Spiking Neural Network with Memristive Based Computing-In-Memory Circuits and Architecture

Fabiha Nowshin

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Electrical Engineering

Yang (Cindy) Yi, Chair
Lingjia Liu
Daniel Sable

May 12, 2021
Blacksburg, VA

Keywords: Neuromorphic Computing, Spiking Neural Network, Reservoir Computing, Pattern Recognition, Digit Recognition, Memristor, Computing-In-Memory, LIF Neuron

Spiking Neural Network with Memristive Based Computing-In-Memory Circuits and Architecture

Fabiha Nowshin

ABSTRACT

In recent years neuromorphic computing systems have achieved a lot of success due to its ability to process data much faster and using much less power compared to traditional Von Neumann computing architectures. There are two main types of Artificial Neural Networks (ANNs), Feedforward Neural Network (FNN) and Recurrent Neural Network (RNN). In this thesis we first study the types of RNNs and then move on to Spiking Neural Networks (SNNs). SNNs are an improved version of ANNs that mimic biological neurons closely through the emission of spikes. This shows significant advantages in terms of power and energy when carrying out data intensive applications by allowing spatio-temporal information processing.

On the other hand, emerging non-volatile memory (eNVM) technology is key to emulate neurons and synapses for in-memory computations for neuromorphic hardware. A particular eNVM technology, memristors, have received wide attention due to their scalability, compatibility with CMOS technology and low power consumption properties. In this work we develop a spiking neural network by incorporating an inter-spike interval encoding scheme to convert the incoming input signal to spikes and use a memristive crossbar to carry out in-memory computing operations. We develop a novel input and output processing engine for our network and demonstrate the spatio-temporal information processing capability. We demonstrate an accuracy of a 100% with our design through a small-scale hardware simulation for digit recognition and demonstrate an accuracy of 87% in software through MNIST simulations.

Spiking Neural Network with Memristive Based Computing-In-Memory Circuits and Architecture

Fabiha Nowshin

GENERAL AUDIENCE ABSTRACT

In recent years neuromorphic computing systems have achieved a lot of success due to its ability to process data much faster and using much less power compared to traditional Von Neumann computing architectures. Artificial Neural Networks (ANNs) are models that mimic biological neurons where artificial neurons or neurodes are connected together via synapses, similar to the nervous system in the human body. here are two main types of Artificial Neural Networks (ANNs), Feedforward Neural Network (FNN) and Recurrent Neural Network (RNN). In this thesis we first study the types of RNNs and then move on to Spiking Neural Networks (SNNs). SNNs are an improved version of ANNs that mimic biological neurons closely through the emission of spikes. This shows significant advantages in terms of power and energy when carrying out data intensive applications by allowing spatio-temporal information processing capability.

On the other hand, emerging non-volatile memory (eNVM) technology is key to emulate neurons and synapses for in-memory computations for neuromorphic hardware. A particular eNVM technology, memristors, have received wide attention due to their scalability, compatibility with CMOS technology and low power consumption properties. In this work we develop a spiking neural network by incorporating an inter-spike interval encoding scheme to convert the incoming input signal to spikes and use a memristive crossbar to carry out in-memory computing operations. We demonstrate the accuracy of our design through a small-scale hardware simulation for digit recognition and demonstrate an accuracy of 87% in software through MNIST simulations.

Acknowledgments

I would like to thank my advisor, Dr. Yang (Cindy) Yi, for her unwavering support, kindness, encouragement, and constant motivations. In the past two years Dr. Yi has always made research fun and ensured we were all taking care of ourselves during this pandemic. She was always ready to help at all times and this would not have been possible without her guidance. I would also like to thank Dr. Lingjia Liu, Dr. Daniel Sable and Dr. Dong Dong for being great committee members.

I would like to thank Dr. Kangjun Bai for helping me learn so much about Analog IC and Neuromorphic Computing and Nima Mohammadi for helping me fill in my gaps with the software part. I would also like to thank Dr. Hongyu An for helping me learn about memristors. Thank you to my colleagues at the MICS lab, Jinhua, Ryan, Moqi, Jiayuan, Chenyuan, Yibin, Victor, Shiya, Honghao, Varsha and Osaze.

I am forever grateful to my parents Rashid Ahamed Hossaini and Umma Kulsum, my brother Ashraf Ansary and my sister-in-law Rufiaat Rashid for always supporting me and being there for me. I am thankful to all my friends from Bangladesh and the friends I have made throughout my years at Virginia Tech.

Table of Contents

Table of Figures

List of Tables

Chapter 1: Introduction

Compared to traditional Von Neumann computing architectures, neuromorphic computing systems have the ability to process data much faster and consume much less power. Neuromorphic computing, introduced by Dr. Carver Mead in 1980, can emulate biological processes using highly parallelized computing architectures [1]. Artificial Neural Networks (ANNs) are models that mimic biological neurons. In ANNs, neurodes or artificial neurons are connected to other neurodes via synapses, similar to the nervous system, and they are usually arranged in a layer [2]. The inputs applied to these neurodes are multiplied by certain weights and it is through the adjustment of these weights that learning is mimicked in ANNs. A transfer function is used to transform the neurons input value where the most commonly used ones include sigmoid or hyperbolic tangent functions [2]. The output values are obtained from the summation of the products of the input values and the corresponding weights.

Feedforward neural networks (FNN) and Recurrent Neural Networks (RNN) are the two most common ANNs. In FNNs, there are no loops found internally and the connections between the neurons happen only in one direction from the input to the output layer [3]. Due to their feedforward nature, only static spatial input patterns can be processed, as the data received from the output layer has information about one particular moment of the input.

RNNs, derived from FNNs, have internal loops or recurrent connections in the hidden layer that allows them to keep the data in the network for a certain time period [4]. This creates dynamical behavior in RNNs and gives it the ability to process both spatial and temporal data. Therefore, RNNs can emulate biological neurons more closely.

1.1 Reservoir Computing

The major drawback of RNNs is that the training procedure is extremely complex and expensive due to the recurrent connections in the hidden layers. Derived from RNNs is Reservoir Computing that possesses the spatio-temporal information processing capability. Echo state networks (ESNs) and Liquid State Machines (LSMs) are subsets of

reservoir computing and they take advantage of the dynamical property of RNNs [5], [6]. ESN and LSM are a subset of reservoir computing and they are a simplified version of the RNNs. ESN and LSM are constructed with a layer of nonlinear neurons to add nonlinearity to the system and have fixed, untrained weights in the reservoir layer. The weights in the output layer are only trained in reservoir computing, which greatly simplifies the training procedures. ESN and LSM has proven to be very successful in the past years. For instance, ESNs have been used in intelligent stock trading systems and automatic speech recognition [7], [8]. LSMs have also shown successful results in speech recognition and pattern recognition on FPGAs [9], [10], [11].

An evolution of ESN and LSM is the Delay-Feedback Reservoir (DFR) Computing which uses the embedded delay property [3], [12], [13]. This is derived from the theory that delay is present in the biological system [14]. In DFR, the entire recurrent network is substituted by one single nonlinear node followed by neurons that are separated by delays. Since delay is present in biological systems, DFR systems are able to mimic biological systems closely. DFR systems have shown successful results in photonic implementations due to their simplicity in hardware implementations [15] [16], [17]. They have also shown to exhibit near chaotic behavior [18]. A complete analog IC implementation of DFR has also been designed and fabricated, as well as been analyzed and evaluated using Monte Carlo simulations [19]. Memristor synapses have also been used with the Deep DFR to show successful results with the applications on spoken digit recognition and handwritten digit classification [20].

In this part of the thesis for background study we first cover the basic architecture and elaborate on the different types of reservoir computing including ESNs, LSMs and Delayed Feedback systems. Electronic Reservoir Computing is explained in the next section with a focus on Analog Implementations, Field-Programmable-Gate-Array (FPGA), Very-Large-Scale-Integrated Circuits (VLSI) and memristors. Other types of reservoir Computing are explained afterwards followed by applications for Reservoir Computing with a conclusion.

1.2 Spiking Neural Network

Deep neural networks that use deep learning has shown excellent results in the field of large data analysis [21]. Deep neural network can be applied to speech recognition, digit recognition, pattern classification and natural language processing. The efficiency of deep neural networks can be improved significantly by using Spiking Neural Networks (SNNs). SNNs mimic biological neurons more closely because of their spiking nature; once a threshold is exceeded, a spike is fired, and the information depends on the specific time of the spikes or the sequence of the spikes [22]. Furthermore, the binary nature of the spikes in SNNs contribute to efficiency in terms of both power and energy. Spiking neural networks are therefore said to be the third generation of artificial neural networks [23].

A number of encoding schemes have been developed to convert the data into spike events. The two main encoding schemes are rate encoding and temporal encoding [24]. Rate encoding is where the data is encoded into the frequency of the spikes and temporal encoding is where the data is encoded into the timing of spikes. In temporal encoding scheme some common ones are time-to-first-spike (TTFS), phase and burst encoding. Temporal encoding is more energy efficient compared to rate encoding because of the fewer spike events generated. In this thesis we use an inter-spike interval (ISI) encoding scheme to encode the data into spike events. The key idea of this ISI encoding scheme is that it has higher information density compared to its TTFS counterpart since it carries information in both the timing of the spikes as well as the distance between the spikes based on the intensity of the information.

Conventional CMOS technology has been employed to build neuromorphic systems which is the use of hardware components to build neural networks. Some notable examples of neuromorphic computing chips include Loihi by Intel and TrueNorth by IBM [25], [26]. Loihi, that consumes only 0.001% of the power consumed by traditional computing architectures while TrueNorth has the capability of classifying multiple objects while consuming only 65mW of power [26].

Compared to the traditional CMOS memory technologies that include static random access memory (SRAM), dynamic random access memory (DRAM) and flash, emerging

non-volatile memory technologies (eNVMs) has shown to emulate biological neurons and synapse more closely that is invaluable to in-memory computations for neuromorphic hardware. A particular eNVM called resistive random access memory (ReRAM) or memristor has received wide attention due to its scalability, compatibility with CMOS technology, analog conductance modulation and low power consumption [27]. Furthermore, using ReRAMs for in-memory computation operations replaces the need for power-hungry and area-hungry analog-to-digital and digital-to-analog converters.

In this work we use the VT memristor model to develop our neural network. This memristor has a novel heat dissipation capability that reduces the resistance variation by almost 30% [28]. The VT memristor also has a competitive cycle-to-cycle variation of only 4%. The high on and off resistance ratio of this VT memristor also provides a stable high and low state that is favorable for binarized neural network.

With the ISI encoding scheme and the VT memristor model a deep spiking neural network is then proposed. The pre and post processing engine of the memristor is discussed in detail in this work that shows how the network can be scaled easily. The proposed system also has a spatio-temporal information processing capability where multiple rows of pixel values can be taken in at a time. A TTFS based output post processing technique is discussed as well that allows to do classifications for different machine learning applications such as digit or pattern recognition. A small-scale hardware model is proposed with using a 3 layer spiking neural network architecture and the accuracy is shown using pixelated digit images of 0-9 as inputs. A 3-layer neural network software model is then developed to test this network with MNIST dataset that has shown to have 86% accuracy.

The main contributions in this part of the thesis includes

1. A novel scalable deep neural network design with compute-in-memory architecture for ReRAM crossbars.

2. A spatio-temporal information processing capability where each row of pixels can be taken in for one clock cycle.

3. Incorporates inter-spike interval encoding scheme for information processing and uses it as an interface between layers.

4. Uses the VT memristor model for the design of the crossbar that allows for matrix-vector-matrix multiplication followed by a current compensator stage that adds variation to compensate for both binary weights as well as leakage current.

5. Evaluation of accuracy on hardware using the pixels from images of digits 0-9 and shows a 100% accuracy with the TTFS classification scheme.

6. A software model to evaluate the large-scale model of the designed neural network and comparing it with the TTFS counterpart.

Chapter 2: Reservoir Computing

The fundamental framework of reservoir computing is discussed in detail in this section. This section also covers the detailed working procedures of the three common reservoir computing framework which includes ESNs, LSMs and Delayed Dynamical Systems.

2.1 Architecture of Reservoir Computing

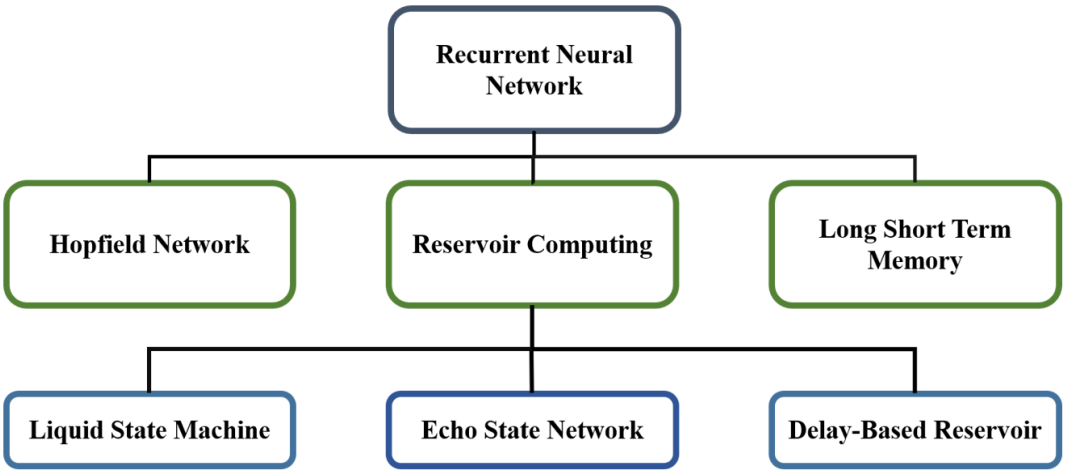

Figure 2-1: Basic Architecture of Reservoir Computing

The overview of the well-known Recurrent Neural Network sections is shown in Figure 2-1. Derived from FNNs, a typical example of RNN is shown in Fig. 2-2. A dynamical memory is made in RNNs that stems from the recurrent connections. The processing of temporal information is made easier in RNNs because each state can depend on previous states and they can be used for tasks such as time series prediction or speech recognition [3].

