons needed, which is 64 is for the dataset used in this work. At first, the number of epochs was kept constant at 20 and the number of number neurons were changed starting from 10 and going up to 1000, the result of which is plotted in Fig. 5.15a. At first, with only 10 output neurons, the accuracy of the network was 60%. Adding more output neurons results in a rapid increase in accuracy and the network reaches over 80% accuracy with just 50 output neurons. Since then, the rate of increase of accuracy slows down with increasing number of output neurons. The maximum accuracy was found to be 87%, reached by a network having 940 output neurons. The accuracy at with low number of neurons is due to the fact that with lower number of columns, there are not enough output neurons to learn all the different patterns of the same digit. As more output neurons become available, the different forms of the same digit can now be learned and hence accuracy rises rapidly from 10 to 50 output neurons. However, as there is only a few different forms of the same digit, adding more output neuron does not drastically improve the accuracy anymore, and hence the accuracy begins to plateau.

For evaluating the dependency on the number of training epochs, the number of output neurons was kept constant at 20 while the number of training epochs was changed from 0.1 to 200, the result of which is plotted in 5.15b. With only 2 training epochs, the accuracy of the network reaches 80%. This rapid increase of accuracy slows down gradually and reaches 85% with 10 training epochs. The accuracy increase is now saturated and at the end of the 20th training epoch, the accuracy of the network is found to be 87%. With lower number of epochs, the rate of learning is higher with input images contributing to rapid changes in synaptic weight and accumulation. As the neurons keep seeing the same set of images, the learning levels off and the synaptic weights converge on to their expected points, and the

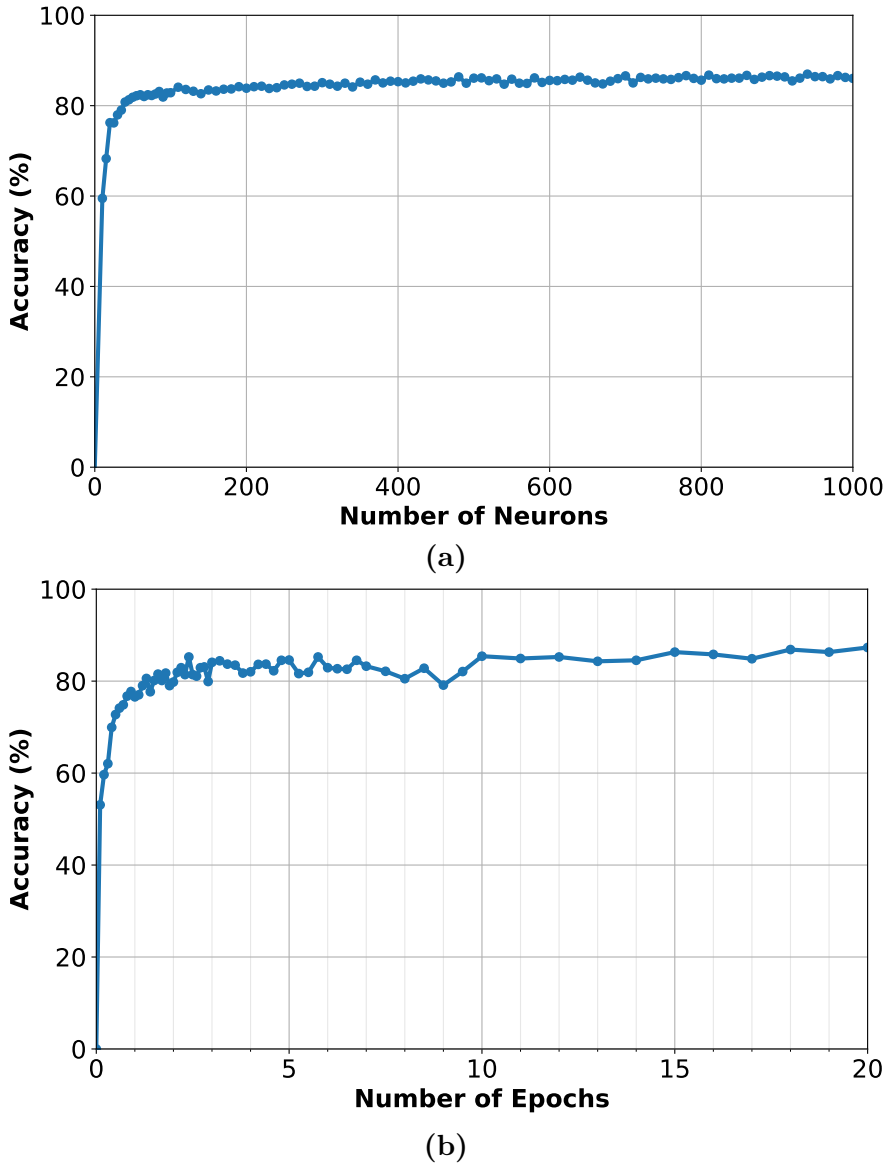


Figure 5.15: Change of accuracy of the designed spiking neuromorphic system with the network parameters. (a) Change of accuracy with increasing number of output neurons with 20 training epochs (b) Change of accuracy with increasing number of training epochs with 200 output neurons.

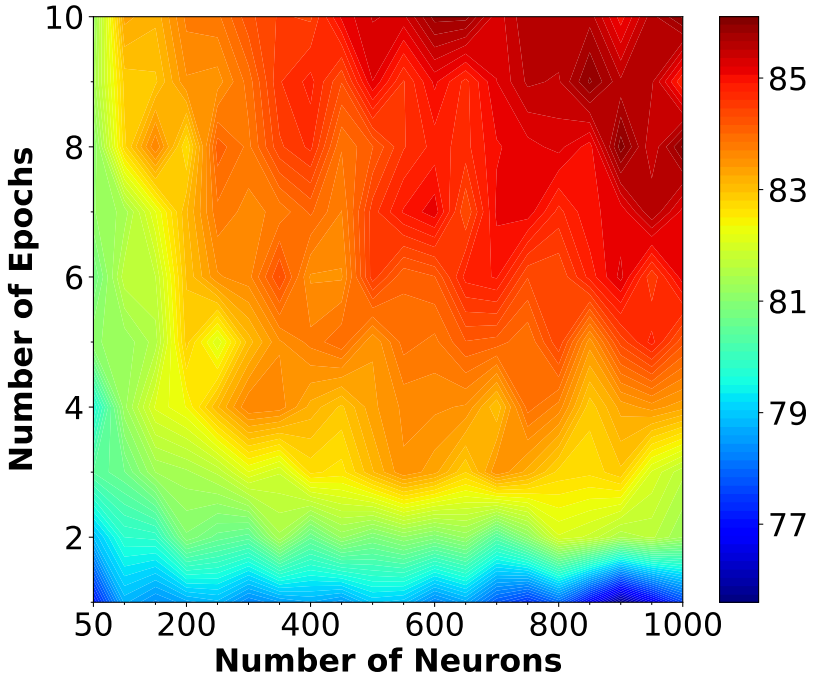
same neurons with same sets of input produces similar spiking patterns, resulting in little to no change in accuracy.

The number of output neurons and training epochs were varied at the same time to show the overall effect on the accuracy of the network, the result of which is presented in Fig. 5.16a. The number of epochs was changed from 1 to 10, in increments of 1 and the number of output neurons was changed from 50 to 1000, with step size of 50. As expected, the accuracy of the network gets diagonally higher, although the increase in accuracy is gradually slower.

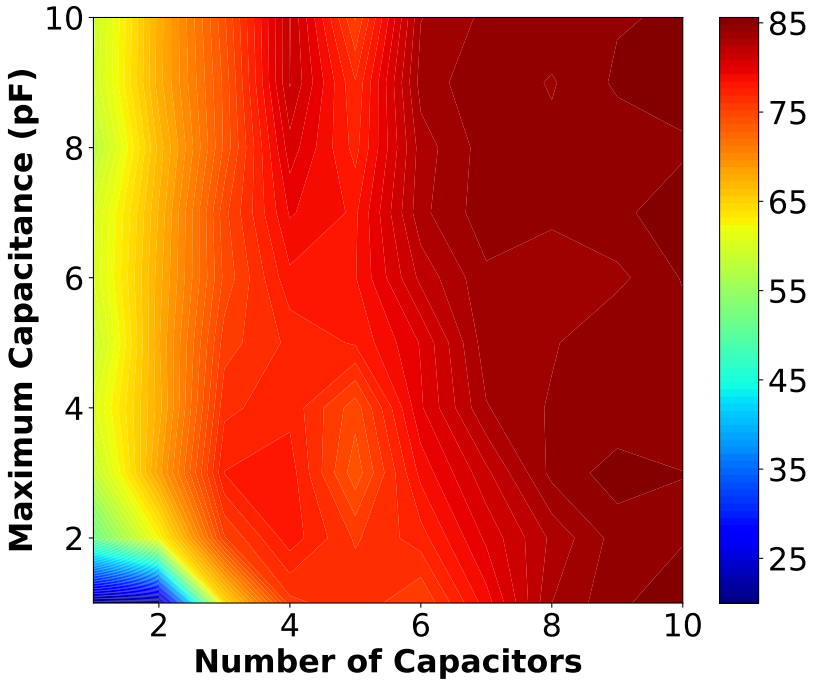
The best performing network among all the simulations runs achieved an accuracy of 90.09% with 1000 neurons and 15 epochs. With 50 output neurons, the median accuracy of the network was found to be 81%, which performs better than the 76.8% listed in [37] and on par with the 81% reported in [71]. The networks used in these works are similar in nature, as they also use a single-layer spiking system, but the results are reported using a different dataset and the output neurons are not readily translated into hardware. Higher accuracy networks have been reported in [82, 34, 101] with unsupervised learning. However, they either use multi-layer SNNs or pre-process the input data to produce easily distinguishable spiking inputs. As a result, direct comparison cannot be made with the method presented here, which employs a single-layer memristive crossbar with lightweight hardware design.

5.5.2 Effect of Increasing Capacitors

The capacitors used in the feedback path of the integrator of the output neuron plays a vital part in learning as the homeostasis mechanism is realized by them. Hence, a study is presented here to quantify their impact on the performance of the designed network. The number of parallel capacitor as well as the maximum capacitance used in the design was varied and the result is shown in Fig. 5.16b. The accuracy of the network increases with both the parameters, but the increase in number of parallel capacitors provide greater benefit. This can be explained by looking at the mechanism of homeostasis in the neuron. The parallel capacitors used in the design controls the resolution of homeostasis, as greater number of capacitors mean there would be more variation in the accumulation rate. With higher



(a)



(b)

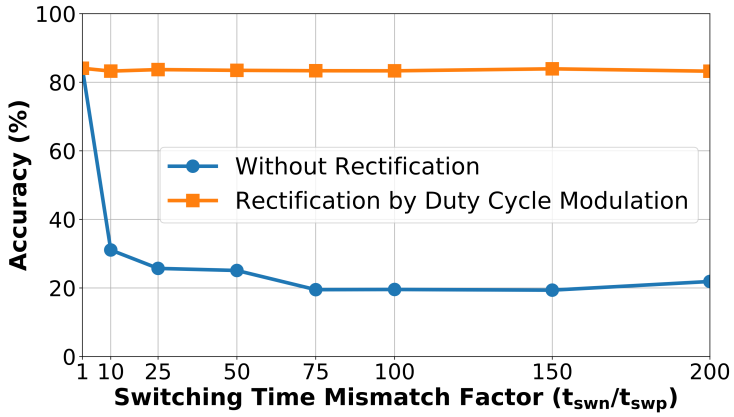
Figure 5.16: (a) The accuracy of the network increases as the number of output neurons and training epochs increase, showing more output neurons and training epochs yielding higher accuracy, but the rate of change slows down. (b) The accuracy of the network increases as the number of capacitors and the maximum capacitance increase for a network with 50 output neurons and 10 training epochs. The increase with number of capacitors is higher than the increase with maximum capacitance.

variation of accumulation rate, the neurons are able to settle on an optimal accumulation rate, and hence the accuracy increases. The maximum capacitance is used to stop a neuron from dominating the spiking competition. As there is a limit to the maximum synaptic weight, that is imposed by the memristor, increasing the maximum capacitance beyond a certain point does not produce significant gain. Hence, there is a greater benefit in increasing the number of capacitors, instead of increasing the max capacitance, which is noticeable in Fig. 5.16b, as the contours become darker on the X-axis faster.

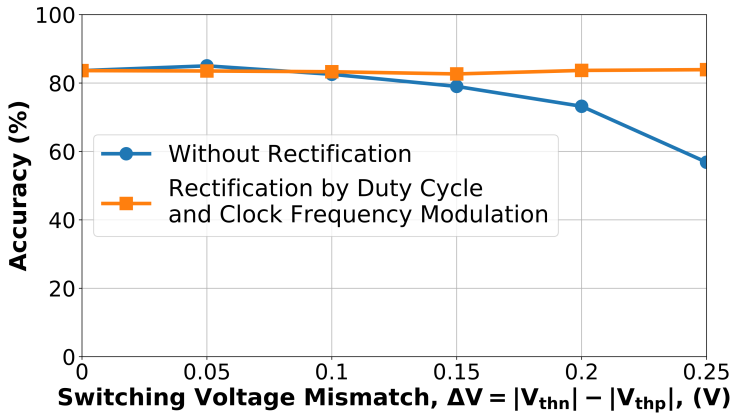
5.5.3 Effect of Switching Mismatch of the Memristors

The memristor switching parameters listed in section 5.5 are assumed to be symmetric, which may not be the case in fabricated memristive devices. There have been reports of mismatch between switching times (t_{swp}, t_{swn}), as well as the switching voltages (V_{thp}, V_{thn}) [12, 88, 52]. At first, the mismatch between the switching times is considered, as the mismatch is reported to be more severe. The switching time for HRS to LRS transition can be two orders of magnitude less than the the switching time for LRS to HRS transition ($t_{swp} < t_{swn}$). The effect of the mismatch is found be severe, as the network performance plunges to 20% when $t_{swn}/t_{swp} = 100$ (Fig. 5.17a). This is due to the drastic memristance change during potentiation because of faster HRS to LRS transition compared to the depression, where the change is from LRS to HRS. To rectify the impact of the mismatch, the applied voltage during the potentiation phase can be adjusted. Instead of applying a full pulse, the duty cycle of the pulse can be reduced to decrease the change in memristance. A reduced duty cycle pulsing scheme is presented in Fig. 5.17c, where a pulse with 50% duty cycle is applied during the potentiation phase, to recover the accuracy. The pulse applied during the depression phase is not changed. With the proposed approach of duty cycle modification, the network was retested, the result of which is presented in Fig. 5.17a showing recovered performance.