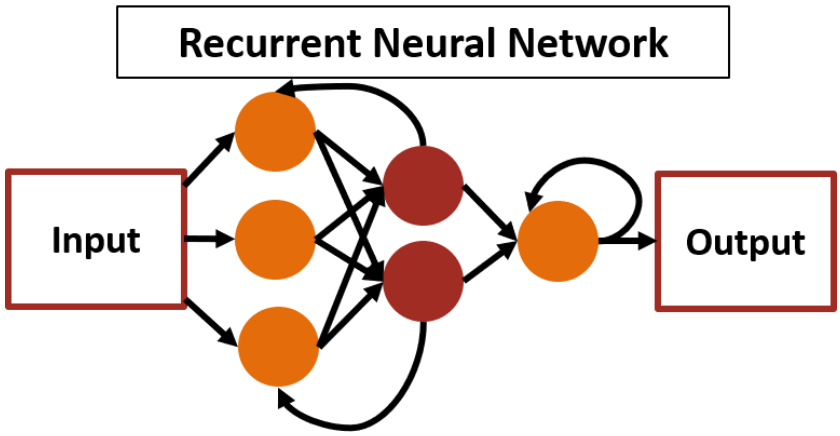

Figure 2-2: Recurrent Neural Network

Hopfield network, a subset of RNN, consists of attractors [29], [30]. The input can be classified into an attractor which represents a class. Each network has a scalar value which is defined as the energy of the network. The nodes in the network are updated and eventually the nodes converge to a local minimum. The major issue with Hopfield networks is that it is unable to process information at any particular moment and only the final result can be read [3]. Long Short Term Memory, another subset of RNN, has shown successful results in handwritten digit recognition and speech recognition [31], [32], [33]. It has feedback connections and is able to process a series of data [34]. Reservoir Computing has recently shown successful results in computing challenging tasks such as speech recognition, grammar modeling, character recognition, generation and prediction of chaotic time series and noise modeling [35], [36], [37], [38], [39], [40], [41], [42]. The inputs are nonlinearly transformed to a high dimensional space to classify the data and passed to the reservoir layer where the inputs are randomly connected, similar to the RNN structure. Only the weights in the output layer are trained as depicted in Figure 2-3 [3], [5], [6].

Since only the output layer is trained, the training process can be simplified to a linear training algorithm such as a linear classifier [3]. The reservoir also implements the fading memory property because each state depends on the values from the recent past due to the recurrent connections and the values from the distant past get faded away over time [43], [44], [45], [46]. The reservoir should also have different responses dynamically to different types of inputs to be able to separate these inputs into different classes. Apart

from the fading memory property and the separation property from the nonlinearly transformed inputs, another important property for reservoirs is that it needs to be able to approximate such that similar inputs are mapped to the same class.

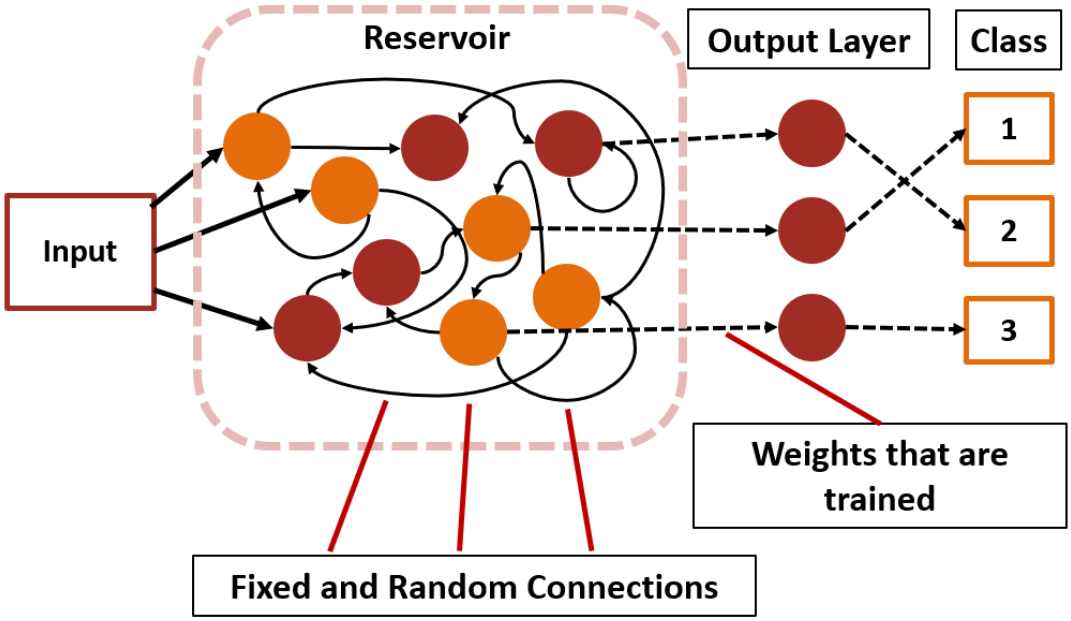

Figure 2-3: Reservoir Computing Architecture

2.1.1 Echo-State Machines

The Echo-State Network (ESN) model was developed in the early 2000s [5], [40], [47], [48], [49], [50]. The ESN functions as described previously where only the output layer is trained. The reservoirs states at each time is a combination of the individual nodes. The state of a node at a particular time step n in the reservoir can be described by Equation (1) [5]. From the equation x(n) is the vector of the node state during the time step n. The input matrix is u(n-1) which is at the previous time step due to causality. W^{in} and W are the input and the reservoir weight matrices respectively that are usually random and are scaled based on the requirements of the system. The function f is the nonlinear activation function, typically a sigmoid, that is used for the high-dimensional projection.

$$x(n) = \mathrm{f}(W^{in}u(n-1) + Wx(n-1) \qquad (1)$$

$$y(n) = gW^{out}x(n) \quad (2)$$

Equation (2) shows the output matrix which is calculated by using an output matrix W^{out} and an output activation function g. This weight matrix is trained to reduce the difference between the target output and the actual output. The echo-state property has to work in order for ESN to work. This property dictates that the reservoir should remove the data from the initial condition [5]. It has also been observed that, if for any input the spectral radius is smaller than unity, then the echo-state property has been met.

2.1.2 Liquid State Machines

Liquid State Machines (LSM) and ESN were developed simultaneously. While ESN is a rate-based approximation, LSM uses a spiking neural network inspired from biological neurons [6]. The Spiking Neural networks (SNN) used in LSM is capable of storing the activation energy in one bit and consume very low power, as low as 20pJ, for each spike [50]. These SNNs were shown to be as powerful as the activation function sigmoid and the threshold neurons [51]. The LSM has a similar structure like the basic framework of Reservoir Computing shown in Figure 2-3 and has an input layer, a liquid layer like the reservoir and an output or readout layer [52]. The synaptic connections to the input and the liquid layer are fixed and random connections. Different inputs will produce different results when projected into the higher-dimensional space. LSMs are shown to be used in embedded systems since they are robust to noise. The dynamics of the reservoir is described is Equation (3) and Equation (4) [6], [53], [54].

$$x(t) = L^m u(t) \quad (3)$$

$$y(t) = f^m x(t) \quad (4)$$

From Equation (3) the t is the continuous time and x is the reservoir state. The input u is a form of spike train and the L^m is the filter that is used to convert the input signals to reservoir states. From Equation (4), y(t) is the output and f^m is the map used in the readout layer. This f^m contains the trained map derived from a simple algorithm. If L^m is chosen

from a class of filter that are time-invariant and has the fading memory property and the separation property and f^m is chosen from a class that exhibits the approximation property, the transformation from input u to output y can be approximated from any degree of precision [5], [53].

2.1.3 Delayed Dynamical Systems

Since the delay property comes in a number of real-life systems, nonlinear systems with delayed feedback are dynamical systems that received a lot of attention [14], [55]. This delayed dynamical system is similar to the ESN model except the entire network of nonlinear nodes connected are replaced one single nonlinear node as shown in Figure 2-4. [3], [12], [13]. The inputs in the delayed feedback system is preprocessed to prevent the loss of parallelism and then fed to one nonlinear node [3], [12], [13], [56]. This preprocessing is referred to as the masking procedure. Different scaling factors are imprinted with the input and time-multiplexed to ensure the system is operating in the transient regime. The input is injected into the nonlinear node and the signal stays in the delay line for a time period τ. The different states in the delay line are referred to as the nodes or neurons of the system. The virtual nodes are separated by a temporal separation θ. This θ is the interval with which the states of the delay line are read out. The time interval between two nodes can be calculated using $\theta = \tau/N$, where N is the time points of the virtual nodes. Finally, in the output layer the node's transient dynamical response is read out and the responses are combined in a weighted sum. The delayed dynamical system in [3] was implemented using a feedback loop and electronic circuitry and showed successful results in the nonlinear autoregressive moving average (NARMA)-10 time series prediction task and the spoken digit recognition task.

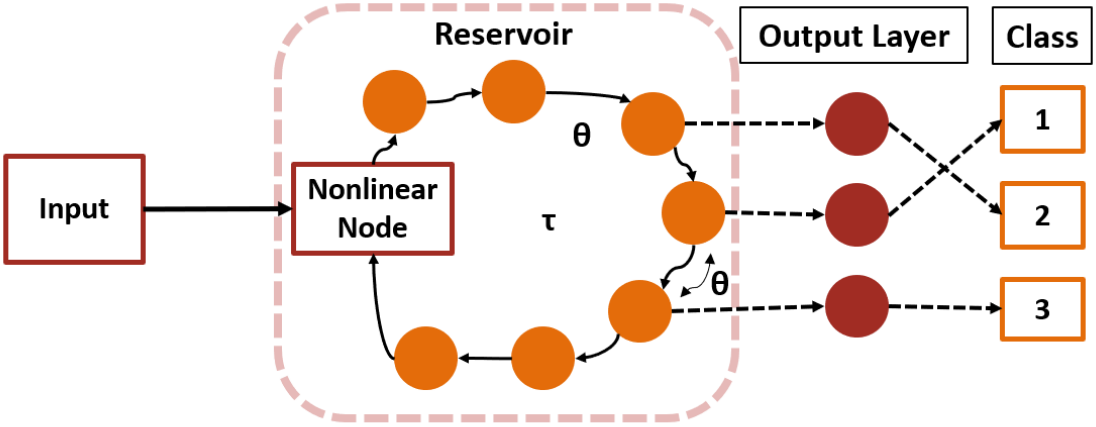

Figure 2-4: Architecture of Delayed Dynamical Systems

The advantage of using this delayed feedback system is that the entire recurrent network system is replaced by one single nonlinear node allowing immense simplification of the hardware implementations of reservoir computing. Therefore, there was an increased number of single node reservoir computing implementations in optoelectronics, optics and electronic reservoir computing. [15], [16], [17], [18], [19], [57], [58], [59]. Two different ways of building delay based reservoir computing is proposed in [13], using decoupled neurons and using coupled neurons through feedback lines. Benchmark tasks were solved from the neurons in the output layer and from the simulation results it can be concluded that both implementations achieve higher performance than using one neuron.

2.2 Electronic Reservoir Computing

Low power and low cost reservoir computing have been widely used in the field of machine learning. Reservoir computing using a single node is discussed in this section which paved the way for simpler hardware implementations. Currently, electronic circuit implementations focus on making high-speed, energy-efficient, low power and noise immune circuits. Reservoir Computing systems implemented with FPGAs, VLSI circuits and memristive reservoir systems are also discussed in this section.

2.2.1 Analog Circuit Implementations

There has been an increased number of analog circuit implementations of single node reservoirs due to their simplicity [3], [12], [13], [59]. A mixed analog and digital implementation of the single nonlinear node concept is discussed [59] that consists of a nonlinear electronic circuit as its major component. The effects of noise are analyzed using a chaotic time-series prediction task and a classification problem. There is an issue of quantization noise that comes from the analog-to-digital conversion process which affects the system's performance. Therefore, the resolution of the conversion is varied, and the system performance is evaluated for two different tasks to observe the noise sensitivity.

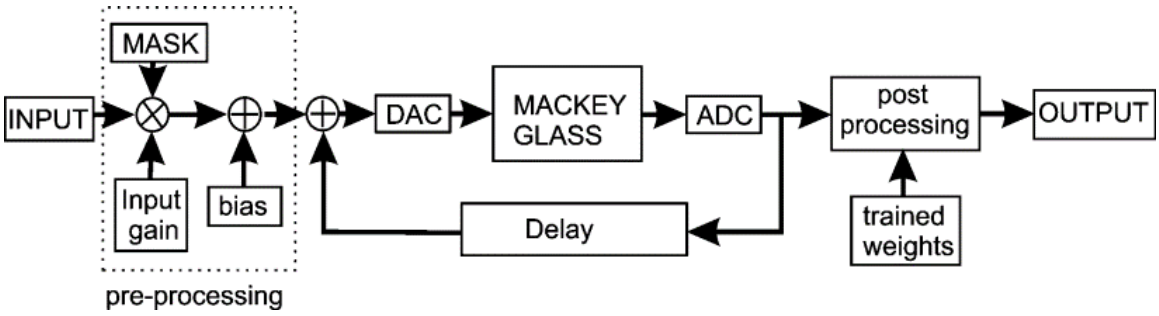

Figure 2-5: Schematic of Reservoir Computing implementation using Mackey-Glass nonlinear function with delay [59].

It can be observed from Figure 2-5. that the delay-based reservoir computing is divided into multiple blocks, with the first stage being the input preprocessing stage to timemultiplex the input data. A Mackey-Glass nonlinearity function is used in this implementation which has the embedded delay property and this nonlinear function is able to spread out the inputs to higher dimensions [60], [61]. The working procedure is similar to the delayed dynamical system discussed in the preceding section. The output is preprocessed and given by a linear weighted sum and these weights are obtained by a simple linear regression from the training process. The Mackey-Glass nonlinearity is implemented in hardware using a summation amplifier, a single bipolar transistor to implement the nonlinear part and another amplifier followed by a filter. The discussed system shows optimum performance and is strong for benchmark tasks. This single neuron design has also been used to implement a delayed feedback reservoir (DFR) using analog hardware [62]. The merging of deep learning and DFR is also discussed. Deep DFR and

multiple-input Deep DFR structures are proposed and compared with leaky ESN with time series prediction tasks. From the results it was observed that both the Deep DFR structures discussed have a much greater performance compared to the leaky ESN.

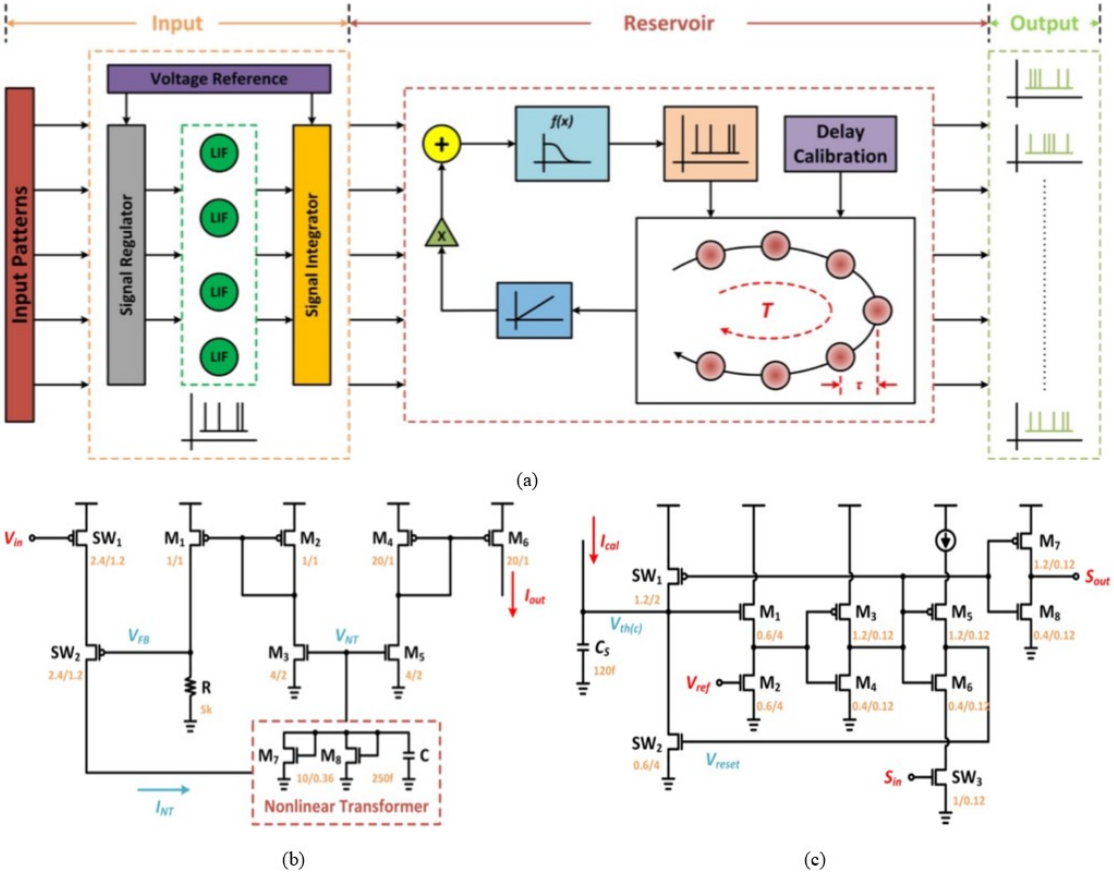

Figure 2-6: (a) Architecture of the DFR system [19]. (b) Circuit implementation of nonlinear system [19]. (c) Circuit Implementation of I&F neuron [19]

A fully analog energy-efficient DFR was designed and fabricated in [19]. As show in Figure 2-6, this design was built using a temporal encoder, a nonlinear transfer function and a delayed feedback loop. The delayed feedback loop follows the similar structure of the Delayed Dynamical system discussed previously. A temporal encoder is used [63] to replace the masking procedure in the input layer. Previously a spike based delayed feedback reservoir was previously proposed by [64] and [65] where spike signals were transmitted instead of analog signals. Similarly, in the DFR design in Figure 2-6. this temporal encoder is implemented where the input patterns are encoded into the distance between the spikes in the temporal spike train. In the reservoir layer the signal is

nonlinearly transformed using an analog circuit configuration shown in Fig. 2-6(b) and converted to spikes after which it is sent to the delay line. The delay time is also calibrated using an operational amplifier and a current mirror array. The circuit implementation of the integrate and fire (I&F) is also shown in Fig. 2-6(c). The delay line is built using a number of I&F neurons where the delay is controlled by the leakage current and the capacitor value. The NARMA10 chaotic time series prediction benchmark was used to evaluate the DFR chip design and it showed a 36%-85% decrease in the error rate compared to the state-of-the-art designs [19].

2.2.2 Field-Programmable-Gate-Arrays

Large amount of data is generated when processing realtime applications. Programmable electronics such as Field Programmable Gate Arrays (FPGA)s can be used to attain such high speeds. The use of reservoir computing using FPGA is also advantageous since the sparse connectivity allows simple wiring techniques that matches with the FPGA requirements [66].

A reservoir computing framework using stochastic bitstream framework is discussed in [39]. Analog neurons were used to build this reservoir computing system on an FPGA. In [9], an LSM system was built on FPGA for real-time speech recognition. The hardware architecture implemented was based on serial processing of dendritic trees using serial arithmetic. The designed reservoir computing system was used to port the speech recognition application to quantized hardware architecture.

In [67], a single-node reservoir computing system was implemented in FPGA. Genetic algorithms were used in this time-delay reservoir system for the dynamic property optimization. The discussed system was prototyped in simulation and evaluated in speech recognition task and verified using FPGA. The experimental results obtained from the FPGA based reservoir computing matches the ones obtained with simulations from noise which showed that this implementation can be applied to physical systems where noise is present.

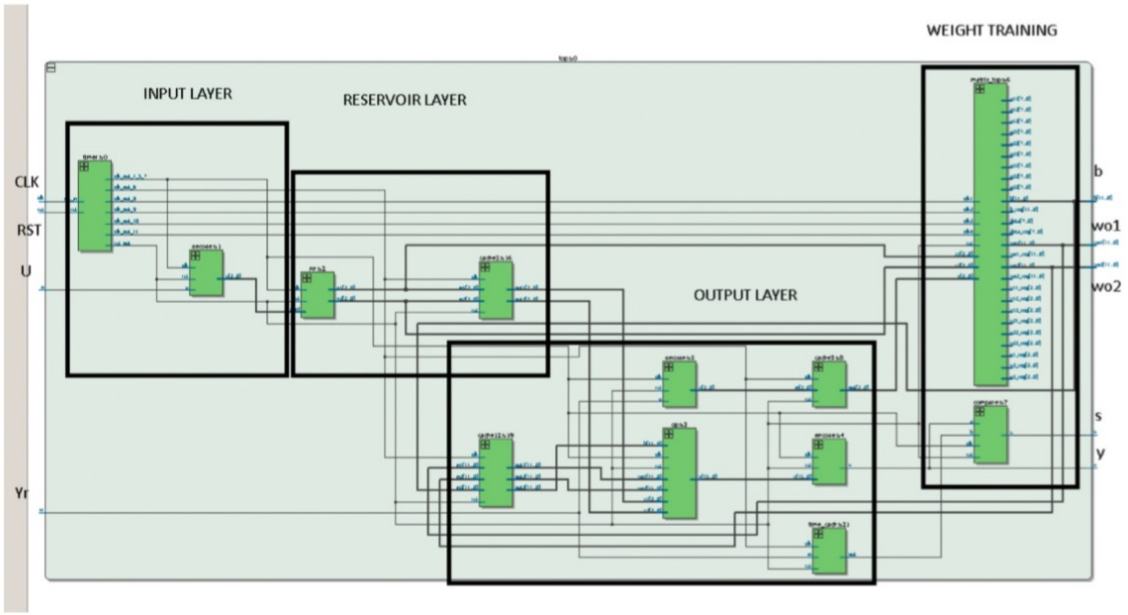

Figure 2-7: Register Transfer Level Schematic of ESN in FPGA [68]

A real-time hardware based FPGA spike-time dependent encoder and reservoir design is discussed in [68] which uses the ESN architecture and is implemented using a reservoir. The discussed design can be trained and implemented in FPGA without the use of any software implementations. In Figure 2-7. the Register Transfer Level (RTL) schematic of the ESN in FPGA is shown that consists of four layers, the input layer, the reservoir layer, the output layer and the weight training layer. The four input signals include the clock signal (CLK), the input signal (U), the reset signal (RST) and the source of the target output signal (Yr). The five output signals are the output signal (Y), the bias signal (b), the output weights (w) and the output matching signal (S). An encoding circuit is proposed followed by an analog-to-digital converter since the FPGA is a digital platform and the provided inputs are analog. An FPGA based Stochastic ESN is discussed in [66] for timeseries forecasting. The stochastic architecture discussed uses comparatively less area than typical hardware implementations. This enables the use of low cost FPGA devices for ESN implementations.

2.2.3 Very-Large-Scale-Integrated Circuits

Reservoir Computing based on pulse signals and spiking signals are becoming more popular due to their high-speed and low power in VLSI implementations. In [69],

real-time wideband signal processing algorithms are processed using mixed signal Printed Circuit Board (PCB) and a digital Application Specific Integrated Circuit (ASIC) prototype of a signal processor. It implements Reservoir Computing as the signal processing approach where the architecture of the reservoir is achieved by using Asynchronous Pulse Processor (APP) for processing analog signals.

An LSM with dendritically enhanced readout layer is described in [70] which has an architecture derived from the nonlinear processing properties of the dendrites. Compared to parallel perceptron readout this architecture has shown to have higher performance with binary synapses that is suitable for VLSI implementations and the discussed learning method is able to choose the connections between the inputs and the dendritic branches that are most feasible. A scalable and hardware-efficient reconfigurable digital LSM model is proposed in [71] that is able to process real-time data. This model uses spatial locality property in the approach and has shown an average accuracy of 85% for epileptic seizure detection. A low power VLSI-based LSM is proposed in [72] for data intensive machine learning algorithm such as speech recognition. The proposed LSM uses online learning method which is local so that the weight update depends only on the presynaptic and the postsynaptic neuron. This reduces communication between the neighboring elements and thus eases the VLSI implementation.

2.2.4 Memristive Reservoir Computing

The switching dynamics and electrical behavior of memristive devices emulate the behavior of synapses and neurons. This makes memristors suitable candidates for brain-inspired computing [73]. Memristors can be used to build large-scale crossbar arrays to form neural networks and can perform in-memory computations [73] [74], [75]. They can interact with analog signals directly and this reduces the cost and energy consumption that arises from the use of analog-to-digital and digital-to-analog converters [73].