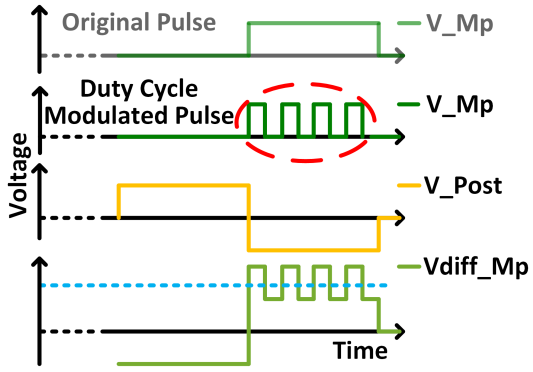
A somewhat lower degree of mismatch in switching voltage has been reported in the literature. The LRS to HRS switching threshold is found to be higher than that of HRS to LRS ($|V_{thn} > V_{thp}$), and the impact of the mismatch on the network performance is shown in Fig. 5.17b. The depression is again weaker compared to the potentiation. For consistency, the potentiation is again weakened by applying the duty cycle modulation



(a)



(b)



(c)

Figure 5.17: (a) Impact of switching time mismatch on the performance of the network and recovery of the performance using a pulsing scheme with duty cycle modulation (b) Impact of switching voltage mismatch on the performance of the network and recovery of the performance using a pulsing scheme with duty cycle modulation and clock frequency reduction. All the networks here has 200 output neurons and uses 10 epochs. (c) Change of pulsing scheme for duty cycle modulation technique. The pulse is applied to the memristor M_p only during potentiation. V_M_p represents the pulse applied to M_p, V_Post represents the postsynaptic node voltage and $Vdiff\_M_p$ represents the voltage difference across the memristor. The dashed blue line represents the memristor switching threshold.

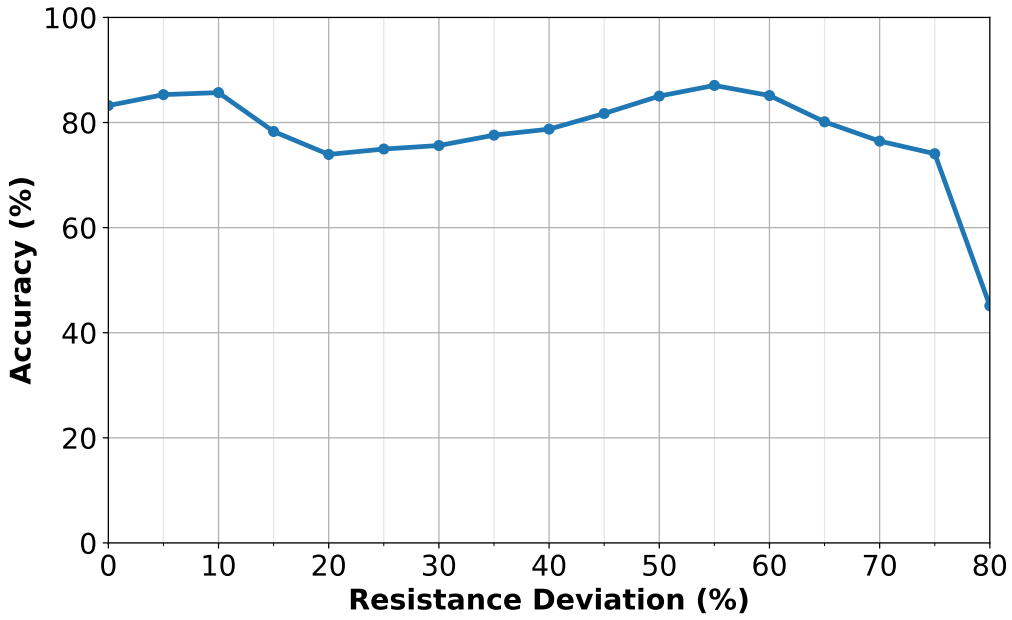
technique described above. However, as both the learning operations are now weakened, the period that the pulses are applied for is increased so that the rate of change in conductance for learning remains equal to the case of symmetric switching threshold voltages. This is achieved by decreasing the clock frequency of the system. By applying both the techniques, the accuracy of the network can be restored, as shown in Fig. 5.17b.

5.5.4 Effect of Memristor Aging

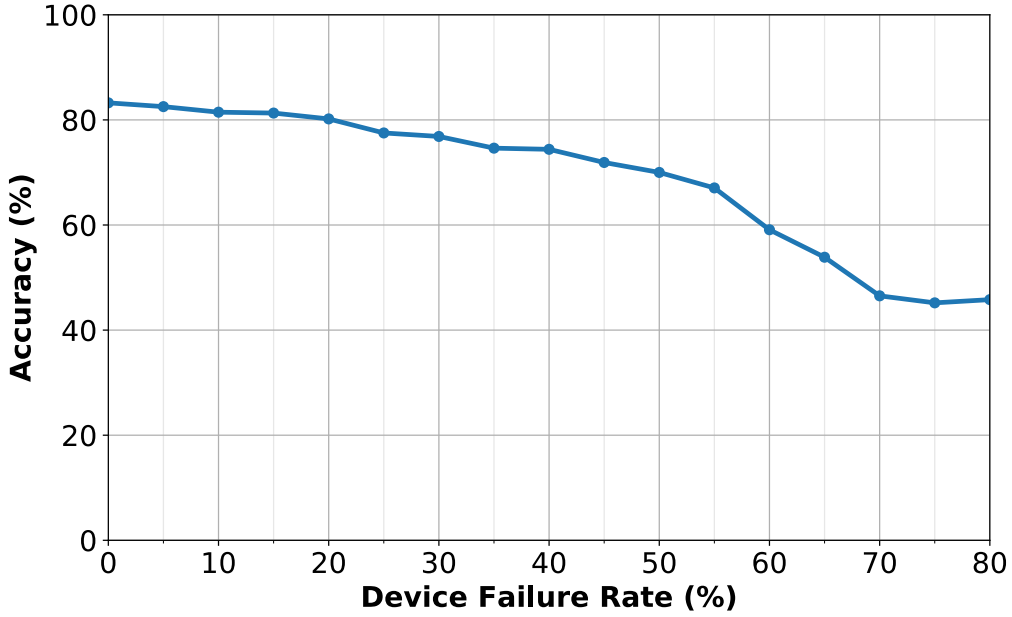
Memristor aging refers to the decreased switching endurance of the memristors due to cycling between HRS and LRS over a significant period of time, which has been reported in [87, 98]. Due to aging, the HRS of the device decreases and the LRS of the device increases, and the window between them decreases. The effect of the switching endurance on the performance of the network is evaluated in Fig. 5.18a with respect to the resistance deviation. Resistance deviation is defined to be the percentage decrease and increase of HRS and LRS, respectively, from the initial HRS and LRS value ($300k\Omega$ and $30k\Omega$, respectively). The unsupervised learning platform proves to be moderately robust against the resistance degradation. The homeostasis mechanism of the output neuron modulates the accumulation rate to account for the shrinking memristor switching window. As a result, the competitive learning is not hampered and 74% is retained even with 75% resistance deviation. With 80% resistance degradation, the switching window of the memristor reduces to only $6k\Omega$, and the accuracy plunges. With more degradation, LRS and HRS become so close that the device does not switch anymore and gets stuck, which is considered a device failure.

5.5.5 Effect of Memristor Device Failure

A memristive device is defined to have failed when the memristance of the device does not change even after a voltage greater than the typical switching threshold has been applied. The failure to switch memristance could be attributed to device fabrication issues [94] or exhaustion of switching endurance [51], as mentioned previously. The performance of the network was evaluated with respect to the failure rate of the devices used to build the crossbar, the result of which is plotted in Fig. 5.18b. The network shows a degree of



(a)



(b)

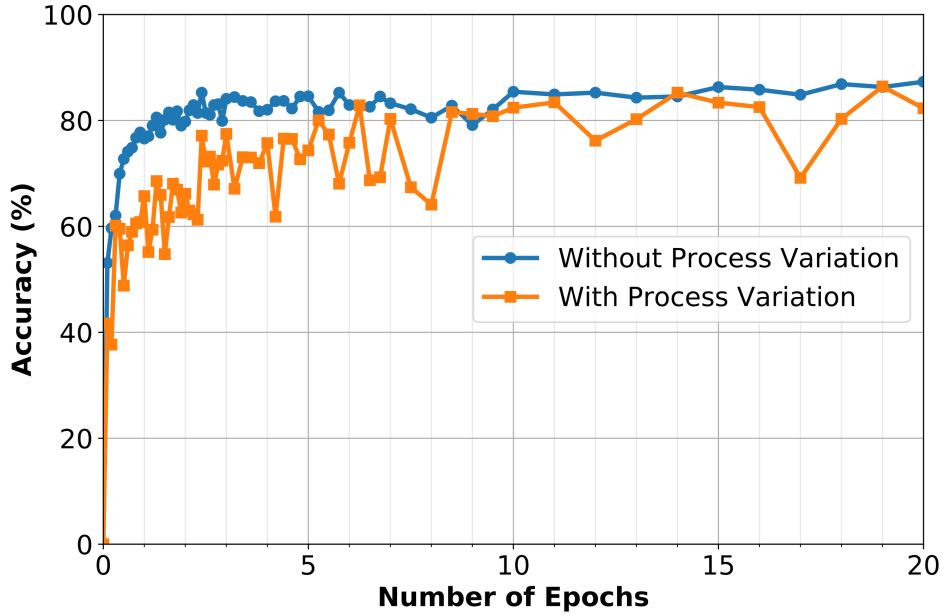
Figure 5.18: (a) Impact of resistance degradation, due to memristor aging, on the performance of the network (b) Impact of memristor device failure on the performance of the network, when the memristive devices appear to be stuck and not switch. All the networks here has 200 output neurons and uses 10 epochs.

robustness against device failures, maintaining over 75% performance when 30% of the devices are assumed to be stuck (failed). The accuracy falls off with more device failures. The initial robustness is due to having two memristors at each synapse, and having one of them (memristor M_n) not switch at all by design, which slashes the effective failure rate seen by the crossbar by 50%. Moreover, having a high number of output neurons provide redundancy which can be leveraged when a few columns fail, and hence the accuracy is not affected so much. However, when the failure rate increases further, more and more columns keep failing and as a result, the accuracy of the network plunges.

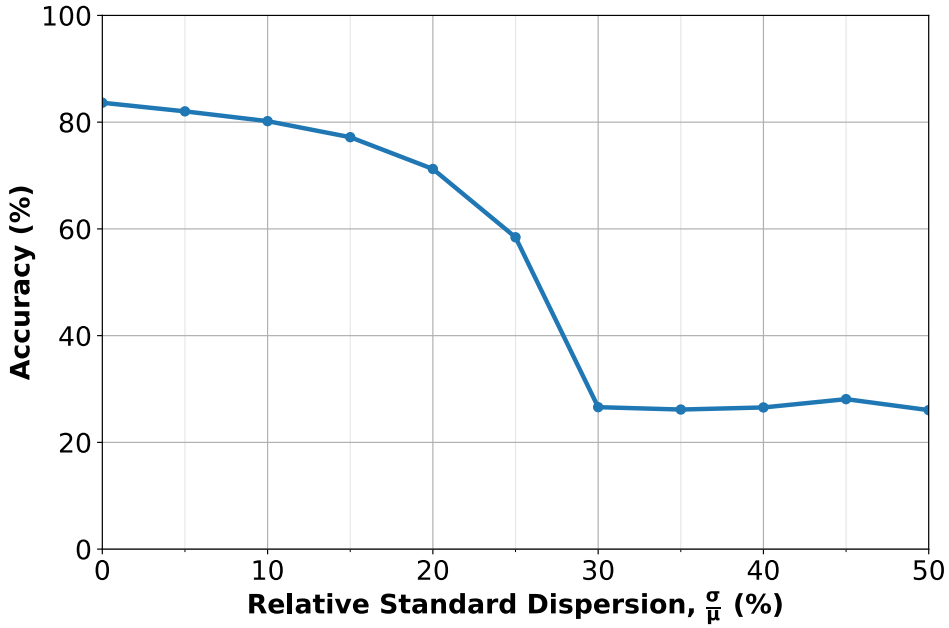
5.5.6 Effect of Process Variation

Memristor device fabrication process technology is not as mature as the state-of-the-art CMOS devices, and hence there is considerable variability in the characteristics of the devices [70]. Some major challenges contributing to the process variations include oxide thickness fluctuations (OTFs), random discrete dopants (RDDs), line-edge roughness (LER) etc [62, 27, 72]. The process variations affects the parameters of the device such as HRS, LRS, V_{thp}, V_{thn}, t_{swp}, and t_{swn}, that are used to generate the simulation results. The typical standard deviation of the parameters has been reported in [88] as follows: $\sigma_{HRS} = 20\%$, $\sigma_{LRS} = 10\%$, $\sigma_{t_{swp}} = 5\%$, $\sigma_{t_{swn}} = 5\%$, $\sigma_{V_{thp}} = 10\%$, $\sigma_{V_{thn}} = 10\%$. Including the variation of the parameters, the performance of the network was analyzed by varying the number of epochs to obtain the learning behavior and compare to the case of Fig. 5.15b. The result plotted in Fig. 5.19a shows that the the network achieved performance of 86% accuracy with 19 epochs with process variations. However, the network needs more epochs to learn and reach a stable classification accuracy compared to the network with process variation.

The performance of the network on the face of process variation was further analyzed by varying the relative stand deviation (RSD=$\frac{\sigma}{\mu}$) of the memristor device parameters from 5% to 50%, the result of which is shown in Fig. 5.19b. There is a gradual decrease in accuracy of the network, going down to 77% with 15% of relative standard dispersion (deviation). However, after that point, the performance of the network plunges rapidly, which is due to the high degree of parameter variation, as well as memristive device failures occurring due to process variation. A histogram of the HRS and LRS distribution with a relative standard



(a)



(b)

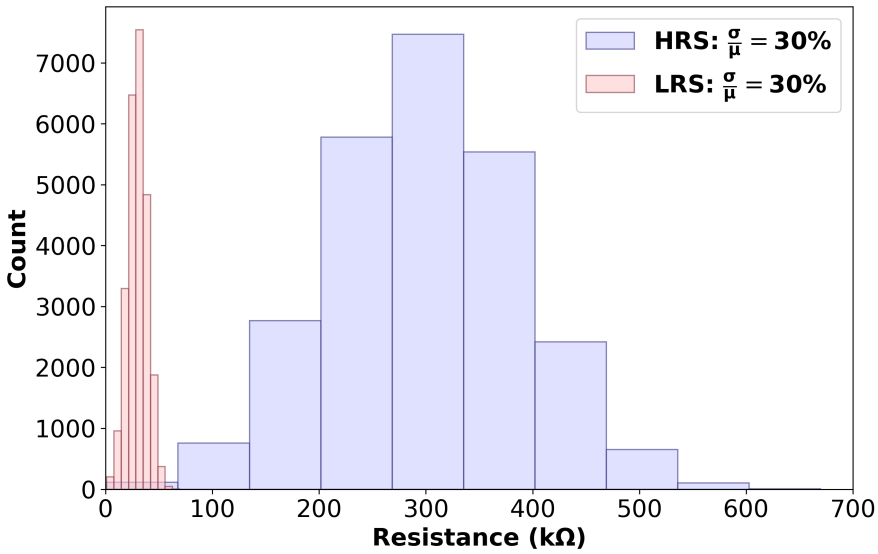
Figure 5.19: (a) Impact on learning speed of the network considering the effect of process variation with 200 output neurons (b) Performance of the network with process variation affecting all the memristive device parameters with 200 output neurons and 20 training epochs.

dispersion of 30% is given in Fig. 5.20a, which shows that the switching window of the memristor (HRS/LRS) could go up to 700, compared to 10 in the typical case. This may cause some of the columns to have much higher total effective weight compares to others, and make the connected neuron dominate the competition even after having the described homeostasis plasticity. Process variations also give rise to device failure, as some devices may have their HRS equal to the LRS, meaning they cannot switch anymore. The failed devices also contribute to the network's poor performance at higher RSD. The increase of failure rate with RSD is shown in Fig. 5.20b.