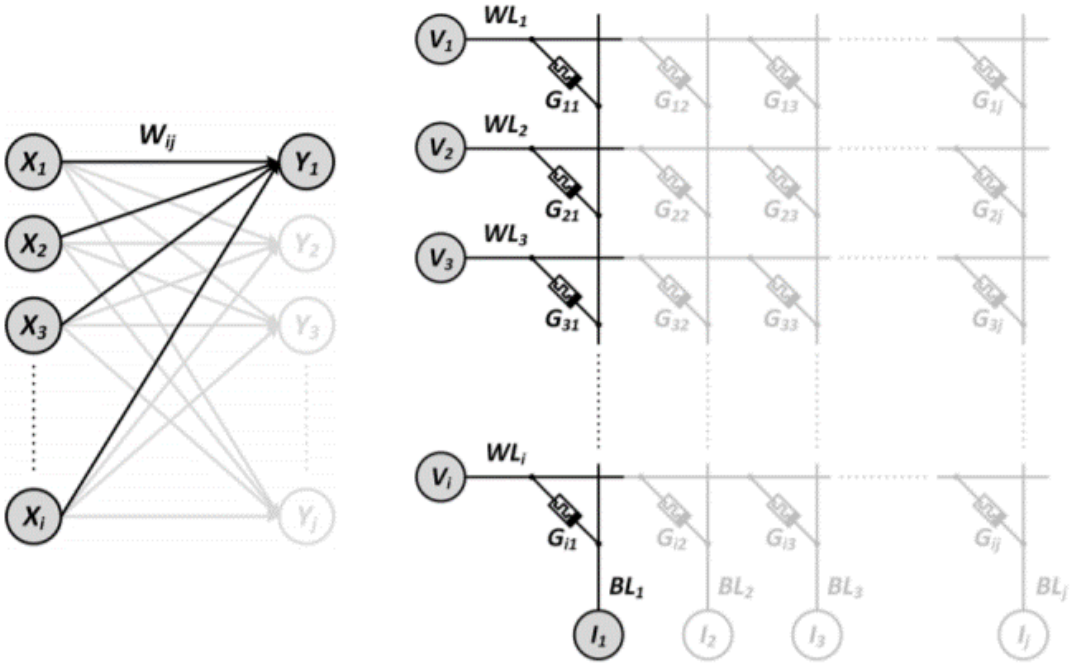

Figure 2-8: Vector-matrix computation using memristor-based crossbar [20]

Memristors can be used as reservoirs to transform inputs to high-dimensional space. A training efficient hybrid Deep Neural Network (DNN) is discussed in [20] that uses memristive synapses in hierarchical information processing method and delay-based spiking neural network (SNN) as the output readout layer. The movement of signal between the DNN layers depend on vector-matrix multiplication. Resistive Random Access Memory (ReRAM) used is a type of memristor that is a two-terminal metal-oxide device which performs like a variable resistor [20]. The memristor crossbar array used in the vector-matrix computation supports a large number of signal connections within a small area, as shown in Figure 2-8. and is a promising candidate to be used in neural networks. The memristive based reservoir computing structure described is capable of handling data-intensive computations and has shown successful results in handwritten digit classification and spoken digit recognition. A Deep Reservoir computing structure is discussed in [76] that shows a memristor configuration with heat dissipation capability to reduce variation. The evaluation of the accuracy was done using a Deep DFR model and the design area, power and latency were significantly reduced compared to typical SRAM memory techniques. Dynamic memristor devices were used in [77] to build a memristor-based

Reservoir Computing system that offers internal short-term memory and demonstrated that only 88 memristor could be used to solve tasks such as handwritten digit recognition.

2.3 Other Types of Reservoir Computing

Aside from electronic reservoirs there are many other types of physical reservoirs including photonic reservoirs, mechanical reservoirs, spintronic reservoirs and biological reservoirs [46]. Advances and trends in photonic reservoir computing are discussed in detail in [46], [78] and [79]. The initial photonic reservoir computing was developed in [80] where a network of coupled Semiconductor Optical Amplifiers (SOA) was used to build the reservoir. The first passive silicon photonics reservoir was developed in [81] which was used to compute Boolean logic operations with memory and use successful results in spoken digit recognition.

The complex behavior arising from the body dynamics of soft and compliant robots can be used to achieve the nonlinear dynamics of reservoir computing [46]. Mass-spring systems can be used to achieve nonlinear behaviors in mechanical reservoir computing [82]. The input to output mapping can be obtained by adding a static and linear readout. Charge and spin of electrons are also being used to develop nanoscale electronic devices in the field of spintronics. These devices can have non-volatile memory and can be used in the reservoir [46]. A nanoscale spintronic oscillator is developed in [83] that demonstrated a similar accuracy using spoken-digit recognition task compared to the state-of-art neural networks. In [84], spin waves were used to develop a reservoir computing device that uses the nonlinearity of the spin waves arising from the magneto-electric effect. Through the spin waves, the developed device was able to generate the essential properties of reservoir computing including high-dimensionality, fading memory and nonlinearity. A different type of spintronic reservoir device can be generated using magnetic skyrmion [85]. A skyrmion network in magnetic films is used to make a two terminal device that has nonlinear characteristics.

2.4 Applications of Reservoir Computing

As discussed previously, Reservoir Computing has been used for machine learning applications due to their simplistic training procedure where only the output layer in trained. It has been used in data-intensive applications such as speech recognition, time series forecasting, short-term memory, signal processing and pattern classification [9], [10], [11], [35], [36], [37], [38], [39], [40], [41], [42].

In [7], an intelligent stock trading system is proposed where genetic algorithm is used to improve the trading rules, and these were used to provide trading suggestions with ESN. A predictive ESN classifier is introduced in [8] that makes use of both ESN and competitive state machine framework. In speech classification experiment the described ESN classifier has proven to be more robust compared to hidden Markov model. LSMs also showed successful results in speech recognition [9]. The use of LSMs in pattern recognition is discussed in [10] and [11]. In [11] it is shown that the LSM with self-organizing network using spike-time-dependent plasticity has better performance in pattern classifications. The applications of Reservoir Computing are demonstrated in [86] through isolated handwritten digit recognition on MNIST dataset and detecting the status of a door using moving pictures from a camera.

Reservoir Computing has also been applied in the field of communication systems. In [87] a Deep-ESN is proposed for symbol detection in 5G multiple-input and multiple-output orthogonal frequency-division (MIMO-OFDM) systems. The proposed design uses both memristive synapses as well as dynamic reservoir layer to improve computation capabilities. The use of ESN in MIMO-OFDM system for symbol detection is also discussed in [88] where it is shown that this method has better performance compared to typical symbol detection methods. Using a deep RC for MIMO-OFDM signals in [89] was also shown to have a faster learning convergence and a reduction in unknown nonlinear radio frequency interference. Reservoir Computing can be used to implement deep reinforcement learning (DNL) by using the temporal correlation of dynamic spectrum access (DSA) network [90]. From the experimental results it was observed that the Reservoir Computing based approach can decrease the collision probability of secondary users with other primary users and secondary users. Reservoir Computing was also used

for attack detection strategies in smart grids [91]. From simulation results it has been shown that the proposed design is able to detect attacks under different attack variations.

Reservoir Computing has also been used to model biological systems, for instance the cerebellum was modeled using LSM that is more computationally powerful than perceptrons [92]. The hippocampus was also modeled using reservoirs, where the bottom layer was constructed using recurrent nodes and fixed weights [93]. Reservoir Computing is also used in patient-adaptive model for monitoring electrocardiogram and from the simulation results it can be concluded that the system provided a cost-effective, accurate and fast patient—customized heartbeat classifier [94].

2.5 Conclusion

Reservoir Computing is a recently developed machine learning framework derived from RNNs. Since only the output weights are trained, Reservoir Computing greatly simplifies the training complexity compared to RNNs which has led to a significant development in this area. With the development of single node reservoirs in delayed dynamical systems, the hardware implementations were also immensely simplified since the entire recurrent network can be replaced by one nonlinear node.

This review summarizes the basic architecture of reservoir computing and discusses the working procedure of the three commonly known subsets including ESN, LSM and Delayed Dynamical Systems. It provides a focus on electronic reservoirs, discussing the various developments in analog circuits, FPGAs, VLSIs and memristors to build reservoirs. Ongoing research is being carried out to determine the best suited electronic reservoir in terms of speed, energy and power efficiency and scalability of reservoirs. Scalability of reservoirs is a challenging aspect in this field and future research is being carried out in developing miniature emerging devices, such as memristors and spintronic devices, to build reservoirs that are capable of handling immense data. Other types of physical Reservoir Computing are mentioned as well including photonic, mechanical, spintronic and biological reservoirs. Due to their spatio-temporal information processing capability and simplified training procedures, Reservoir Computing has applications in various

machine learning applications such as speech recognition, image classification and time series forecasting.

Chapter 3: Spiking Neural Network

3.1 Architecture of the Spiking Neural Network

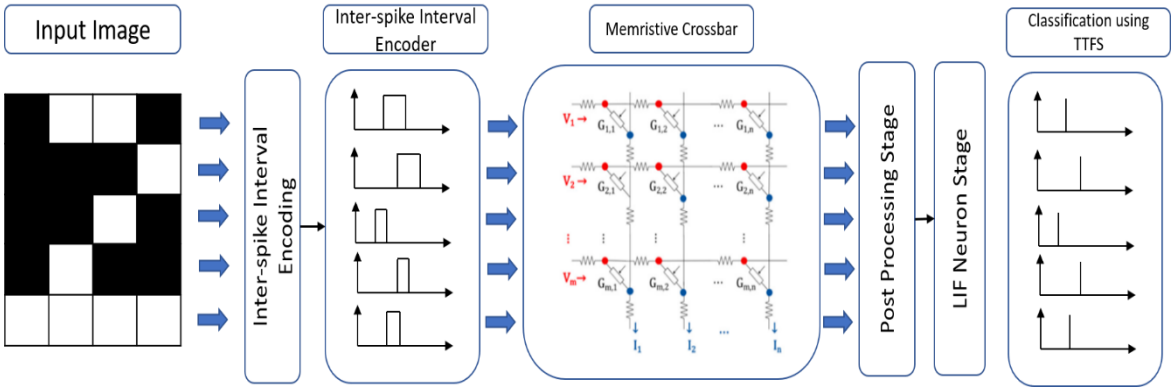

Figure 3-1: Spiking Neural Network Design

A spiking neural network structure was developed using the memristive crossbar shown in Figure 3-1. The VT memristor model was used for the memristive crossbar design. The inputs are applied to the ISI encoding module to encode the data into a pulse-width-modulation (PWM) signal. The network has a spatio-temporal information processing capability where the input signals can be applied parallelly at each row of the crossbar that increases energy and power efficiency. The memristive crossbar contains trained weights that are used for feature extraction from the inputs. These input PWM signals are then applied to the memristor crossbar to compute the matrix-vector-matrix multiplication operation.

The memristive crossbar in this design are built with high resistance state (HRS) and low resistance state (LRS) values. The VT memristor has a very high on and off resistance state of 989MΩ and 1MΩ. The output signal from the crossbar is the accumulated current signal from each bitline of the crossbar. This current signal is then transferred to a post processing stage which consists of a current amplifier and then passed to a Leaky-Integrate-and-fire (LIF) neuron stage. The LIF neuron charges up based on the amount of current generated and then generates one spike. This is the output stage of the network where the output result depends on the time to first spike and the neuron which spikes. This neural network has a scalable architecture, and the dimensions can be increased further by replicating the blocks of the ISI encoding scheme and the crossbar.

The following sections discuss the detailed circuit design of each block and a hardware implementation using the pixels from the digits 0-9.

3.2 The Memristor

The three basic circuit elements include the resistor, the inductor, and the capacitor. In 1971 Professor Leon Chua from the University of Berkeley developed the fourth basic circuit element called the memristor [8]. The three fundamental devices, resistor, capacitor and inductor cover the relationship between current, voltage, charge and flux shown in Figure 3-2. Dr. Chua found the missing link between charge and flux which can be described by the memristor using the equation:

$$d\Phi = M * dq$$

Here M represents the memristance, Φ is the flux and q is the charge stored.

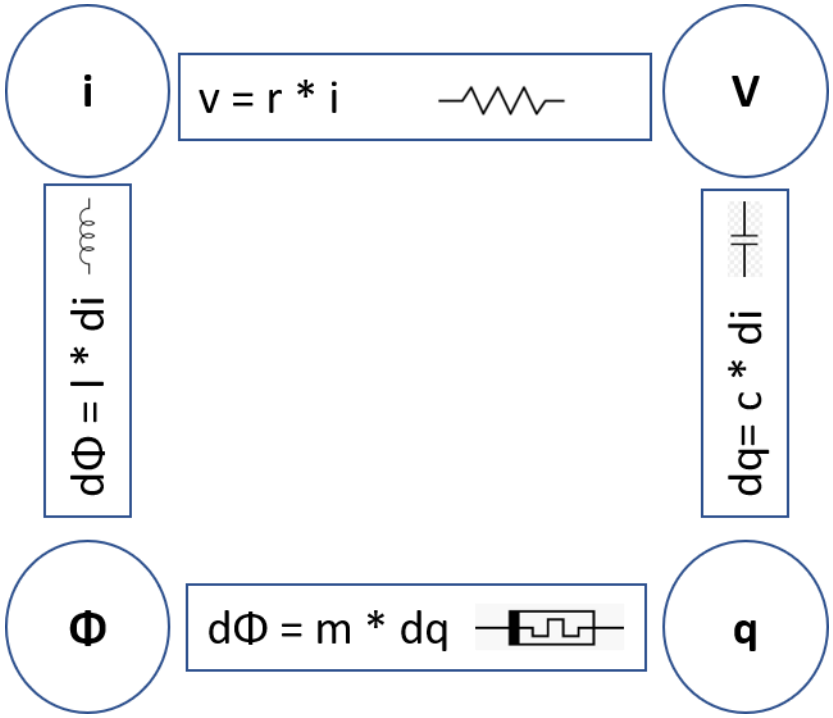

Figure 3-2: Relationship between the 4 Key Components

The memristor is often referred to as a memory resistor and it is a two-terminal device where its resistance increases when charge flows through it in one direction and decreases when the charge flows in the opposite direction. When no voltage is applied

across the memristor, it remembers its previous state which is a unique property in memristors. The current-voltage characteristic curve of the memristor also has the shape of a hysteresis loop. As frequency increases, the memristor starts behaving more and more like a resistor as depicted in Figure 3-3 [95-96].