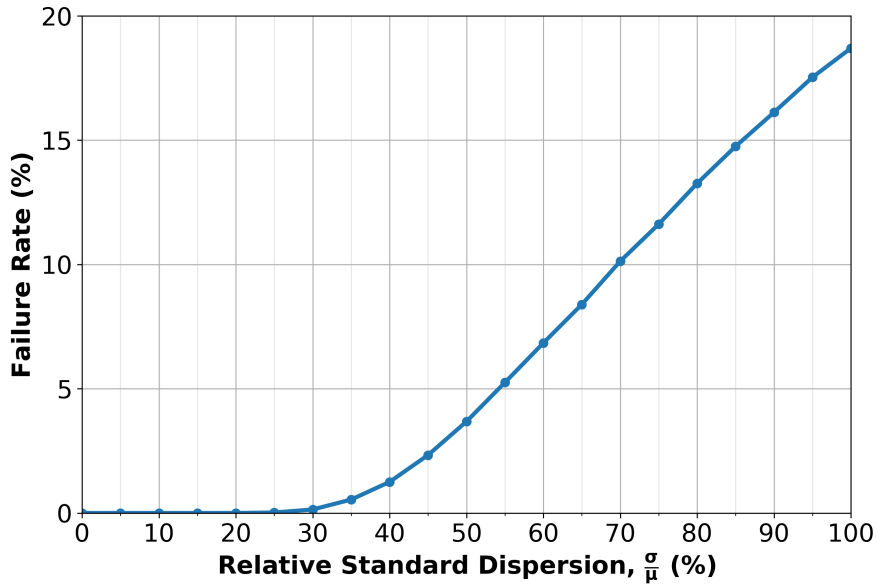
5.5.7 Area Overhead and Energy Consumption

Energy per spike per synapse is the metric that is used to evaluate the energy consumption of the proposed design. The energy is derived by the measuring current from the supply rails during different phases of operation and integrating the current multiplied by the supply voltage over the whole duration of the spike. The energy is averaged over the number of synapses used in the network to obtain the energy consumed per spike per synapse, to be compared to the same metric provided in the literature, which can be seen from Table 5.1. The obtained energy number is on par with the state-of-the-art designs, while providing extra features like homeostatic plasticity in the neuron. The fully digital STDP learning mechanism and the digital control block of the neuron leads to the energy efficient implementation of the proposed design.

In terms of area overhead, the input neuron is a purely digital block, consuming only 500 μm^2, while the output neuron is a combination of digital and analog circuitry, consuming 2350 μm^2 of layout area, which is comparable to the designs shown in other works of Table 5.1. The majority of the layout area is taken up by the capacitors used to implement homeostasis, which is an extra feature of the designed neuron compared to other works. This area is calculated with capacitors of 1pF, 0.3pF, 0.6pF, 0.9pF, and 8pF which require 53.19% of the total neuron area. If the memristive device could be made with higher HRS and LRS value, the current into the neuron would decrease and capacitors with lower capacitance can be used. In [48], devices with LRS-HRS of 250kΩ-2.5MΩ have been demonstrated which would reduce the capacitor area by 90%.



(a)



(b)

Figure 5.20: Effect of process variation on switching window and devic failure (a) HRS and LRS distribution of a 64 × 200 network containing 25600 memristors with 30% RSD. (b) Increase of memristive device failure with respect to increasing RSD.

Table 5.1: Area overhead and energy consumption of relevant memristor based spiking neuromorphic systems.

	Energy (pJ/Spike/Synapse)	Area (μm^2)	Memristor used
This work	3.16	2350(65nm)	30kΩ-300kΩ
[75]	2.4, 8.1	1393(65nm)	15kΩ-150kΩ, 2.5kΩ-12kΩ
[24]	7.69	2000(65nm)	30kΩ-300kΩ
[96]	9.3	10000(180nm)	1000 synapses with 1MΩ each
[42]	36.7	——	70Ω-670Ω

5.5.8 Other Design Considerations

The performance of the proposed network in terms of accuracy (Figs. 5.15, 5.16a) is found to be higher than 83% of [95] and 84.75% of [75], which uses the same handwritten digits dataset. However, this works also deals with analyzing the robustness of the proposed unsupervised learning platform against memristive device imperfections. Similar work presented in [71] achieves immunity against process variation of up to 50% relative standard deviation. However, the software simulation considered the HRS/LRS ratio to be 10000, which is impractical considering fabricated memristors [52]. Another work in [37] varied two parameters of the memristive device individually by 10% RSD, with performance degradation of 1.06% and 1.56%. Here, the combined accuracy degradation was found to be 3.45% at an RSD of 10% of all 6 parameters (HRS, LRS, positive and negative switching time, and positive and negative threshold) of the memristor model.

The dynamic range of pixel values as well as the encoding scheme also impacts the the performance of the proposed network. As the dynamic range increases, the pulse width applied to the memristors will be greater, resulting in increased changes in synaptic weight. Loss of the granularity of the synaptic weight adjust can result in lower performance of the network because it is now difficult to find the global minima. Therefore, to increase the dynamic range, a higher frequency needs to be used, so that even if the number of pulse widths is greater in terms of clock cycles, the time of the pulse itself will be shorter, thus keeping synaptic granularity. For the presented network, it was found that for a dynamic range of -4 to +4, the best frequency is 100MHz. The network throughput with 100MHz frequency for training and testing was found to be 7.69 million and 20 million inputs per second, respectively.

The effect of parasitic resistance on the crossbar wires was also considered. The interconnect resistance associated with the wires for the 65nm IBM PDK used in this work was found to be 0.7Ω per memristor, yielding the worst-case series resistance of 230Ω, which is associated with the memristor at first row and last column of 64×200 crossbar [63] network. The LRS of the memristor is 30kΩ, which is two orders of magnitude higher than the worst case series resistance, and hence the performance drop of the network due to

the voltage drop across parasitic resistors is trivial [47]. Running the simulation with the parasitic resistance, the results confirm the validity of the assumption. As can be seen from Fig. 5.21, the network performance after considering the effect of interconnect resistance is almost identical to the ideal crossbar setup, with the Mean Square Error (MSE) being 0.17.

An issue with analog pulsing scheme is susceptibility to noise and signal degradation, specially for large crossbars. As the designed network uses fully digital pulsing scheme, it provides immunity to noise, and hence is proven to be scalable. Sneak path is another common issue plaguing memristive crossbars, which is tackled here by using half-rail voltages across memristive devices. For accumulation, the memristor M_p and M_n are pulled to positive and negative rails, while the postsynaptic node is at mid-rail voltage. This leads to half of rail-to-rail voltage drop across the devices, decreasing the sneak path current [75]. The effect of sneak path is further reduced by driving specific voltages to memristors and not letting them be floating [47].

Convolutional Neural Networks (CNN) have been reported to achieve near human level performance in classification tasks [39], yielding better accuracy than the proposed spiking neuromorphic system. However, spiking neural networks leverage sparsity in the spiking events and have been proven to be energy efficient by an order of magnitude for training [22] compared to CNNs. Furthermore, the spiking neural network proposed here is amenable to dedicated hardware implementation, making the approach more power efficient by using memristors as synapses. To achieve better recognition accuracy or more complex pattern recognition tasks, the network proposed can be scaled up with additional layers. Moreover, the single-layer SNN presented can be combined with a Reservoir or other type of spiky Recurrent Neural Network (RNN) to improve the performance of the system.

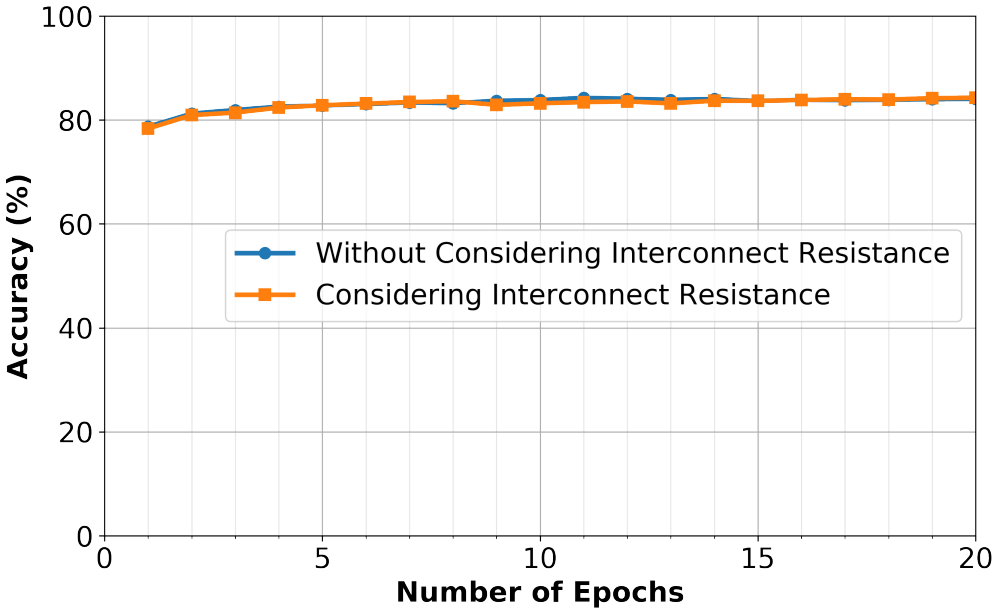


Figure 5.21: Effect of interconnect resistance on the network performance with 200 output neurons.

Chapter 6

Application of Memristive Spiking Neuromorphic System: SPAD-based Vision Sensor

Chapter 5 describes the proposed memristive spiking neuromorphic system, which was employed to classify image data by converting image intensity to spiking events. However, a more biologically plausible and energy efficient approach could be followed, wherein the an image is expressed as spatio-temporal data, producing sparse spiking events that directly drive the input neurons. This can be done using a Single Photon Avalanche Diode (SPAD) based Vision Sensor, enabling on-chip memristive spiking neuromorphic processing with very low energy consumption.

6.1 SPAD based Vision Sensor

Single photon avalanche diode (SPAD) is referred to as a Geiger mode avalanche photodiode capable of detecting a single photon with high accuracy of arrival time of the photon. This allows a SPAD based vision sensor to capture high-speed images at very low lighting conditions and hence, is used in multitude of applications such as autonomous vehicles, robots, healthcare, space, remote sensing and security [100, 13, 90, 79, 6].

Conventional approach to implementing a SPAD based vision sensor is demonstrated in Fig. 6.1, where the time of flight information of the reflected photon is encoded by a time-to-digital converter (TDC). The TDC data is passed on to an off-chip processing unit that builds a histogram to reconstruct the target image and then proceeds to processing the images as required. This off-chip transfer and processing of data leads to increased latency and power consumption [81].

Additionally, SPADs are inherently event based and asynchronous, which makes them suitable for asynchronous address event representation (AER) based readout scheme [80]. AER readout with SPAD and non-SPAD based vision sensors has been presented in [69, 36, 28, 15]. However, all of the approaches rely on off-chip processing resulting in high computational overhead and latency. A frame-based sensor array and event based processing was proposed in [6], but requires conversion from frame to events, adding complexity to the system. Hence, there exists a need for an efficient on-chip processing methodology for the data generated by AER readout of the SPAD based vision sensor system.

6.2 Proposed SPAD based Vision Sensor with Memristive Neuromorphic System

To overcome the shortcomings of existing approaches, in this work we propose a SPAD based Vision Sensor with on-chip processing, connecting the AER readout and SPAD output to the memristive spiking neuromorphic system via a simple decoder circuit, as shown in Fig. 6.2.

6.2.1 SPAD based Vision Sensor with AER Readout

At the heart of the proposed system is the SPAD based Vision Sensor with AER Readout presented in [81, 80]. A pulsed laser illuminates the target image by scanning over it, resulting in photon reflected by the target. The reflected photons are captured on the SPAD, which is a part of a pixel of the vision sensor. The other components of the pixel include a quenching circuit, a Schmitt trigger, an analog counter, and an event generator. Upon the arrival of the

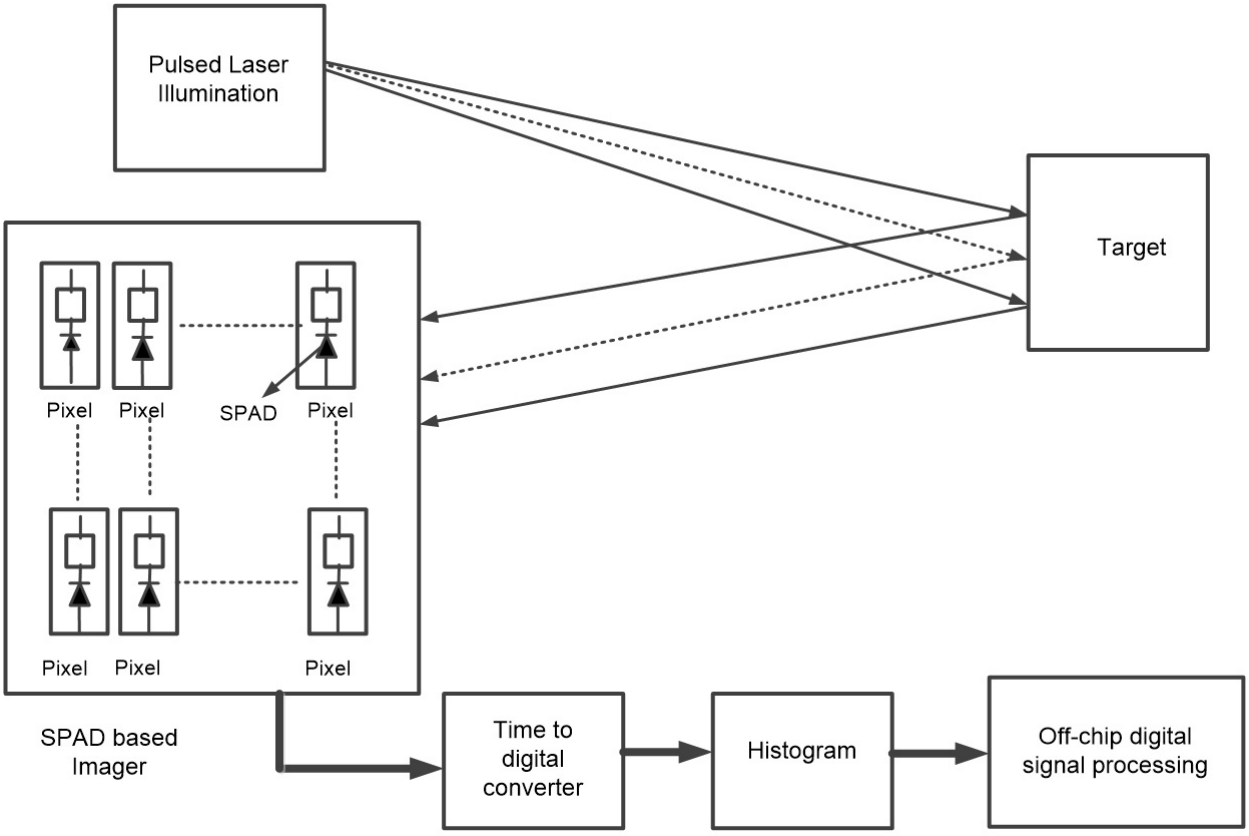


Figure 6.1: Conventional SPAD based vision sensor [81]

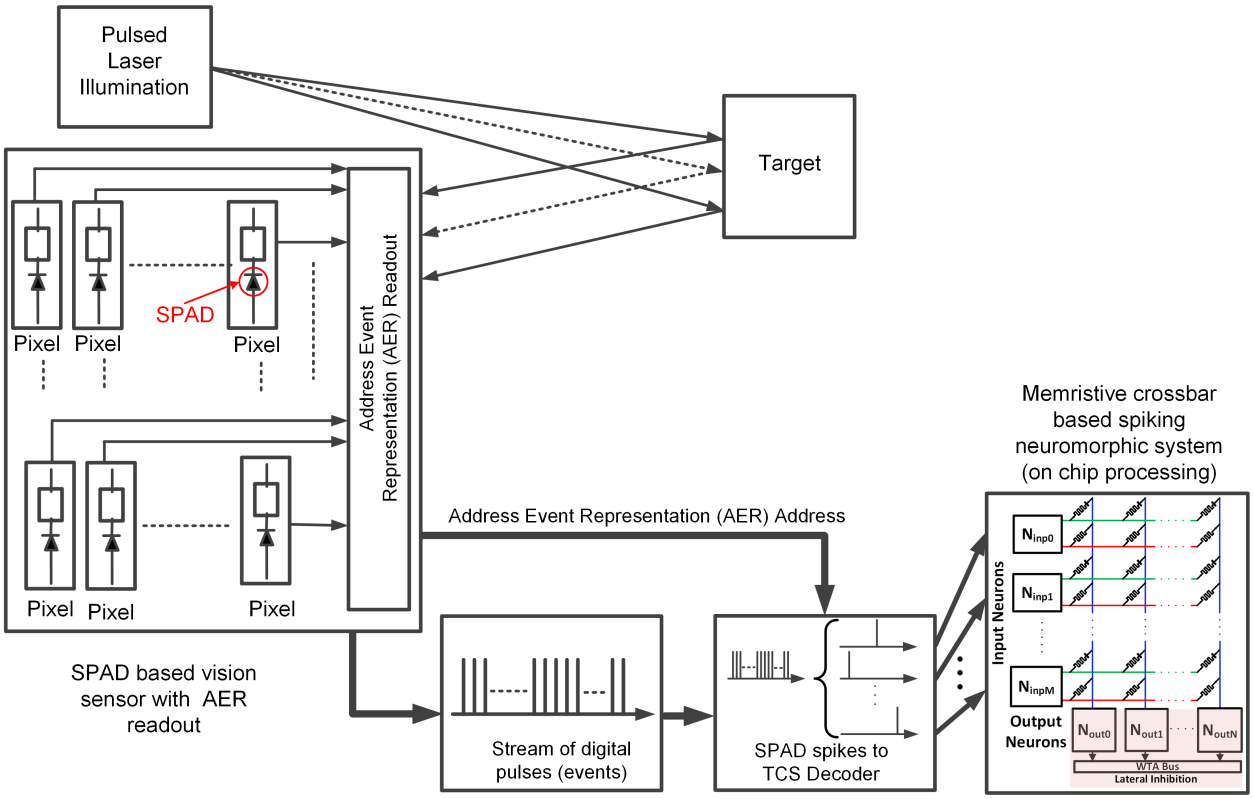


Figure 6.2: Proposed SPAD based vision sensor with AER readout connected to spiking neuromorphic system for on-chip processing via an interfacing unit to convert SPAD pulses into Temporally Coded Spikes (TCS).

photon, these components generate a clean digital event, which is transmitted to the AER system. The AER system generates the address of the pixel where the photon was captured and outputs the row and column of the pixel. The final output of the SPAD based vision sensor is the stream of digital events along with the row and column address of each pixel that generated the event. For the proposed system, the dataset consists of 8×8 images. Hence, for one image, the sensor outputs a maximum of 64 digital spiking events and 64 row and column addresses, 3-bit each to account for 8 rows and columns.

6.2.2 SPAD Pulse to Temporally Coded Spike Decoder

The SPAD based vision sensor generates streams of digital pulses, the timing of which depends on the intensity of the pixels of the target image. The higher the intensity of the pixel, the sooner the photon arrives and hence, the earlier the SPAD event is generated. This pulsing scheme is different than the TCS scheme adopted for the memristive spiking neuromorphic system, which was described in 5.1.1. Moreover, SPAD event generation is asynchronous, but the neuromorphic system used in this work is synchronous. Furthermore, the AER readout generates row and column addresses of the SPAD pixel, but the input neurons of the neuromorphic system are indexed in a single column. To overcome these challenges, an interface between the sensor and the neuromorphic system is needed, which enables the conversion of vision sensor outputs to TCS scheme of the neuromorphic system. The core components of the SPAD to TCS decoder are a 1-to-64 de-multiplexer to transmit the SPAD pulse to the correct input neuron, a pulse synchronizer to synchronize the asynchronous SPAD event, and a delay block, to efficiently convert the SPAD events to TCS. The block diagram of the SPAD pulse to TCS decoder is shown in Fig. 6.3.

The de-multiplexer is used to de-multiplex the SPAD signal to the corresponding input neuron, according to the row and the column address generated by the AER readout. The row and the column addresses are 3-bits each and the the de-multiplexer select pin requires 6-bits to de-multiplex to 64 input neurons. As the input neurons are arranged in row-major-order, the row addresses are tied to the 3 MSBs and column addressed are tied to the 3 LSBs of the select pin.

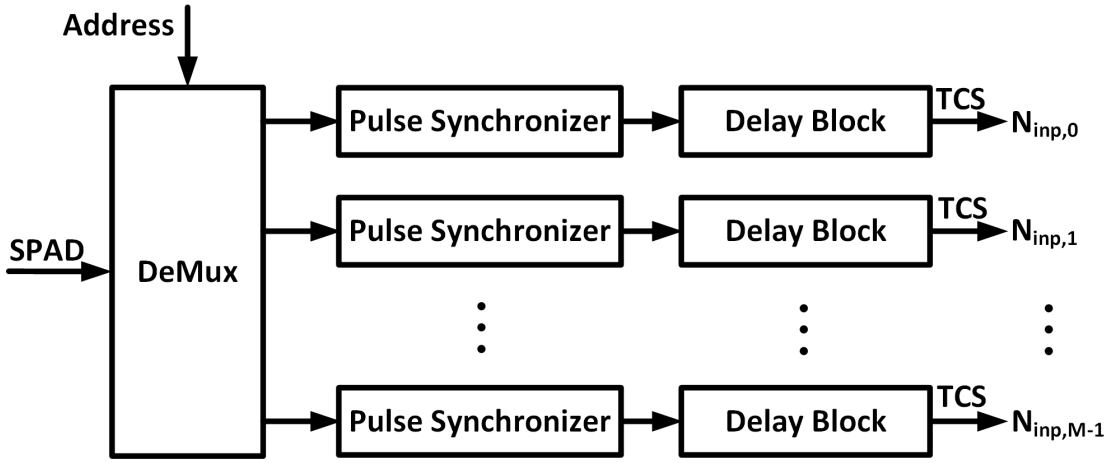


Figure 6.3: Block diagram of SPAD to TCS decoder. The DeMux provides the asynchronous SPAD spike to the appropriate rows, which is first synchronized and then converted to the TCS by Pulse Synchronizer and Delay Block, respectively.

Once the correct input row has been identified and the asynchronous SPAD event is passed on to the corresponding row, the signal is synchronized to the clock. As SPAD event pulse width maybe narrower than the clock period, the asynchronous signal is first provided to a rising-edge detector. Then a Level-to-Pulse converter is used to generate the synchronous SPAD event that is required for TCS conversion. The timing of the synchronized SPAD event and TCS is illustrated in Fig. 6.4. Here, SPAD events for different pixel intensities are shown, assuming that SPAD event collection time window is discretized to 8 timeslots to account for 8 discrete pixel intensity values. As mentioned before, the higher intensity pixels of the target image generate SPAD events earlier. As we decrease intensity from +4 to -4, the SPAD events occur at timeslots 1 to 8, respectively. To convert to TCS, the SPAD events need to be delayed by some clock cycles, the number of which depends on the intensity of the pixel. The TCS timeslots begin 4 clock cycles after the SPAD timeslots for maintaining causality. The delay required is calculated by looking at the SPAD timeslots for both SPAD events and TCS events of the same pixel intensity. For example, the pixel intensity of +3 generates a SPAD spike at SPAD timeslot 2, but for TCS scheme it should generate a spike at SPAD timeslot 7. Hence, the SPAD event is delayed by 5 clock cycles. Table 6.1 lists the delay required for each pixel intensity of the target image. The re-configurable delay block shown in Fig. 6.5 houses a counter to keep track of the timing of the SPAD event and stops counting when there is an SPAD event. Then the number of delay required is generated by the delay decoder and the SPAD event is delayed by that amount. A binary to thermometer decoder drives the delay line to select the number of delay units required to realize a given clock cycle of delay. Thus, the TCS signal is generated from the SPAD event. The layout of the designed input neuron with IBM 65nm CMOS10LPe PDK is shown in Fig. 6.6.

6.2.3 Memristor Crossbar based Spiking Neuromorphic System

With the SPAD to TCS decoder, the SPAD data does not need to be transferred to any off-chip processor and can be directly used as inputs to the designed memristive neuromorphic system. The network used in 5 is connected to the decoder to process the SPAD data on-chip. However, the SPAD data generated from the handwritten digit dataset [35] is degraded in quality, as there will be some loss due to the inherent randomness in spike generation of the

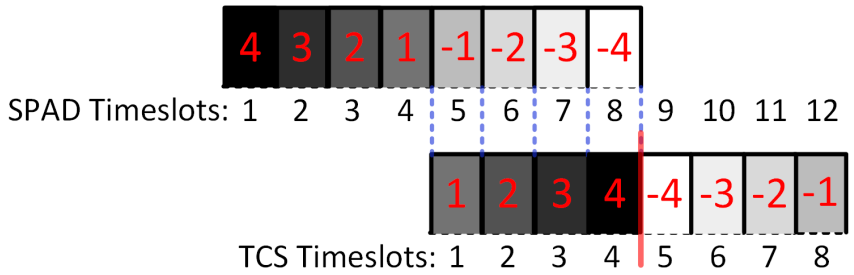


Figure 6.4: Synchronous SPAD event to TCS conversion. Looking at the position of spikes at SPAD timeslots, the delay required for conversion can be calculated.

Table 6.1: The delay required for each pixel intensity to generate TCS from SPAD events.

Intensity	**Delay**
+4	7
+3	5
+2	3
+1	1
−1	7
−2	5
−3	3
−4	1

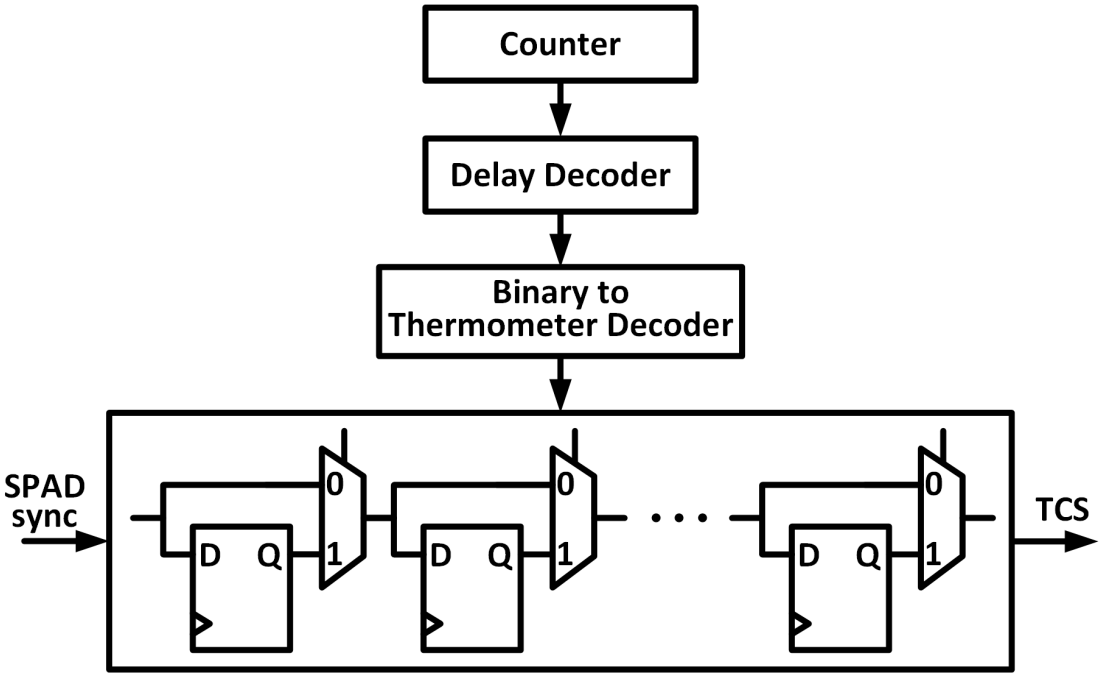


Figure 6.5: The Delay Block to delay the synchronized SPAD event by the required amount. The counter tracks the timeslot of the SPAD event. When the SPAD event occurs, the delay decoder converts the count value into the required delay value. The binary to thermometer decoder generates the select signals for the delay line using the delay value provided by the delay decoder to produce the TCS.