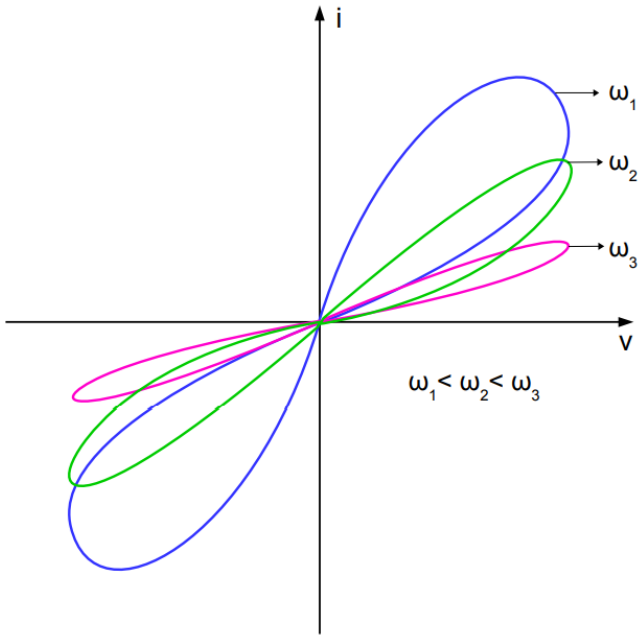

Figure 3-3: I-V Characteristic of Memristor [97]

This memristor is an emerging non-volatile memory (eNVM) technology that has been immensely studied in the recent years and has shown to emulate neurons and synapses to build neuromorphic hardware. The conventional memory technologies of SRAM, DRAM and flash rely on charge storage phenomenon. SRAM stores charges at the cross-coupled inverter nodes, DRAM stores charges at cell capacitor and in flash memory the charge is stored in the floating gate of the transistors. However, the major problem with these memory technologies is scalability which causes the stored charges to be lost and hence exacerbates the performance and introduces noise and reliability issues. Besides being scalable, the new memory technologies need to have non-volatile storage, low operating voltage, high endurance, long retention time, several synapse strength levels, capable of implementing synaptic learning as well as a simple framework [98-100]. Based on these characteristics, several emerging non-volatile memory (eNVM) technologies have been developed which include phase-change random access memory (PCRAM), spin-

transfer-torque magnetic random access memory (STT-MRAM) and resistive random access memory (ReRAM). Out of the discussed eNVMs, ReRAMs or memristors have received wide attention due to their scalability, compatibility with CMOS technologies, analog conductance modulation and low power consumption [101].

The typical ReRAM structure consists of a metal-insulator-metal (MIM) structure shown in Fig. 12 [102]. The ReRAM devices should be able to switch from a high-resistance state (HRS) to a low-resistance state (LRS) demonstrating the resistive switching behavior. The switching behavior of memristors is dependent on both the choice of the metal electrodes as well as the choice of the oxide. The switch from HRS to LRS is called the set process while the switch from LRS to HRS is called the reset process. As depicted on Figure 3-4, unipolar switching depends on the amplitude of the voltage while bipolar switching depends on the polarity of the voltage. Compliance current is a value set to prevent the permanent dielectric breakdown and prevent the damage of the memristive device.

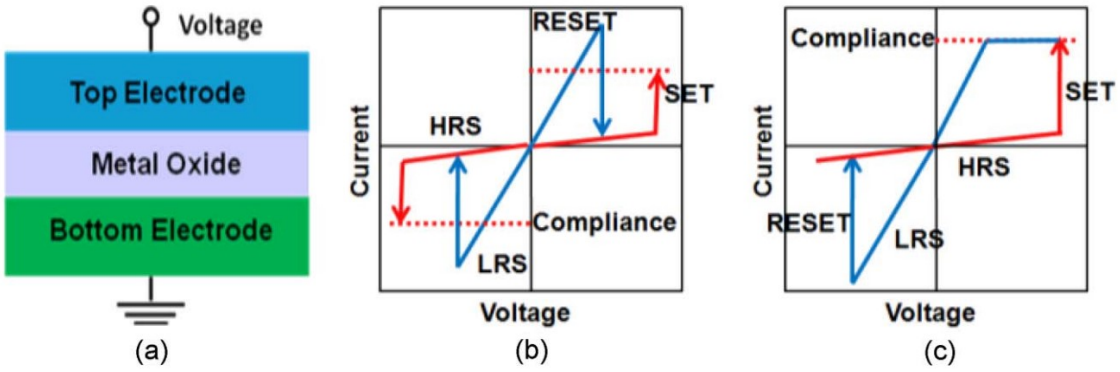

Fig 3-4. (a) ReRAM structure. (b) Unipolar ReRAM operation. (c) Bipolar operation [102].

3.2.1 Memristive Crossbar

The memristive crossbar is shown in Figure 3-5. Based on the MIM structure from Fig. 3-4, the memristor is placed between two nanowire layers so that the memristor can be located at the crossing point [103]. Because of this crossbar structure, the memristor can be used to implement large scale neural networks since it has high density, non-volatile,

nanoscale structure, low operating voltage, low power consumption and can be used for in-memory computations [103].

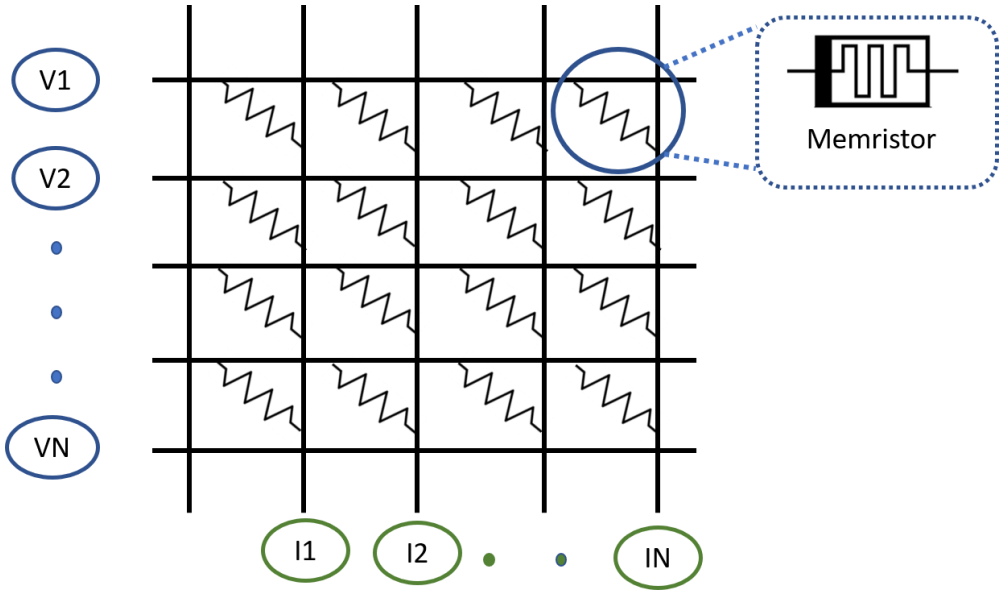

Figure 3-5. Memristive Crossbar

The read and write operations for a memristive crossbar is shown below in Fig. 3-6 [104]. In a memristive crossbar a single cell reading and writing is allowed. The horizontal lines or rows are the wordlines and vertical lines or columns are the bitlines. To perform a read operation on the memristor located in the first row and 4th column, a voltage of V_{read} is applied to the wordline and a voltage of 0 is applied to the bitline. For all the other wordlines and bitlines, the voltage is set to 0. For the selected memristor shown in green, the corresponding current is the one at the bitline. The blue cells are the half selected memristors and the red cells are the unselected memristors that have 0 voltage drop across them. The other cells have a voltage half of the read voltage applied to them or a voltage of 0. This keeps them in a non-conducting state because in an ideal case the voltage does not exceed the threshold that would allow the memristor to change its state. The current I_{leak} is the leakage current that occurs due to nonidealities and even when the voltage is less than the threshold voltage some current is conducted which is a current area of research in memristors. The write operation is carried out similarly where the voltage V_{write} is applied

across the selected cell while half of that voltage is applied to the other wordlines and bitlines [104].

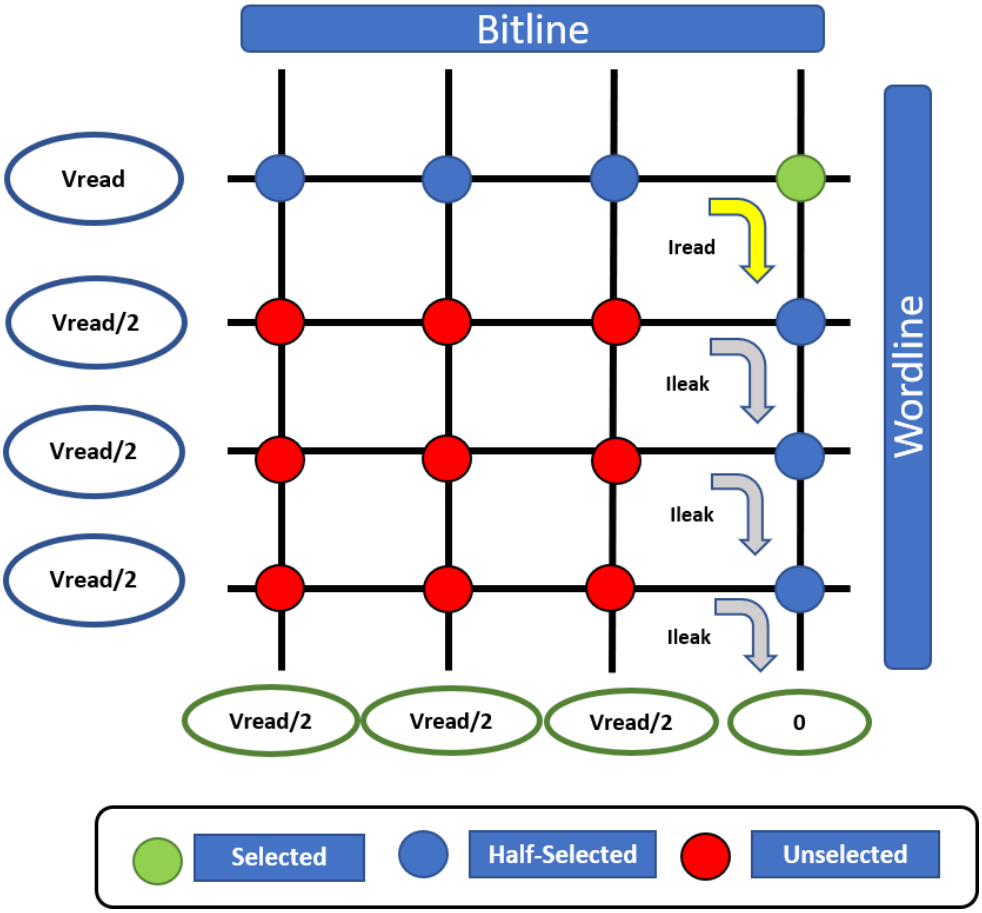

Fig 3-6. Read and Write Operations on Memristive Crossbar

Using the read and write operations the memristive crossbar is used to build large scale neural networks since they can be trained and the weights can be written in the respective memristors [103, 105]. A single layer neural network can be used to show how the memristive crossbar can be used for training shown in Figure 3-7. If there is an input matrix of X containing n elements and an output vector of Y containing m elements, in a single layer neural network, the relationship between X and Y can be described as

$$Y_n = W_{nxm} \times X_m$$

The W_{nxm} matrix is the weight matrix that can be implemented using the conductance states of the memristors inside the crossbar. When the training data is applied at the input, the weight matrix gets updated continuously following the equation

$$\Delta W_{ij} = \mu \frac{\partial (y - y*)^2}{\partial wij}$$

From the equation W_{ij} is the synaptic weight connecting the input and output neurons while μ is the learning rate. The weight matrix W which can be implemented by the memristive crossbar is updated continuously until the difference between the output and the target output y* is minimized [105].

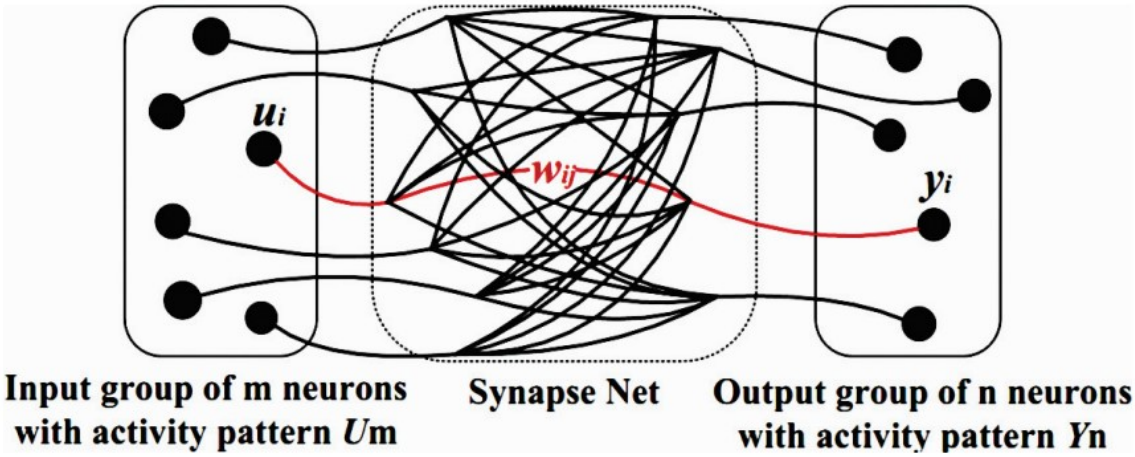

Figure 3-7: Single Layer Neural Network [103, 105]

3.2.2 The VT Memristor

One of the major drawbacks of memristors is their low reliability that prevents them from being used in neural network implementations since it reduces the inference accuracy [102, 106]. The resistance variation is one of the reasons for low reliability. In [107] through a study it was found that the resistance variation comes from the heat accumulation during the metal atom diffusion of the conductive filament (CFs). The VT memristor was developed to solve this problem by having an additional heat dissipation layer to inside the metal electrodes to compensate for resistance switching variation by more than 30% [108].

From Figure 3-8, initially the bonding inside the metallic oxide is strong but when a high voltage is applied the oxygen ions move to the metal electrode. This creates oxygen vacancies inside the metallic oxide and causes the conductive filament to form. The metal atoms from one of the electrodes gets reduced to cations and then migrate to the cathode that has the inert material. When these ions migrate, the forms the conductive filament that connects the two metal electrodes to each other and turns the device into the set process called the on-state R_{on}. When the polarity is switched between the two electrodes, the conductive filament gets destroyed and the resistance increases and the devices switches to the off-state called R_{off} which is the reset process. A significant amount of current flows in the set and reset process and if the heat is not dissipated on time, the temperature of the conductive filament increases and causes a significant metal atom diffusion. This contributes to the resistance variation and eventually reduces inference accuracy [108].

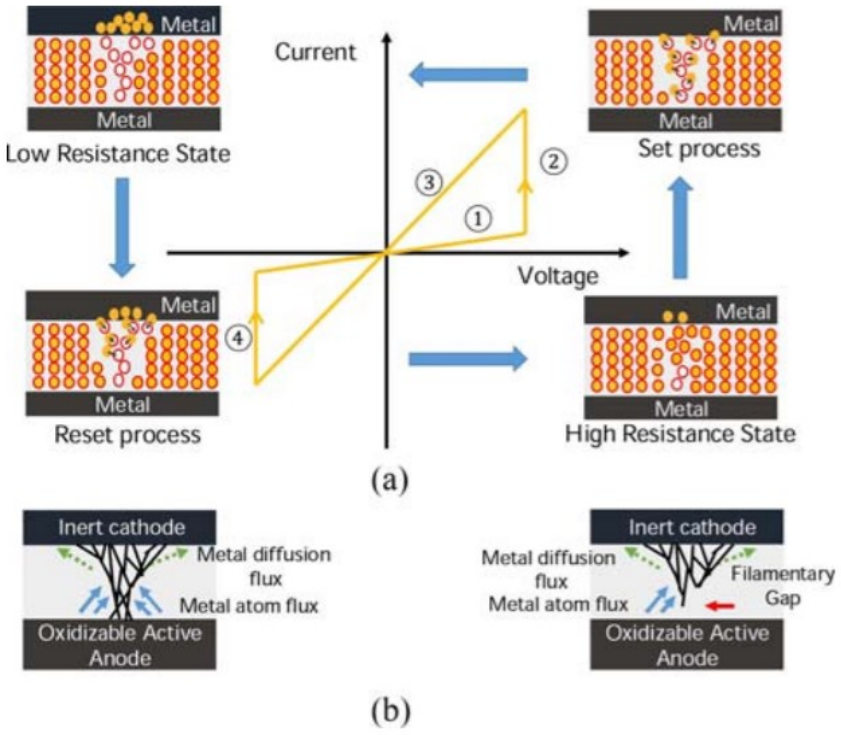

Figure 3-8: Memristor (a) set and reset process (b) Formation of Conductive Filaments [108].

For the VT memristor e-beam evaporation was used to deposit the metal electrodes and solid electrolytes. For the oxide layer Ta_2O_5 pellets were deposited using evaporation. This memristor can achieve a high resistance state of 980MΩ and a low resistance state of

1MΩ which provides a high on and off ratio for the resistances. Due to this high ratio the leakage current problem is mitigated. Figure 3-9 shows the fabricated memristor die and the memristor crossbar. Furthermore, in the VT memristor the additional metal dissipation layer of Cr metal is used to mitigate the heat related problem [108].

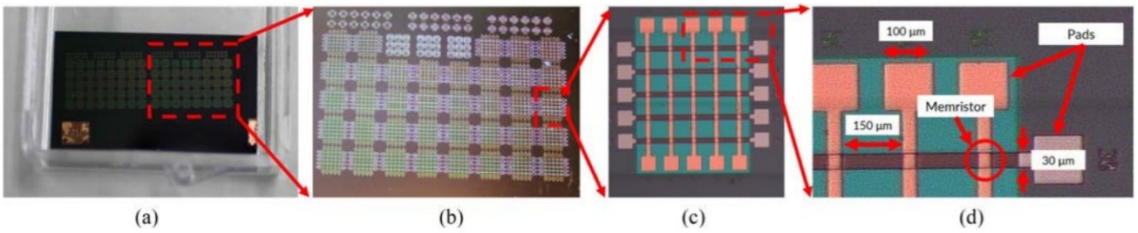

Figure 3-9: Fabricated VT Memristor die (a) die; (b) Zoomed-in view; (c) 5x5 crossbar structure; (d) Memristor at Crosspoint [108]

3.3 Inter-Spike Interval Encoding Scheme

3.3.1 LIF Neuron

The key component in this Inter-spike interval (ISI) encoding scheme is the LIF neuron. The structure of the LIF neuron is shown below in Figure 3-10.

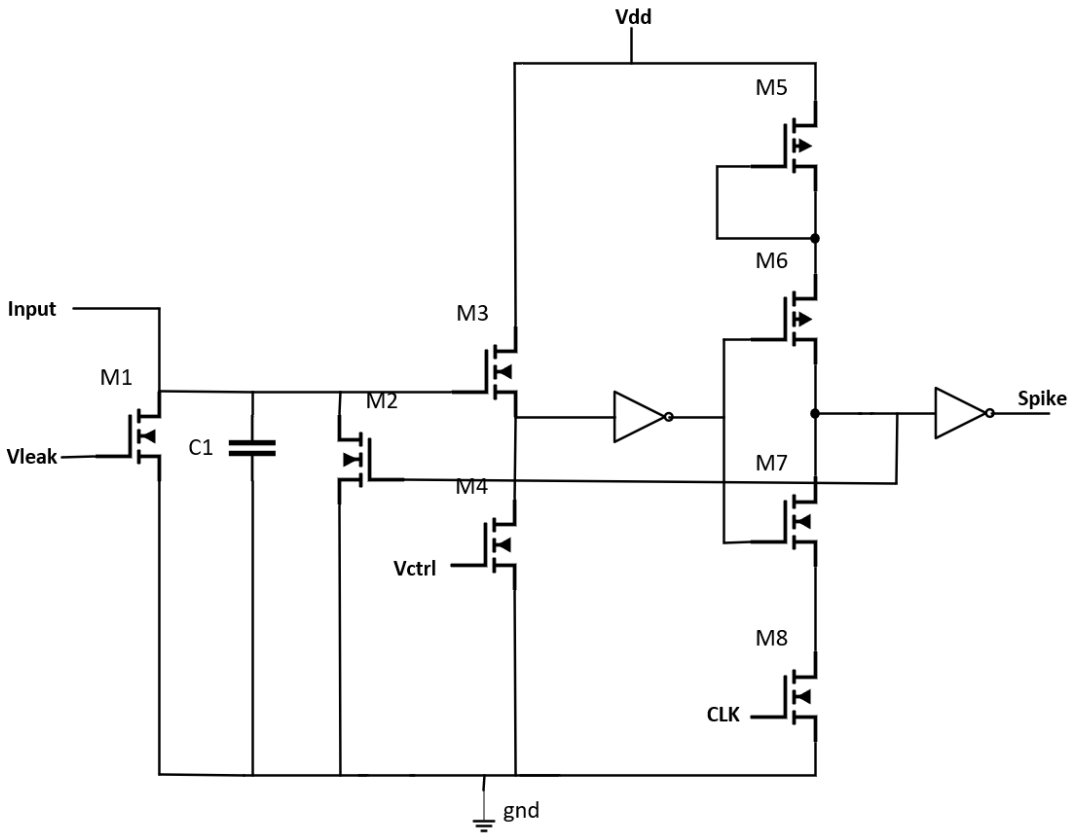

Figure 3-10: LIF Neuron

In the biological nervous system, impulses are produced that are transmitted in form of spikes. Similarly, we use this LIF neuron model in this neural network to generate the spikes and convert the incoming signals to spikes. The neuron takes a current signal as the input and the incoming current is the excitation current that can be written as

$$I = C(\frac{dV}{dt}) + I_{leak}$$

From the equation, C acts as the membrane capacitance of the neuron and V is the membrane voltage. I_{leak} is the leakage current of the neuron. This leakage current is controlled by the NMOS transistor connected in parallel with the capacitor and V_{leak}

represents a very low gate voltage. The clock signal controls the frequency of the spikes. The input excitation current charges up the capacitor and the voltage across it increases. Once it exceeds the threshold voltage which is set to 1V, a spike is fired and passes through the buffer stage. When the spike is fired and the clock is low, the voltage is reset to 0V by the feedback transistor.

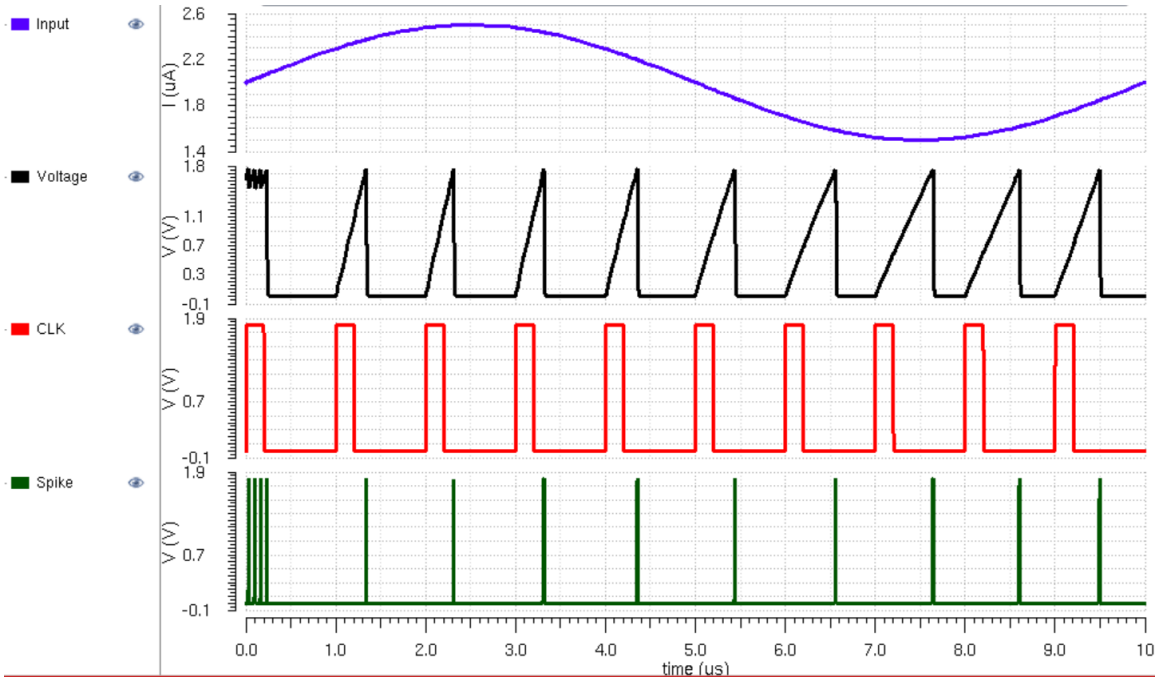

Figure 3-11: LIF Neuron Plot

The plot in Figure 3-11 shows the LIF neuron output using an excitation current of 500nA with an offset of 2uA. The second plot is the voltage across the capacitor. When the clock signal is high, the current charges up the capacitor and the voltage increases. Once it exceeds the threshold voltage, the signal passes through the buffer stage and a spike is fired. It can also be noticed from the plot that the higher the amplitude of the current, the sooner the spike occurs with respect to the clock signal. The average power consumption of the neuron is only 4.37µW. The layout the LIF neuron is also shown below in Figure 3-12, covering an area of 21.37µm x 21.28µm.