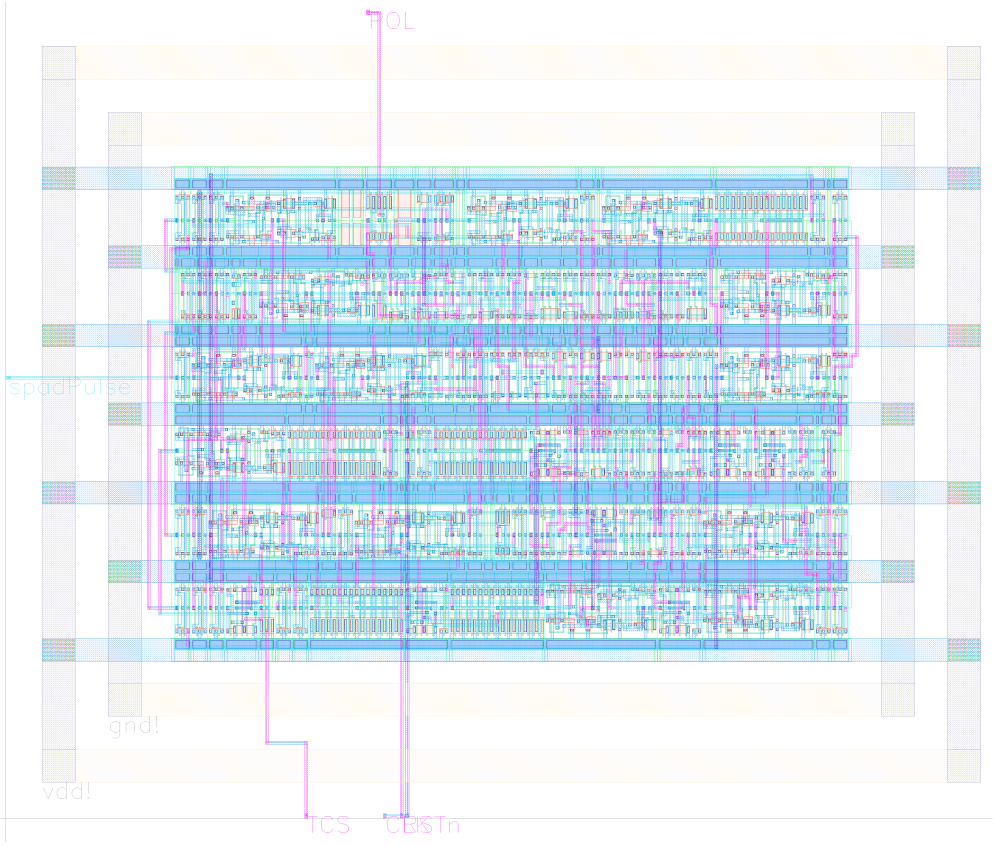


Figure 6.6: SPAD to TCS decoder layout designed with IBM 65nm CMOS10LPe PDK with an area of $43\mu m \times 30\mu m$.

SPAD sensor. Hence, the performance of the network needs to be optimized by leveraging the parameters available to the designer.

6.3 Results

The developed vision sensing system is the first SPAD based vision sensing system with integrated memristive spiking neuromorphic system adopting the benefits of SPAD's high quantum efficiency and energy efficiency of memristive spiking neuromorphic processing. The proposed SPAD based vision sensor includes biologically inspired address event representation (AER) readout to generate asynchronous digital address and spiking events at the output, which is readily applied to the on-chip neuromorphic processor via a simple interface. The dataset used in chapter 5 was converted to SPAD events, but resulted in a Normalized Mean Absolute Error of 5.6% relative to the previously encoded data. Although the input data has deviated from the original data, the various design parameters are tuned to optimize the performance of the spiking neuromorphic system. Number of neuron, epochs, capacitors and the maximum capacitance values were varied and the results are shown in this section.

As before, the number of input neurons were fixed at 64, assigning one for each pixel. The number of output neurons were varied from 10 to 1000, in steps of 10 neurons, keeping the number of epoch constant at 10. Due to the inherent randomness of memristor initialization, each simulation was run 25 times and median accuracy of the result of the runs are presented in 6.7. The networks starts out at an accuracy of 59% with only 10 output neuron and quickly improves to 80% with 80 output neurons. As we keep on increasing the number of neurons, the accuracy increases very slowly with decaying returns and eventually saturates with minor noise. At 550 neurons, the network reaches 87% for the first time. Among all the runs, the absolute maximum accuracy reached by a network was 89.65% with 530 neurons.

To optimize the network for number of epochs, the number of output neurons was kept constant at 200 and the number of epochs was varied from 0.1 to 20, collecting accuracy data more frequently at lower number of epochs to show the learning trend. The result is plotted in Fig. 6.8, showing the accuracy reaching 78.85% just after one epoch of learning.

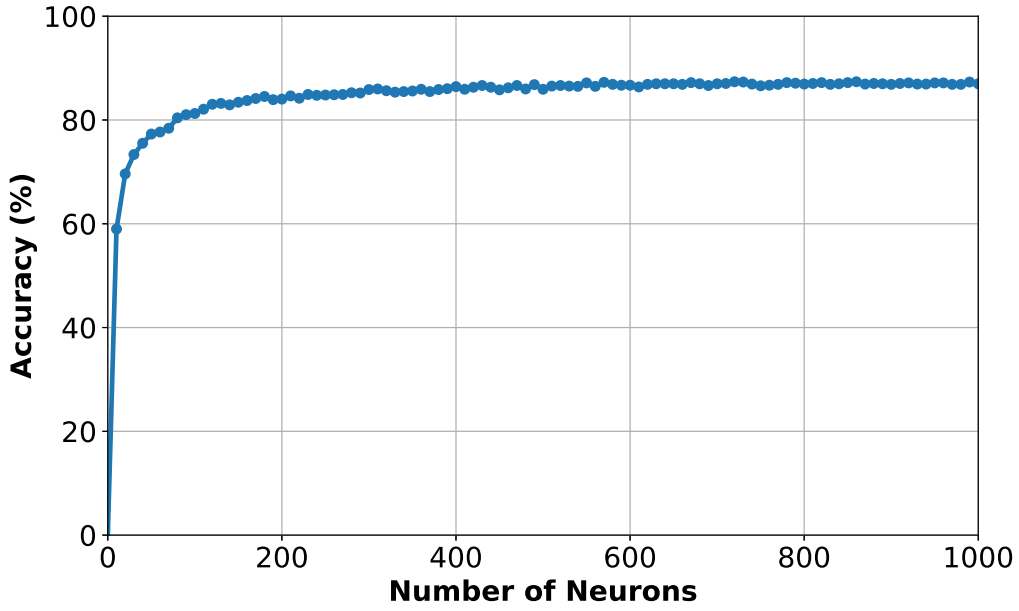


Figure 6.7: Change of accuracy with increasing number of output neurons of the spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor with 10 epochs.

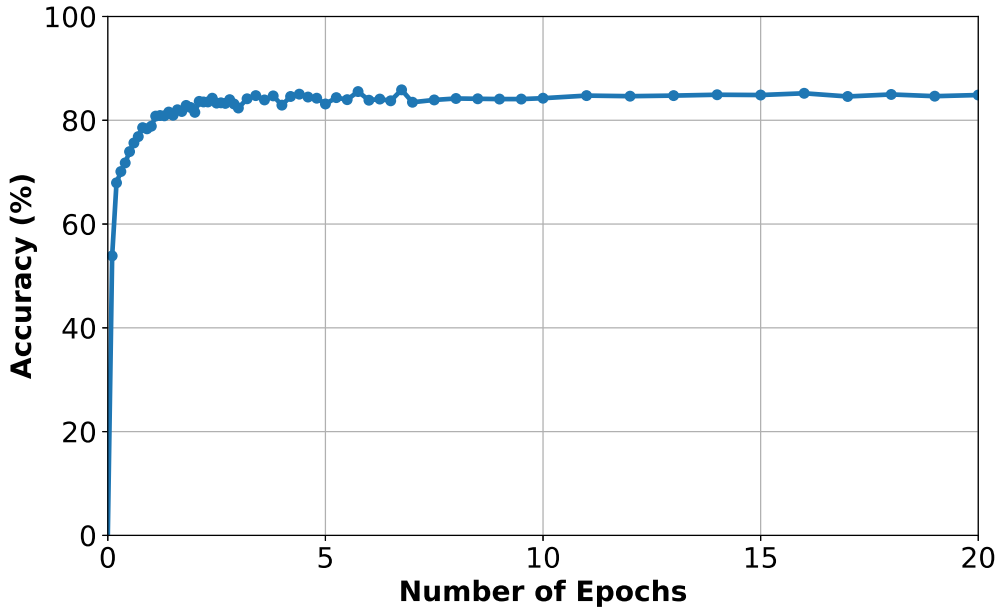


Figure 6.8: Change of accuracy with increasing number of epochs of the spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor with 200 output neurons.

A similar trend is observed here compared to the result of increasing neurons. At first the rate of learning is very high, but after the network has learned most of the data patterns, the accuracy settles around a certain value and shows minor deviations. With only 2 epochs, the accuracy goes above 80% (81.52%). At later stages, with more epochs, the accuracy does tend to rise, but with very small slope. Most of patterns are already learned, and providing the same set of data, does not significantly improve performance after a certain point. The accuracy settles around 85%, with maximum median accuracy reaching 85.2% with 16 epochs.

For the sake of completeness, the number of neurons and epochs were varied at the same time and results are shown in Fig. 6.9. The number of epochs were changed from 1 to 10, in increments of 1 and the number of neurons were changed from 50 to 100, in increments of 50. This result also shows the improvement in accuracy with both the number of neurons and epochs and the diminishing returns. The maximum median accuracy was 87.31% with 950 neurons and 9 epochs, whereas the absolute maximum accuracy of all the results collected was found to be 89.54% with 850 neurons and 10 epochs.

The circuit level parameters to fine-tune the performance of the neuromorphic system are the number of capacitors used and their capacitance. The number of capacitors as well as the largest capacitance value was varied to optimize performance, the result of which is presented in 6.10. The accuracy increases with both the parameters, but the most gain is brought upon by the number of capacitors used, substantiated by the contours growing darker red faster on the X-axis. As discussed before, having more capacitors enables the neuron to have more resolution in homeostatic plasticity. Now that the data is degraded by 5.6%, having more granularity provides significant gain. This is also beneficial from a circuit level perspective, as multiple small capacitors consume less area compared to one huge capacitor.

Previously similar characters recognition task obtained from an event based sensor was tackled by [66] and similar task was chosen here to provide a direct comparison with previously publish works, as shown in Table 6.2. In [66], Orchard et al. reported an accuracy of 84.9% $\pm$ 1.9% for the digits (0-9) and characters(A to Z) recognition using a four layer hierarchical SNN model. Although the achieved accuracy (84.47%) of our proposed

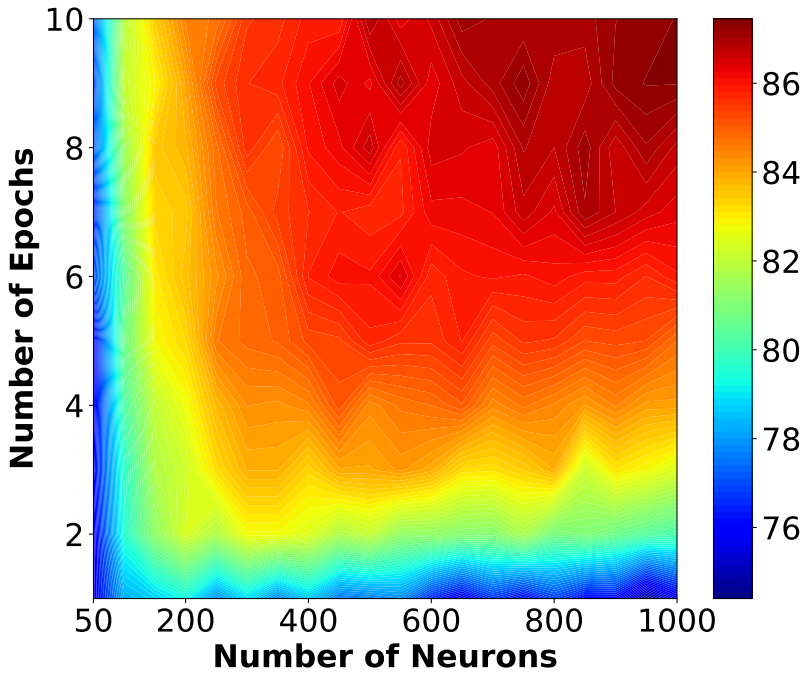


Figure 6.9: Effect of increasing number of neurons and epochs on the accuracy of the spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor .

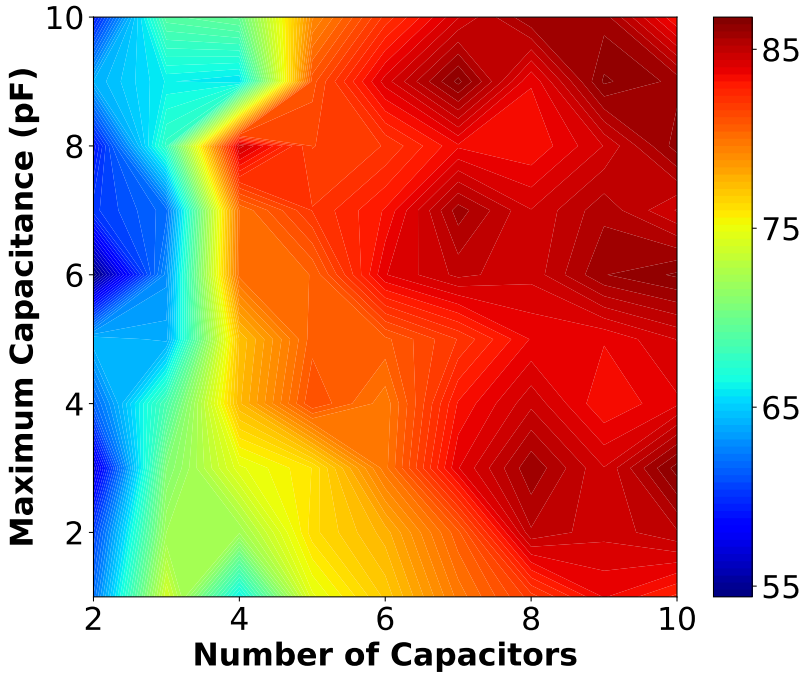


Figure 6.10: Effect of increasing number of capacitors and maximum capacitor on the accuracy of spiking neuromorphic system for on-chip processing of the proposed SPAD based vision sensor.

Table 6.2: Performance Comparison of Proposed Memristive Spiking Neuromorphic Processing with Other Event based Processing

Parameter	**This work**	[66]	[102]	[7]
Data Type	Character	Character	Object	Object
Accuracy	87%	84.9% ± 1.9%	68.16% - 99.83%	50% - 88%
Single-layer SNN	Yes	No	No	No
Integrated with Vision sensor	Yes	No	No	Yes

system with 200 neurons and 10 epochs is on par with that for the same task, the developed memristive spiking neuromorphic system provides a more energy and area efficient processing due to the use of nano-scale memristor device. Furthermore, there are multiple parameters in the proposed design that could be optimized and the attained highest accuracy was found to be 89.54%. Moreover, a separate Dynamic Vision Sensor (DVS) [55] is used to image the digits which are then processed by the developed hierarchical SNN model in [66]. In comparison the proposed work provides a complete system which consists of both the SPAD imager to image the digits and the integrated memristive spiking neuromorphic system to enable on-chip processing.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This work presents a device-aware design of a memristive spiking neuromorphic system, laying out the foundation with circuit level optimizations by defining memristor forming and programming scheme and continuing on to establish a robust unsupervised learning framework using STDP. Here, a programming protocol has been presented that reuses the forming circuit to selectively program memristors along with CMOS memory devices, reducing the pin count by sequentially scanning the data into the system. Energy efficiency of the proposed forming and programming approach was evaluated, suggesting the need for a push towards device level improvement to lower forming voltages, or produce forming-free devices altogether and achieve higher memristance value. The scan-in scheme was further enhanced by accounting for multi-bit memristor programming and reading with two separate scan registers, leveraging the parallelism of memristor access and data scanning. The proposed access method was found to be energy efficient, while being scalable to account for high number of memristance state as well as memristors with minimal increase in circuit overhead. With the formed and programmed memristors, a synapse composed of two complementary connection of memristors was built to realize on-chip STDP based learning. The effect of design parameters on the STDP behavior of the proposed synapse was analyzed, providing the designer choices in selecting the approach to control the synaptic weight change. A sparse spiking recurrent network was built using the designed synapse to validate

its efficacy by performing different data classification tasks. The twin memristor synapse design was improved upon to enable crossbar compatibility to develop an unsupervised learning platform utilizing STDP enabled synapses and homeostasis enabled neurons. A hardware friendly temporal encoding scheme was developed to convert input data into coherent spikes to realize input neurons supporting pulse width modulated fully digital STDP in the twin memristor synapse. A dynamically adjusted parallel capacitor bank was utilized to implement homeostatic plasticity in the output neurons that communicates with WTA bus logic for lateral inhibition, enabling a competitive learning mechanism. The performance of the designed single-layer memristive crossbar based SNN was evaluated by applying it to a pattern recognition task of classifying handwritten digits. The effect of varying the design parameters of the network such as the number of output neurons, training epochs, capacitors was documented, providing the designer with a plethora of design variables to fine-tune the network performance. The designed spiking neuromorphic system with unsupervised learning is proven to be resilient against memristor device non-idealities such as parameter asymmetry, endurance degradation, device failure and fabrication process variability with the aid of circuit level modifications such as clock frequency and duty cycle modulation. The spiking neuromorphic system is also shown to be comparable with other similar systems in terms of area overhead and energy efficiency, while providing added feature of homeostasis in the neuron. As an application of the designed memristive spiking neuromorphic system, a vision sensor based on SPAD with AER readout was interfaced with the neuromorphic system by converting the asynchronous SPAD event into Temporally Coded Spikes. The performance of the proposed vision sensor enabling on-chip neuromorphic processing was analyzed by varying the neuromorphic network parameters as well as circuit level parameters.

7.2 Future Work

Spiking neuromorphic systems offer a biologically plausible energy efficient learning paradigm by encoding data into sparse spiking events. The discovery of memristor, a nanoelectronic device, has accelerated the development of SNNs, as the inherent plasticity of the device makes it a viable candidate to mimic a biological synapse. The memristive spiking

neuromorphic system coupled with SPAD based vision sensing could be a powerful method in image recognition applications, being biologically plausible as well as energy efficient. Following are some directions to advance the research work presented here:

- Testing of the building blocks of the proposed spiking neuromorphic system, such as input neurons, synapses, output neurons etc. and the designed SPAD to TCS decoder.
- Investigating alternative methods of implementing homeostasis in the neurons. The key element to realize the homeostasis mechanism is the plasticity of the neuronal threshold. Memristors themselves provide plasticity by changing states. Hence, in addition to building synapses, memristors could be used to build homeostatic neurons.
- Exploring new applications for the designed single layer memristive crossbar based SNN. The single layer network can work as the final classification layer of a multi-layer SNN, RNN or a readout layer for reservoir computing. To even extend beyond the spiking neuromorphic realm, the designed crossbar based network could be combined with CNN and used as a fully connected layer.
- Expanding the single layer network to be compatible with multi-layer network operation, by developing algorithm to design hidden neurons for feature extraction. This coupled with the spiking nature of operation could pave the way for a highly efficient and accurate learning paradigm.
- Extending the image recognition application of the SPAD based vision sensor with neuromorphic processing to recognize moving images or objects and eventually video data. The energy efficient event based approach with on-chip processing could enable the edge-devices to perform machine learning for applications such as autonomous vehicles, robots, healthcare, space, remote sensing and security.

Bibliography

[1] Adhikari, S. P., Yang, C., Kim, H., and Chua, L. O. (2012). Memristor bridge synapse-based neural network and its learning. *IEEE Transactions on Neural Networks and Learning Systems*, 23(9):1426–1435. 11

[2] Adnan, M. M., Amer, S., and Rose, G. S. (2018a). A novel scan-in scheme for cmos/reram programmable logic circuits. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE. 15

[3] Adnan, M. M., Amer, S., Sayyaparaju, S., Hasan, M. S., and Rose, G. S. (2019). A scan register based access scheme for multilevel non-volatile memristor memory. In *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 630–633. IEEE. 15, 81

[4] Adnan, M. M., Sayyaparaju, S., Brown, S. D., Shawkat, M. S. A., Schuman, C. D., and Rose, G. S. (2021). Design of a robust memristive spiking neuromorphic system with unsupervised learning in hardware. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 17(4):1–26. 55

[5] Adnan, M. M., Sayyaparaju, S., Rose, G. S., Schuman, C. D., Ku, B. W., and Lim, S. K. (2018b). A twin memristor synapse for spike timing dependent learning in neuromorphic systems. In *2018 31st IEEE International System-on-Chip Conference (SOCC)*, pages 37–42. IEEE. 40, 58

[6] Afshar, S., Hamilton, T. J., Davis, L., Van Schaik, A., and Delic, D. (2019). Event-based processing of single photon avalanche diode sensors. *arXiv preprint arXiv:2001.02060*. 98, 99

[7] Afshar, S., Hamilton, T. J., Davis, L., Van Schaik, A., and Delic, D. (2020). Event-based processing of single photon avalanche diode sensors. *IEEE Sensors Journal*, 20(14):7677–7691. 112

[8] Amer, S., Hasan, M. S., Adnan, M. M., and Rose, G. S. (2018). Spice modeling of insulator metal transition: Model of the critical temperature. *IEEE Journal of the Electron Devices Society*, 7:18–25. 13

[9] Amer, S., Hasan, M. S., and Rose, G. S. (2017a). Analysis and modeling of electroforming in transition metal oxide-based memristors and its impact on crossbar array density. *IEEE Electron Device Letters*, 39(1):19–22. 2, 17

[10] Amer, S., Rose, G. S., Beckmann, K., and Cady, N. C. (2017b). Design techniques for in-field memristor forming circuits. In *Proceedings of IEEE International Midwest Symposium on Circuits and Systems MWSCAS*, Boston, Massachusetts. 8, 15

[11] Amer, S., Sayyaparaju, S., Rose, G. S., Beckmann, K., and Cady, N. C. (2017c). A practical hafnium-oxide memristor model suitable for circuit design and simulation. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE. x, 7, 9, 81

[12] Beckmann, K., Holt, J., Manem, H., Van Nostrand, J., and Cady, N. C. (2016). Nanoscale hafnium oxide rram devices exhibit pulse dependent behavior and multi-level resistance capability. *Mrs Advances*, 1(49):3355–3360. 2, 19, 25, 30, 49, 53, 86

[13] Bellisai, S., Bronzi, D., Villa, F. A., Tisa, S., Tosi, A., and Zappa, F. (2013). Single-photon pulsed-light indirect time-of-flight 3d ranging. *Optics express*, 21(4):5086–5098. 98

[14] Berdan, R., Prodromakis, T., and Toumazou, C. (2012). High precision analogue memristor state tuning. *Electronics letters*, 48(18):1105–1107. 10

[15] Berkovich, A., Datta-Chaudhuri, T., and Abshire, P. (2015). A scalable 20× 20 fully asynchronous SPAD-based imaging sensor with AER readout. In *IEEE International Symposium on Circuits and Systems*, pages 1110–1113. 99

[16] Berzina, T., Erokhina, S., Camorani, P., Konovalov, O., Erokhin, V., and Fontana, M. (2009). Electrochemical control of the conductivity in an organic memristor: a time-resolved x-ray fluorescence study of ionic drift as a function of the applied voltage. *ACS Applied materials & interfaces*, 1(10):2115–2118. 6

[17] Bi, G.-q. and Poo, M.-m. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience*, 24(1):139–166. 10, 61

[18] Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E., and Beigne, E. (2019). Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(2):1–35. 1

[19] Brown, S. D., Chakma, G., Adnan, M. M., Hasan, M. S., and Rose, G. S. (2019). Stochasticity in neuromorphic computing: Evaluating randomness for improved performance. In *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 454–457. IEEE. 68

[20] Campbell, K. A., Drake, K. T., and Barney Smith, E. H. (2016). Pulse shape and timing dependence on the spike-timing dependent plasticity response of ion-conducting memristors as synapses. *Frontiers in bioengineering and biotechnology*, 4:97. 53

[21] Cantley, K. D., Subramaniam, A., Stiegler, H. J., Chapman, R. A., and Vogel, E. M. (2012). Neural learning circuits utilizing nano-crystalline silicon transistors and memristors. *IEEE transactions on neural networks and learning systems*, 23(4):565–573. 6

[22] Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66. 96

[23] Chakma, G., Adnan, M. M., Wyer, A. R., Weiss, R., Schuman, C. D., and Rose, G. S. (2017a). Memristive mixed-signal neuromorphic systems: Energy-efficient learning at the circuit-level. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*. 40, 42, 43, 44, 49

[24] Chakma, G., Adnan, M. M., Wyer, A. R., Weiss, R., Schuman, C. D., and Rose, G. S. (2018). Memristive mixed-signal neuromorphic systems: Energy-efficient learning at the circuit-level. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(1):125–136. 2, 10, 13, 32, 58, 68, 81, 94

[25] Chakma, G., Sayyaparaju, S., Weiss, R., and Rose, G. S. (2017b). A mixed-signal approach to memristive neuromorphic system design. In *60th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Boston, MA. 21

[26] Chanthbouala, A., Garcia, V., Cherifi, R. O., Bouzehouane, K., Fusil, S., Moya, X., Xavier, S., Yamada, H., Deranlot, C., Mathur, N. D., et al. (2012). A ferroelectric memristor. *Nature materials*, 11(10):860–864. 6

[27] Chaudhuri, A. and Chakrabarty, K. (2018). Analysis of process variations, defects, and design-induced coupling in memristors. In *2018 IEEE International Test Conference (ITC)*, pages 1–10. IEEE. 90

[28] Chen, S. and Guo, M. (2019). Live demonstration: Celex-v: a 1m pixel multi-mode event-based sensor. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1682–1683. IEEE. 99

[29] Chua, L. (1971). Memristor-the missing circuit element. *IEEE Transactions on circuit theory*, 18(5):507–519. 2, 6

[30] Cong, J. and Xiao, B. (2011). mrfpga: A novel fpga architecture with memristor-based reconfiguration. In *2011 IEEE/ACM international symposium on Nanoscale Architectures*, pages 1–8. IEEE. 10

[31] Covi, E., Brivio, S., Serb, A., Prodromakis, T., Fanciulli, M., and Spiga, S. (2016). Analog memristive synapse in spiking networks implementing unsupervised learning. *Frontiers in neuroscience*, 10:482. 13

[32] Dean, M. E., Schuman, C. D., and Birdwell, J. D. (2014). Dynamic adaptive neural network array. In *International Conference on Unconventional Computation and Natural Computation*, pages 129–141. Springer. 21

[33] Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository. 49

[34] Diehl, P. U. and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99. 2, 13, 84

[35] Dua, D. and Graff, C. (2017). UCI machine learning repository. 73, 104

[36] Guo, M., Huang, J., and Chen, S. (2017). Live demonstration: A 768× 640 pixels 200meps dynamic vision sensor. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–1. IEEE. 99

[37] Guo, Y., Wu, H., Gao, B., and Qian, H. (2019). Unsupervised learning on resistive memory array based spiking neural networks. *Frontiers in neuroscience*, 13:812. 2, 13, 14, 84, 95

[38] Hashem, N. and Das, S. (2012). Switching-time analysis of binary-oxide memristors via a nonlinear model. *Applied Physics Letters*, 100(26):262106. 49

[39] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. 1, 96

[40] Hebb, D. O. (1949). *The organization of behavior: A neuropsychological approach*. John Wiley & Sons. 10

[41] Hu, M., Chen, Y., Yang, J. J., Wang, Y., and Li, H. (2016). A memristor-based dynamic synapse for spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 53

[42] Hu, M., Chen, Y., Yang, J. J., Wang, Y., and Li, H. H. (2017). A compact memristor-based dynamic synapse for spiking neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(8):1353–1366. 94

[43] Hu, P., Li, X., Lu, J., Yang, M., Lv, Q., and Li, S. (2011). Oxygen deficiency effect on resistive switching characteristics of copper oxide thin films. *Physics Letters A*, 375(18):1898–1902. 6

[44] Ielmini, D. (2011). Modeling the universal set/reset characteristics of bipolar rram by field-and temperature-driven filament growth. *IEEE Transactions on Electron Devices*, 58(12):4309–4317. 17

[45] Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73. 13

[46] Jeong, D. S., Kim, K. M., Kim, S., Choi, B. J., and Hwang, C. S. (2016). Memristors for energy-efficient new computing paradigms. *Advanced Electronic Materials*, 2(9):1600090. 1, 2

[47] Jeong, Y., Zidan, M. A., and Lu, W. D. (2017). Parasitic effect analysis in memristor-array-based neuromorphic systems. *IEEE Transactions on Nanotechnology*, 17(1):184–193. 96

[48] Jiang, L., Lv, F.-C., Yang, R., Hu, D.-C., and Guo, X. (2019). Forming-free artificial synapses with ag point contacts at interface. *Journal of Materiomics*, 5(2):296–302. 92

[49] Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters*, 10(4):1297–1301. 6

[50] Kim, H., Sah, M. P., Yang, C., and Chua, L. O. (2010). Memristor-based multilevel memory. In *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, pages 1–6. IEEE. 10

[51] Kumar, S., Wang, Z., Huang, X., Kumari, N., Davila, N., Strachan, J. P., Vine, D., Kilcoyne, A. D., Nishi, Y., and Williams, R. S. (2017). Oxygen migration during resistance switching and failure of hafnium oxide memristors. *Applied Physics Letters*, 110(10):103503. 88

[52] Kuzum, D., Yu, S., and Wong, H. P. (2013). Synaptic electronics: materials, devices and applications. *Nanotechnology*, 24(38):382001. 14, 86, 95

[53] Lecerf, G., Tomas, J., and Saïghi, S. (2013). Excitatory and inhibitory memristive synapses for spiking neural networks. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 1616–1619. IEEE. 11

[54] Li, Y., Zhong, Y., Xu, L., Zhang, J., Xu, X., Sun, H., and Miao, X. (2013). Ultrafast synaptic events in a chalcogenide memristor. *Scientific Reports*, 3:1619. 6

[55] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128 × 128 120 dB 15μ s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576. 113

[56] Linares-Barranco, B., Serrano-Gotarredona, T., Camuñas-Mesa, L. A., Perez-Carrasco, J. A., Zamarreño-Ramos, C., and Masquelier, T. (2011). On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex. *Frontiers in neuroscience*, 5:26. 13

[57] Liu, S.-C. and Minch, B. A. (2002). Silicon synaptic adaptation mechanisms for homeostasis and contrast gain control. *IEEE Transactions on Neural Networks*, 13(6):1497–1503. 13

[58] Masquelier, T., Guyonneau, R., and Thorpe, S. J. (2009). Competitive stdp-based spike pattern learning. *Neural computation*, 21(5):1259–1276. 75

[59] Medeiros-Ribeiro, G., Perner, F., Carter, R., Abdalla, H., Pickett, M. D., and Williams, R. S. (2011). Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution. *Nanotechnology*, 22(9):095702. 6, 53

[60] Mehonic, A., Cueff, S., Wojdak, M., Hudziak, S., Jambois, O., Labbé, C., Garrido, B., Rizk, R., and Kenyon, A. J. (2012). Resistive switching in silicon suboxide films. *Journal of Applied Physics*, 111(7):074507. 6