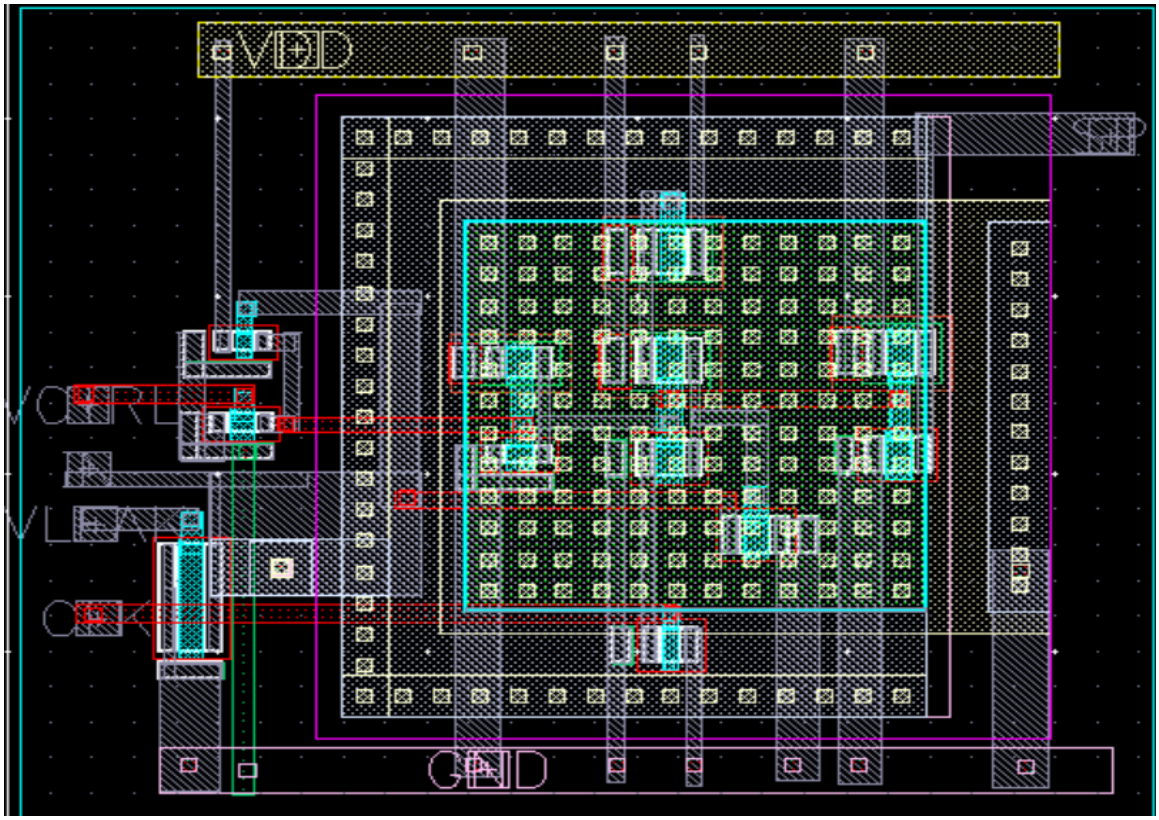

Figure 3-12: LIF Neuron Layout

3.3.2 Current Mirror Stage

Before the ISI encoding stage is made, a precision current to current converter needs to be used for the input preprocessing part. For this stage, an opamp configuration is used with a current mirror configuration shown below in Figure 3-13. The opamp used in this design is a very low power opamp that was made with minimum transistor sizes and consumes a DC power of 64.62µW.

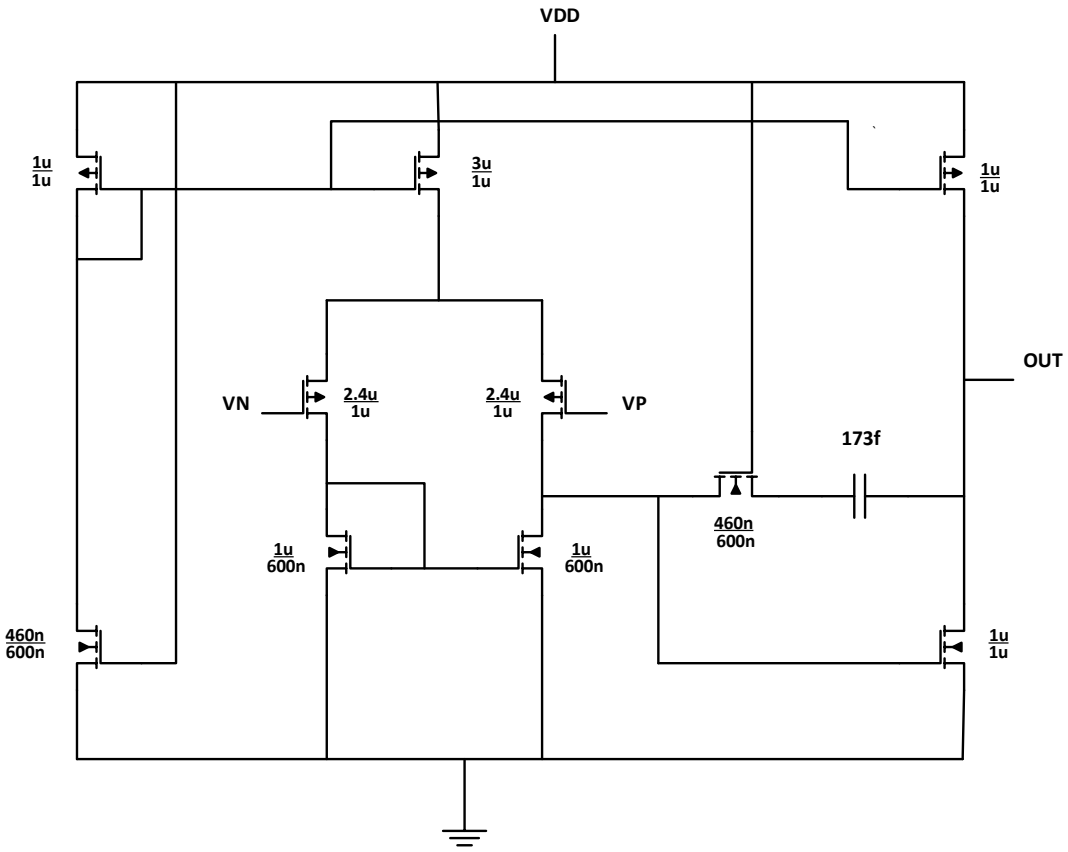

Figure 3-13: Opamp Design

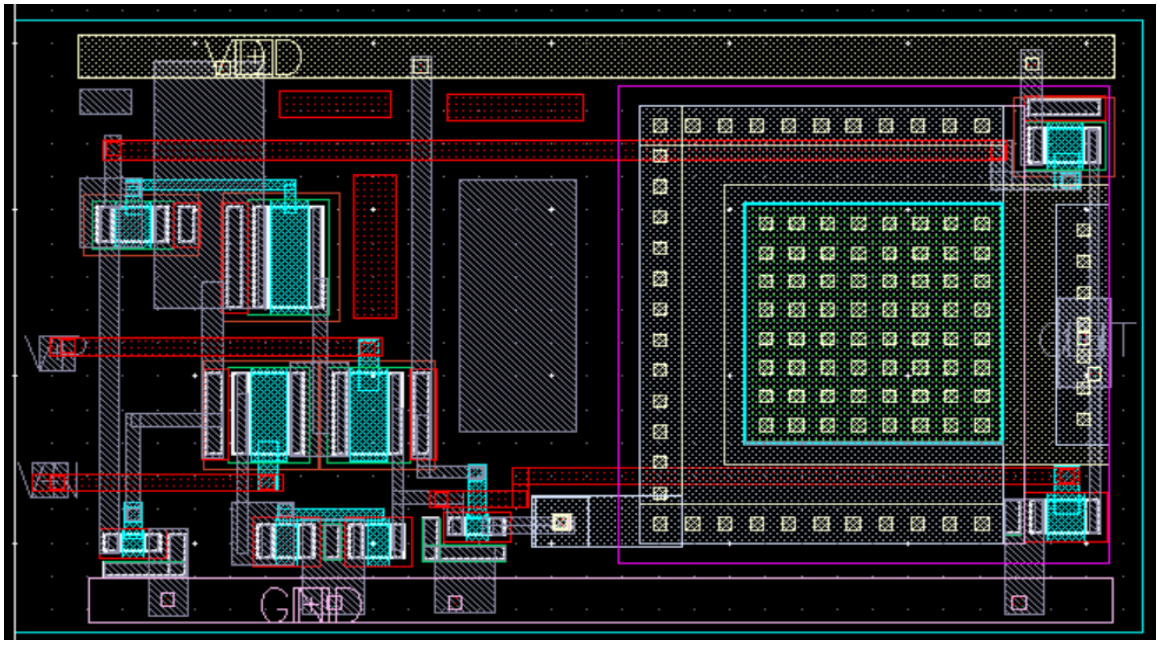

Figure 3-14: Opamp Layout

Using this opamp the precision current to current converter is designed that is used to preprocess the incoming input signal shown in Figure 3-15. Since the opamp has high input impedance, this configuration can be used to drive the neurons in the next stage. The input current signal is applied to the noninverting input of the opamp while the inverting input is grounded. This current then passes to the drain of the transistor M1 and transistors M2 and M3 are matched so the current can be copied from the left hand side to the right hand side.

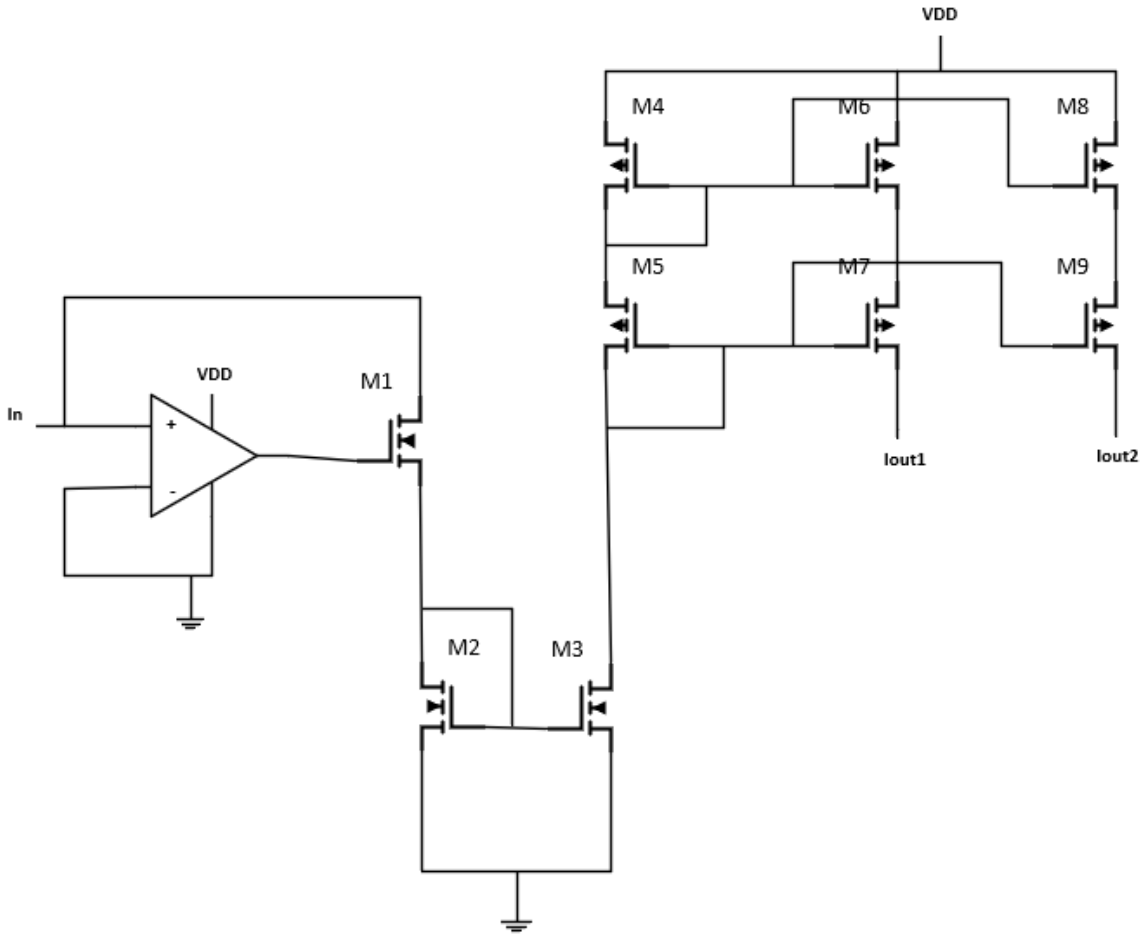

Figure 3-15: Current Mirror Stage

To have an efficient current mirror structure, a cascode structure is used with a diode connected PMOS stage as a load. This prevents the variation of the drain to source voltage of the PMOS transistor to affect the output current. Using two different current ratios, in this stage the input current is converted to two different current values of I_{out1} and I_{out2} and fed to the neuron stage explained below.

3.3.3 ISI Encoding Block

After the current signal is split into two levels using the precision current to current converter, it is applied to the encoding stage which consists of two LIF neurons and an extractor followed by a buffer stage shown in Figure 3-17. In the current mirror stage I_{out1}

is set to have a higher current value than I_{out2}. When applied to the LIF neuron, I_{out1} causes the first neuron to fire the spike first and turns on the NMOS transistor. Once this transistor is turned on, the capacitor is charged up to VDD and remains at this point. When I_{out2} causes the second neuron to fire, the capacitor can discharge the voltage across it. This produces a pulse-width modulation signal that can be applied to the memristor crossbar to conduct the matrix-vector-matrix multiplication. The resulting output plot from the ISI encoding stage is shown below in Figure 3-16.