[61] Merced-Grafals, E. J., Dávila, N., Ge, N., Williams, R. S., and Strachan, J. P. (2016). Repeatable, accurate, and high speed multi-level programming of memristor 1t1r arrays for power efficient analog computing applications. *Nanotechnology*, 27(36):365202. 8

[62] Mostafa, H. and Ismail, Y. (2016). Statistical yield improvement under process variations of multi-valued memristor-based memories. *Microelectronics Journal*, 51:46–57. 2, 90

[63] Ngoc Truong, S. (2019). A parasitic resistance-adapted programming scheme for memristor crossbar-based neuromorphic computing systems. *Materials*, 12(24):4097. 95

[64] Noack, M., Partzsch, J., Mayr, C., Hänzsche, S., Scholze, S., Höppner, S., Ellguth, G., and Schüffny, R. (2014). Switched-capacitor realization of presynaptic short-term-plasticity and stop-learning synapses in 28 nm cmos. *arXiv preprint arXiv:1412.3243*. 53

[65] Oblea, A. S., Timilsina, A., Moore, D., and Campbell, K. A. (2010). Silver chalcogenide based memristor devices. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–3. IEEE. 6

[66] Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., and Benosman, R. (2015). Hfirst: a temporal approach to object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(10):2028–2040. 110, 112, 113

[67] Panchula, A. (2007). Oscillating-field assisted spin torque switching of a magnetic tunnel junction memory element. US Patent 7,224,601. 6

[68] Pedretti, G., Milo, V., Ambrogio, S., Carboni, R., Bianchi, S., Calderoni, A., Ramaswamy, N., Spinelli, A., and Ielmini, D. (2017). Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Scientific reports*, 7(1):5288. 1, 11

[69] Posch, C., Matolin, D., and Wohlgenannt, R. (2010). A QVGA 143db dynamic range asynchronous address-event pwm dynamic image sensor with lossless pixel-level video compression. *IEEE International Solid-State Circuits Conference*, pages 400–401. 99

[70] Pouyan, P., Amat, E., and Rubio, A. (2014). Reliability challenges in design of memristive memories. In *2014 5th European Workshop on CMOS Variability (VARI)*, pages 1–6. IEEE. 2, 90

[71] Querlioz, D., Bichler, O., Dollfus, P., and Gamrat, C. (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions on Nanotechnology*, 12(3):288–295. 2, 13, 14, 84, 95

[72] Rajendran, J., Maenm, H., Karri, R., and Rose, G. S. (2011). An approach to tolerate process related variations in memristor-based applications. In *2011 24th Internatioal Conference on VLSI Design*, pages 18–23. IEEE. 90

[73] Rovere, G., Ning, Q., Bartolozzi, C., and Indiveri, G. (2014). Ultra low leakage synaptic scaling circuits for implementing homeostatic plasticity in neuromorphic architectures. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2073–2076. IEEE. 13

[74] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science. 1

[75] Sayyaparaju, S., Adnan, M. M., Amer, S., and Rose, G. S. (2020). Device-aware circuit design for robust memristive neuromorphic systems with stdp-based learning. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 16(3):1–25. 2, 14, 94, 95, 96

[76] Sayyaparaju, S., Weiss, R., and Rose, G. S. (2018). A mixed-mode neuron with on-chip tunability for generic use in memristive neuromorphic systems. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 441–446. IEEE. 13

[77] Schuman, C. D., Plank, J. S., Disney, A., and Reynolds, J. (2016). An evolutionary optimization framework for neural networks and neuromorphic architectures. In *International Joint Conference on Neural Networks*, Vancouver. IEEE. 21

[78] Schuman, C. D., Plank, J. S., Rose, G. S., Chakma, G., Wyer, A., Bruer, G., and Laanait, N. (2017). A programming framework for neuromorphic systems with emerging technologies. In *Proceedings of the 4th ACM International Conference on Nanoscale Computing and Communication*, page 15. ACM. 51

[79] Shawkat, M. S. A., Habib, M. H. U., and McFarlane, N. (2018). An analog CMOS silicon photomultiplier using perimeter-gated single-photon avalanche diodes. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65:3830–3841. 98

[80] Shawkat, M. S. A. and Mcfarlane, N. (2020). A digital cmos silicon photomultiplier using perimeter gated single photon avalanche diodes with asynchronous aer readout. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(12):4818–4828. 99

[81] Shawkat, M. S. A., Sayyarparaju, S., McFarlane, N., and Rose, G. S. (2020). Single photon avalanche diode based vision sensor with on-chip memristive spiking neuromorphic processing. In *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 377–380. IEEE. xvi, 99, 100

[82] Sheri, A. M., Hwang, H., Jeon, M., and Lee, B.-g. (2014). Neuromorphic character recognition system with two pcmo memristors as a synapse. *IEEE Transactions on Industrial Electronics*, 61(6):2933–2941. 2, 14, 84

[83] Shi, X., Zeng, Z., Yang, L., and Huang, Y. (2018). Memristor-based circuit design for neuron with homeostatic plasticity. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(5):359–370. 13

[84] Snider, G. S. (2008). Spike-timing-dependent learning in memristive nanodevices. In *Proceedings of the 2008 IEEE International Symposium on Nanoscale Architectures*, pages 85–92. IEEE Computer Society. 2, 21

[85] Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926. 1, 11

[86] Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *nature*, 453(7191):80–83. 2, 6

[87] Uddin, M., Majumder, M. B., Beckmann, K., Manem, H., Alamgir, Z., Cady, N. C., and Rose, G. S. (2017a). Design considerations for memristive crossbar physical unclonable functions. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 14(1):1–23. 88

[88] Uddin, M., Majumder, M. B., and Rose, G. S. (2017b). Robustness analysis of a memristive crossbar puf against modeling attacks. *IEEE Transactions on Nanotechnology*, 16(3):396–405. 23, 86, 90

[89] Uddin, M., Majumder, M. B., Rose, G. S., Beckmann, K., Manem, H., Alamgir, Z., and Cady, N. C. (2016). Techniques for improved reliability in memristive crossbar puf circuits. In *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, pages 212–217. IEEE. 6

[90] Villa, F., Lussana, R., Bronzi, D., Tisa, S., Tosi, A., Zappa, F., Dalla Mora, A., Contini, D., Durini, D., Weyers, S., et al. (2014). CMOS imager with 1024 SPADs and TDCs for single-photon timing and 3-D time-of-flight. *IEEE journal of selected topics in quantum electronics*, 20(6):364–373. 98

[91] Wang, X., Li, S., Liu, H., and Zeng, Z. (2017). A compact scheme of reading and writing for memristor-based multivalued memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(7):1505–1509. 38

[92] Wang, Z., Joshi, S., Savel'ev, S., Song, W., Midya, R., Li, Y., Rao, M., Yan, P., Asapu, S., Zhuo, Y., et al. (2018). Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics*, 1(2):137–145. 14

[93] Wang, Z., Zhao, W., Kang, W., Zhang, Y., Klein, J.-O., and Chappert, C. (2014). Ferroelectric tunnel memristor-based neuromorphic network with 1t1r crossbar architecture. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 29–34. IEEE. 2, 11

[94] Wu, H., Yao, P., Gao, B., Wu, W., Zhang, Q., Zhang, W., Deng, N., Wu, D., Wong, H.-S. P., Yu, S., et al. (2017). Device and circuit optimization of rram for neuromorphic computing. In *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 11–5. IEEE. 88

[95] Wu, X., Saxena, V., and Zhu, K. (2015a). Homogeneous spiking neuromorphic system for real-world pattern recognition. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):254–266. 2, 13, 14, 75, 95

[96] Wu, X., Saxena, V., Zhu, K., and Balagopal, S. (2015b). A cmos spiking neuron for brain-inspired neural networks with resistive synapses and in situ learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(11):1088–1092. 94

[97] Wyer, A., Adnan, M. M., Ku, B. W., Lim, S. K., Schuman, C. D., Pooser, R. C., and Rose, G. S. (2017). Evaluating online learning in memristive neuromorphic circuits. In *Neuromorphic Computing Symposium: Architectures, Models, and Applications (NCAMA)*, Knoxville, TN. 23

[98] Yang, J. J., Zhang, M.-X., Strachan, J. P., Miao, F., Pickett, M. D., Kelley, R. D., Medeiros-Ribeiro, G., and Williams, R. S. (2010). High switching endurance in tao x memristive devices. *Applied Physics Letters*, 97(23):232102. 6, 88

[99] Yilmaz, Y. and Mazumder, P. (2017). A drift-tolerant read/write scheme for multilevel memristor memory. *IEEE Transactions on Nanotechnology*, 16(6):1016–1027. 10

[100] Zappa, F., Tisa, S., Tosi, A., and Cova, S. (2007). Principles and features of single-photon avalanche diode arrays. *Sensors and Actuators A: Physical*, 140(1):103–112. 98

[101] Zhou, E., Fang, L., and Yang, B. (2018). Memristive spiking neural networks trained with unsupervised stdp. *Electronics*, 7(12):396. 84

[102] Zhou, S. and Wang, W. (2018). Object detection based on lidar temporal pulses using spiking neural networks. *arXiv preprint arXiv:1810.12436*. 112

Appendices

A Verilog Codes

A.1 Input Neuron

A verilog code for the proposed input neuron of section 5.1.2 is attached here:

```
module topINP(
    input CLK,
    input RST,
    input TCS,
    input POL,
    output VoMp,
    output VoMn
    );

    wire RSTb,clk2,clk4,clk8,clk8b,sq,sqb,sclk,sqbclk8,qreg4,voutp, sclkb,
    snq, rst, Vp, Vn, xn, xn1, OEMp, OEMn;
    assign v1 = 1'b1;
    assign v0 = 1'b0;
    assign RSTb = ~RST;

    clockdiv2 div2(CLK,RSTb,clk2);
    clockdiv2 div4(clk2,RSTb,clk4);
    clockdiv2 div8(clk4,RSTb,clk8);

    //For positive intensity pixels
    //Creating the signal for accumulation
    assign sclk = TCS & clk8;
    dffNegEdge dffneg2(v1,sclk,RST,sq);
    assign sqb = ~sq;
    assign sqbclk8 = sqb & clk8;

    //Replicating the signal for potentiation
    reg4 reg1(sqbclk8,CLK,clk8,RST,qreg4);
    assign voutp = sqbclk8 | qreg4;

    //For negative intensity pixels
    assign sclkb = TCS & (~clk8);
    assign rst2 = RST | clk8;
    dff dff1(v1,sclkb,rst2,snq);
    assign clk8b = ~clk8;

    assign Vp = voutp & POL;
    assign Vn = (~snq) & (~POL) & clk8b;

```

```
40      //Digital signal generation for positive and negative inputs
41      assign Vom = (clk8)?Vp:Vn;
42
43      //generating output enables
44      assign xn1 = clk8 & Vp;
45      assign xn = (POL)?xn1:clk8b;
46
47      assign OEMp = xn & RSTb;
48      assign OEMn = POL & sqbclk8;
49
50
51      assign VoMp = (OEMp) ? Vom : 1'bz;
52      assign VoMn = (OEMn) ? v0 : 1'bz;
53
54  endmodule
55
56
57
58  module clockdiv2(inclk,en,outclk);
59      input inclk,en;
60      output reg outclk;
61
62  always_ff@(posedge inclk or negedge en)
63  begin
64      if (en==0)
65          outclk <= 0;
66      else
67          outclk <= ~outclk;
68  end
69
70
71
72  module dff(d,clk,clr,q);
73      input d,clk,clr;
74      output reg q;
75
76      always_ff@(posedge clk, posedge clr)
77      begin
78          if (clr)
79              q <= 0;
80          else
81              q <= d;
82      end
83
84  endmodule
```

```
module dffNegEdge(d,clk,clr,q);

    input d,clk,clr;
    output reg q;

    always@(negedge clk, posedge clr)
    begin
        if (clr)
            q <= 0;
        else
            q <= d;
    end

endmodule



module reg4(d,clk,en,clr,qbit1);
    parameter n=4;
    input d,clk,clr,en;
    output qbit1;
    integer k;

    reg [n-1:0]q;

    assign qbit1 = q[0];

    always@(posedge clk, posedge clr)
    begin
        if (clr)
          q <= {n{1'b0}};
        else
        begin
            if (en==1)
            begin
                q[n-1] <= d;
                for (k=n-1;k>0;k=k-1)
                    q[k-1] <= q[k];
            end
            else
                q <= q;
        end
    end
endmodule
```

A.2 Output Neuron Control Block

A verilog code for the control block of the proposed output neuron of section 5.1.3 is given here:

```
module outNeuCB(Spike_async, WTA, NXT, CLK, RST, Pot, Pot_b, Dep, Dep_b,
    DC, DC_b, Accum, Accum_b, Sw0, Sw0_b, Sw1, Sw1_b, RST_b, Spike);
    input Spike_async;//Asynchrous spike
    input WTA;//WTA bus signal If WTA = 1, other neuron has fired.
    input NXT;//next input
    input CLK, RST;

    output Pot, Dep, DC, Accum;
    output Sw0, Sw1;

    output Pot_b, Dep_b, DC_b, Accum_b;
    output Sw0_b, Sw1_b;

    output RST_b;
    output Spike;

    reg [3:0]cnt;
    reg [1:0]switches;
    wire WIN,los, spk_sync, anyN, qDC, CLK2, CLK4, CLK8, CLK8b, P,
    rstAny, rstAnyb;

    //Reset for either global reset or Next input
    assign rstAny = RST | NXT;
    assign rstAnyb = !rstAny;

// Active low reset
    assign RST_b = !RST;

// Clock division
    clockdiv2 div2(CLK,rstAnyb,CLK2);
    clockdiv2 div4(CLK2,rstAnyb,CLK4);
    clockdiv2 div8(CLK4,rstAnyb,CLK8);
    assign CLK8b = !CLK8;

// Neuron phase generation
    dff dffSpk(1'b1,Spike_async,rstAny,Spike);
    assign WIN = Spike & !WTA; //Win when no neuron fired but this one did
    dff dff1(WIN,CLK8b,rstAny,spk_sync);
    assign Pot = CLK8 & spk_sync;
    assign Dep = CLK8b & spk_sync;
```

```
41
42      assign anyN = Spike | WTA;
43      dff dff2(!anyN,CLK8b,rstAny,qDC);
44      assign DC = qDC | anyN | rstAny;
45      assign Accum = (!DC) & rstAnyb & CLK8;
46
47  // Homeostasis and Reverse Homeostasis
48      assign los = !Spike & WTA;
49      Counter #(4) cntrREV (.CLK(los), .RSTn(!WIN & RST_b), .EN(1'b1),
50      .COUNT(cnt));
51      assign P = &cnt;
52      // If P = 1, Reverse Homeostasis (Down count)
53      // If P = 0, Homeostasis (Up count)
54      downUpbCounter #(2) cntrSW (.CLK(Pot|(P & NXT)), .RSTn(RST_b), .EN(1'b1),
55      .Down(P),.COUNT(switches));
56      //assign Sw0 = Accum & switches[0];
57      //assign Sw1 = Accum & switches[1];
58      assign Sw0 = switches[0];
59      assign Sw1 = switches[1];
60
61      assign Pot_b = !Pot;
62      assign Dep_b = !Dep;
63      assign Accum_b = !Accum;
64      assign DC_b = !DC;
65      assign Sw0_b = !Sw0;
66      assign Sw1_b = !Sw1;
67
68  endmodule
69
70  module clockdiv2(inclk,en,outclk);
71      input inclk,en;
72      output reg outclk;
73
74  always_ff@(posedge inclk or negedge en)
75  begin
76      if (en==0)
77          outclk <= 0;
78      else
79          outclk <= ~outclk;
80  end
81
82  endmodule
83
84  module dff(d,clk,clr,q);
85      input d,clk,clr;
```

```
86      output reg q;
87
88      always_ff@(posedge clk, posedge clr)
89      begin
90          if (clr)
91              q <= 0;
92          else
93              q <= d;
94      end
95
96  endmodule
97
98
99
100 module Counter  #(parameter n = 2) (
101     input CLK,RSTn,EN,
102     output reg [n-1:0]COUNT
103     );
104
105     always_ff@(posedge CLK or negedge RSTn)
106     if (!RSTn)
107         COUNT = 0;
108     else
109         if (EN)
110             COUNT = COUNT + 1;
111
112 endmodule
113
114
115
116 module downUpbCounter  #(parameter n = 2) (
117     input CLK,RSTn,EN,Down,
118     output reg [n-1:0]COUNT
119     );
120
121     always_ff@(posedge CLK or negedge RSTn)
122     begin
123         if (!RSTn)
124             COUNT = 0;
125         else
126         begin
127             if (EN)
128             begin
129                 if (Down && COUNT > 0)
130                     COUNT = COUNT - 1;
```

```
                else if (!Down && COUNT < (2**n - 1))
                    COUNT = COUNT + 1;
                else
                    COUNT = COUNT;
            end
        end
    end

endmodule
```

A.3 SPAD to TCS Decoder

A verilog code for the SPAD to TCS decoder described in section 6.2.2 is attached here:

```
module spadDecoder(
    input spadPulse,
    input CLK,
    input RSTn,
    output POL,
    output TCS
    );

    reg [3:0]COUNT;
    reg SPAD;
    wire noSpad;
    wire spadSync, SPAD_b;
    logic [2:0]delays;

    assign SPAD_b = !SPAD;

    //3 bit Counter to detect where spadPulse is ocurring
    always_ff@(posedge CLK or negedge RSTn)
    begin
        if (!RSTn)
            COUNT <= 0;
        else
        if (SPAD_b)
            if (COUNT >= 8)
                COUNT <= COUNT;
            else
                COUNT <= COUNT + 1;
    end

    always_ff@(posedge spadPulse, negedge RSTn)
```

```
31      begin
32          if (!RSTn)
33              SPAD <= 0;
34          else
35              SPAD <= 1;
36      end
37
38      always_comb
39      begin
40          case(COUNT)
41              0     :   delays = 0;
42              1     :   delays = 7;
43              2     :   delays = 5;
44              3     :   delays = 3;
45              4     :   delays = 1;
46              5     :   delays = 7;
47              6     :   delays = 5;
48              7     :   delays = 3;
49              8     :   delays = 1;
50              default : delays = 0;
51          endcase
52      end
53
54      assign POL = ((COUNT <= 4) ? 1 : 0) & RSTn;
55      assign noSpad = (COUNT == 8);
56
57      LtoPmoore SPADsynchronization (.Level(SPAD | noSpad),.CLK(CLK),
58      .RESET(!RSTn), .Pulse(spadSync));
59
60      delayBlockN genDelays (.CLK(CLK), .RSTn(RSTn),.sig(spadSync),
61      .delayN(delays),.delayedSig(TCS));
62
63  endmodule
64
65
66  module LtoPmoore(
67      input Level,
68      input CLK,
69      input RESET,
70      output logic Pulse
71  );
72
73  enum { STATE0, STATE1, STATE2} curr_state, next_state;
74
75  //Next state logic
```

```
always_comb
begin
    case(curr_state)
        STATE0: next_state = Level ? STATE1 : STATE0;
        STATE1: next_state = Level ? STATE2 : STATE0;
        STATE2: next_state = Level ? STATE2 : STATE0;
        default: next_state = STATE0;
    endcase
end

//Flip Flop
always_ff @ (posedge CLK or posedge RESET)
begin
    if (RESET)
        curr_state <= STATE0;
    else
        curr_state <= next_state;
end

//output logic
always_comb
begin
    case(curr_state)
        STATE0: Pulse = 1'b0;
        STATE1: Pulse = 1'b1;
        STATE2: Pulse = 1'b0;
        default: Pulse = 1'b0;
    endcase
end

endmodule


module delayBlockN (CLK, RSTn,sig,delayN,delayedSig);
    parameter N = 3;
    parameter NBIT = 2**N - 1;
    input CLK,RSTn,sig;
    input [N-1:0]delayN;
    output delayedSig;

    //For 3 bit bin2therm, 7 bit thermometer signal--- 110 ---> 011_1111
    wire [NBIT-1:0]delayEnable;
    wire [NBIT:0]delSigs;

    assign delSigs[NBIT] = sig;
```

```
121     assign delayedSig = delSigs[0];
122 
123     bin2therm #(.N(3)) bin2thermEnc (.binin(delayN),.thermout(delayEnable));
124 
125     genvar i;
126     generate
127         for (i=NBIT-1; i>=0; i--)
128         begin: delayGeneration
129             delayUnit delayLine (.CLK(CLK),.RSTn(RSTn),.sig(delSigs[i+1]),
130             .delayEN(delayEnable[i]),.delayedSig(delSigs[i]));
131         end
132     endgenerate
133 
134 endmodule
135 
136 module bin2therm (binin, thermout );
137     parameter N = 3;
138     parameter NTHERM = (1<<N)-1;
139     input  [N-1:0]   binin;
140     output [NTHERM-1:0] thermout;
141 
142     generate
143     genvar i;
144     for(i=0; i<=NTHERM-1; i=i+1)
145     begin
146         assign thermout[i] = (binin > i) ? 1 : 0;
147     end
148     endgenerate
149 
150 endmodule
151 
152 module delayUnit(
153     input CLK,
154     input RSTn,
155     input sig,
156     input delayEN,
157     output delayedSig
158 );
159 
160     reg Q;
161 
162     always_ff@(posedge CLK, negedge RSTn)
163     begin
164         if(!RSTn)
165             Q <= 0;
```

```
166         else
167             Q <= sig;
168     end
169 
170     assign delayedSig = delayEN ? Q : sig;
171 
172 endmodule
```

B Testing Results

The proposed design of the input neuron explained in section 5.1.2 was fabricated using the TSMC 65nm process with a slight modification to the design. Due to limitation of pins on the test chip, it was not possible to dedicate pins for the circuit. Instead, a scan chain strategy was implemented to shift-in inputs and shift-out outputs. A scan enable pin was used for scanning data in and out of the circuit. As there were multiple other test circuits, the input neuron was assigned an address, and after scanning in the address bits to select the input neuron, the data bits for that particular circuit was scanned in. As soon as the scanning-in of the data bits is complete, the circuit was enabled to generate results, which are stored in another scan register. After the circuit operation finishes, scan enable was asserted high again to shift out the results. The test setup with a logic analyzer is shown in Fig. B.1.

The results obtained from the testing of the input neuron is shown in Figs. B.2 and B.3 for both positive and negative inputs, respectively. The TCS spike generated from the input and the polarity is supplied as the input signal to the circuit via the *scan_in* pin. The output is read when the *scan_enable* signal goes high for the second time. When the TCS represent higher magnitude pixel value, the pulse width is longer and vice versa. When the TCS represents positive input, the pulses are pulled to the positive rail. When the TCS represents negative input, the pulses are pulled to the negative rail. Here, the results are shown for pixel values of ± 1 and ± 3, showing that the pulses are one cycle and three cycle wide respectively.

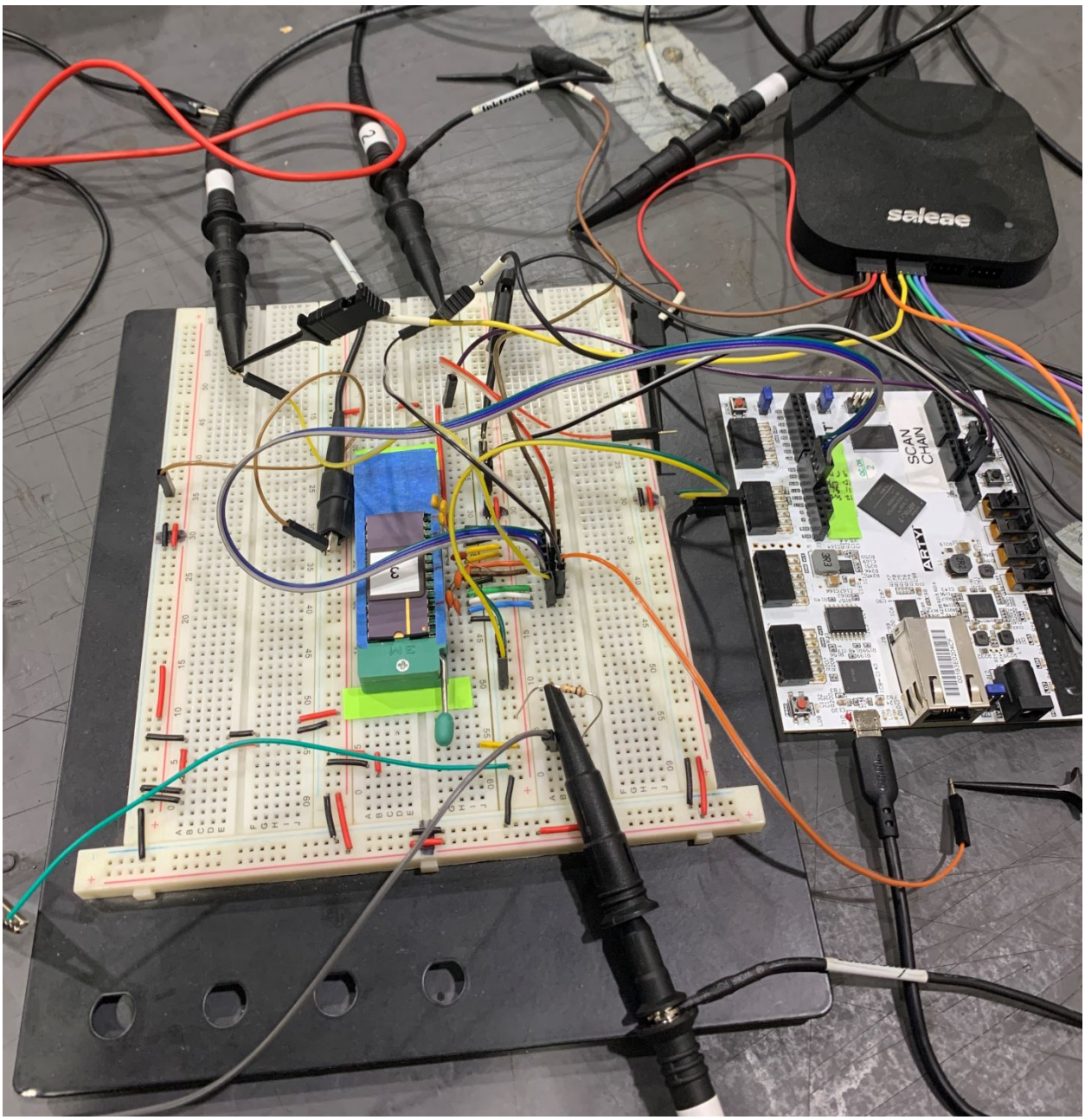
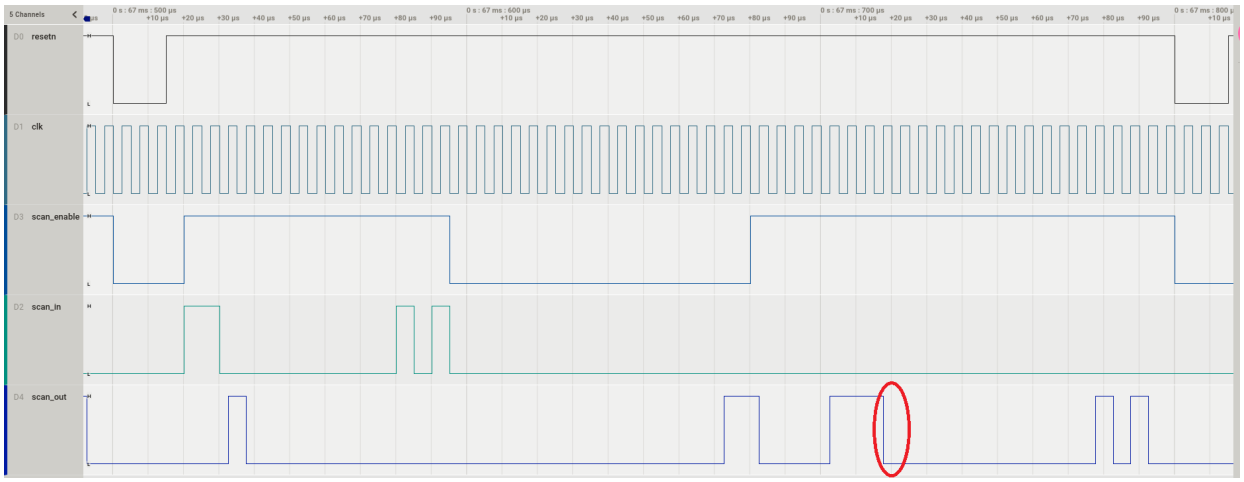


Figure B.1: Test setup for testing the input neuron with a logic analyzer.

(a)

(b)

Figure B.2: Input neuron testing for TCS representing positive inputs. *resetn* is the global reset signal, *clk* is the clock signal, *scan_enable* enable the shifting of data, *scan_in* is the data to be scanned in, *scan_out* is the data that is shifted out. (a) Waveforms for TCS representing the input '+1' that produces a one cycle wide positive pulse during accumulation and potentiation. (b) Waveforms for TCS representing the input '+3' that produces a three cycle wide positive pulse during accumulation and potentiation.



(a)

(b)

Figure B.3: Input neuron testing for TCS representing negative inputs. *resetn* is the global reset signal, *clk* is the clock signal, *scan_enable* enable the shifting of data, *scan_in* is the data to be scanned in, *scan_out* is the data that is shifted out. (a) Waveforms for TCS representing the input '-1' that produces a one cycle wide negative pulse during depression. (b) Waveforms for TCS representing the input '-3' that produces a three cycle wide negative pulse during depression.

Vita

Md Musabbir Adnan received his B.Sc. degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2015. He joined the University of Tennessee, Knoxville in Fall 2016 to pursue his PhD in Computer Engineering. His research interest includes emerging nano-devices, digital logic design, neuromorphic computing, and VLSI circuit designs. He worked as a research intern at Facebook Inc. in 2020, working on prototyping the next generation AR sensor. During his graduate studies, he was a member of UTK Neuromorphic group (TENNLab) and SENECA research group, designing nano-electronic circuits for neuromorphic computing.