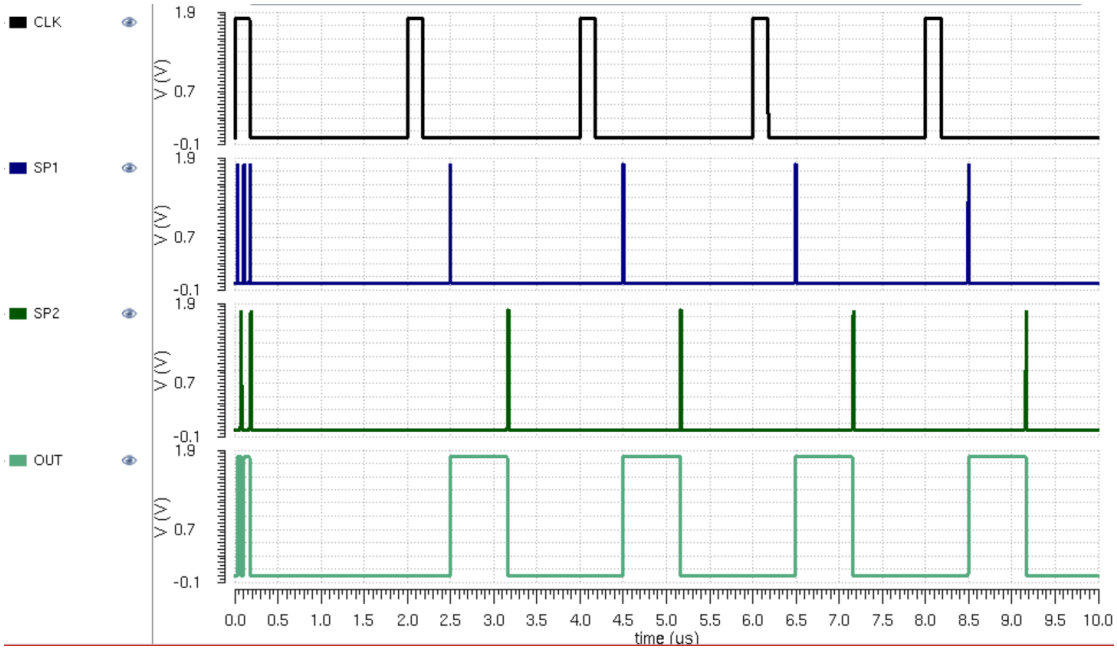

Figure 3-16: ISI Encoder Plot

Using the same clock signal for both the LIF neurons, a current of 2µA is applied to neuron 1 and a current of 1.2µA is applied to neuron 2. Neuron 1 generates a spike first and charges up the capacitor and neuron 2 generates the spike afterwards that discharges the capacitor. The resulting output signal is shown by the plot OUT in Figure 3-16.

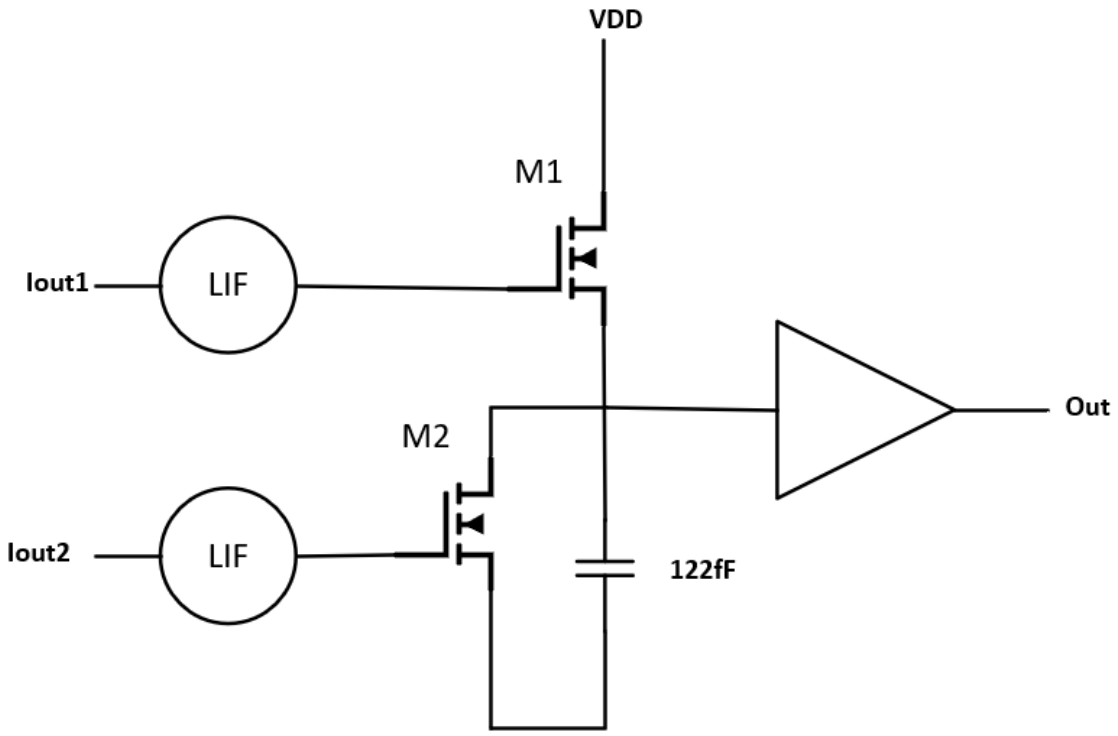

Figure 3-17: ISI Encoder

3.3.4 Intermediate Stages

The output signal from the memristive crossbar is a current signal. The memristive crossbar contains a weighted matrix with the memristor value set to a specific resistance. When the pulse-width-modulation signal from the ISI encoding stage is applied to the memristive crossbar, it is multiplied with the resistance values and the output from the crossbar is the accumulated current from each column of the crossbar. This SNN is a scalable network more layers can be added by simply replicating the input preprocessing stage in the intermediate layers. This is done by using the current mirror stage and the encoding stage after the crossbar. The current output from the crossbar can be split into two and then applied to two LIF neurons. After that the extractor will be used to generate the PWM signal.

3.3.5 Output Stage

For the output stage the current output is taken from the columns of the memristive crossbar and then passed to the current amplifier to reduce the loading effect as shown in Figure 3-18. The similar current mirror structure can be used from the previous stage and the amplified current generated from the current mirror can be applied to the LIF neuron. The LIF neuron then generates a spike based on the amount of current from the crossbar. The time to first spike and the column from which it spikes can be used as a method for classification in this design.

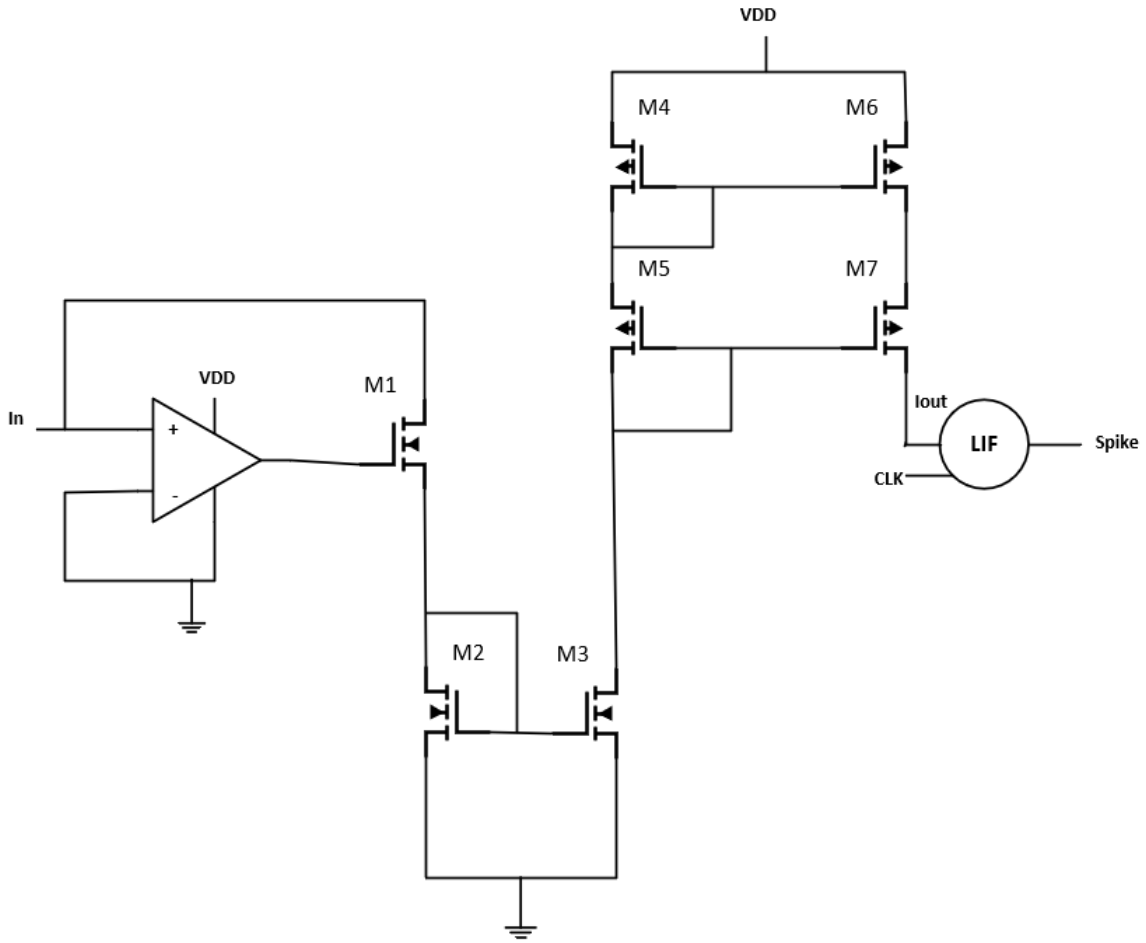

Figure 3-18: Output Stage Circuit

3.4 Analysis and Verification

3.4.1 Hardware Simulations

To evaluate the SNN design we use the images 0 to 9 as inputs to the network. A three layer neural network is used to process the images. Since a small scale neural network is used with an output layer of 4x4 crossbar, the weights are trained to identify the digits 0 to 3. The hardware simulation architecture is shown in Figure 3-19.

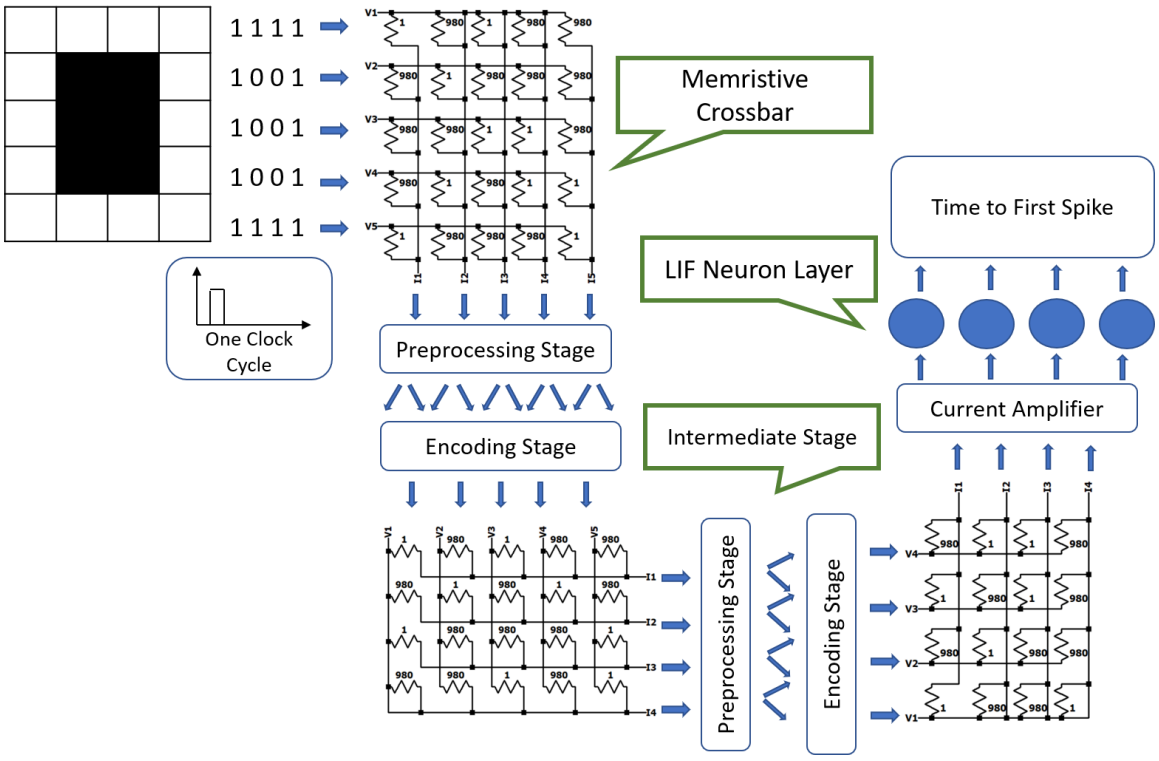

Figure 3-19: Hardware Simulation Architecture

The image pixels are taken in for one clock cycle and applied to a 5x5 crossbar. The crossbar has the weights shown in the Figure 3-20. The input signals are preprocessed first by applying them to the weighted input crossbar to add some variation to the signal. The resulting output current I1 to I5 is shown below when the image pixels from the number 0 is applied to the crossbar. IN1 to IN5 are the input signals to the system and 1 – 5 are output currents. From the output of the first crossbar there are three levels of output currents, 0μA, 1.8μA and 3.6μA that are the results of the multiply and accumulate operations.

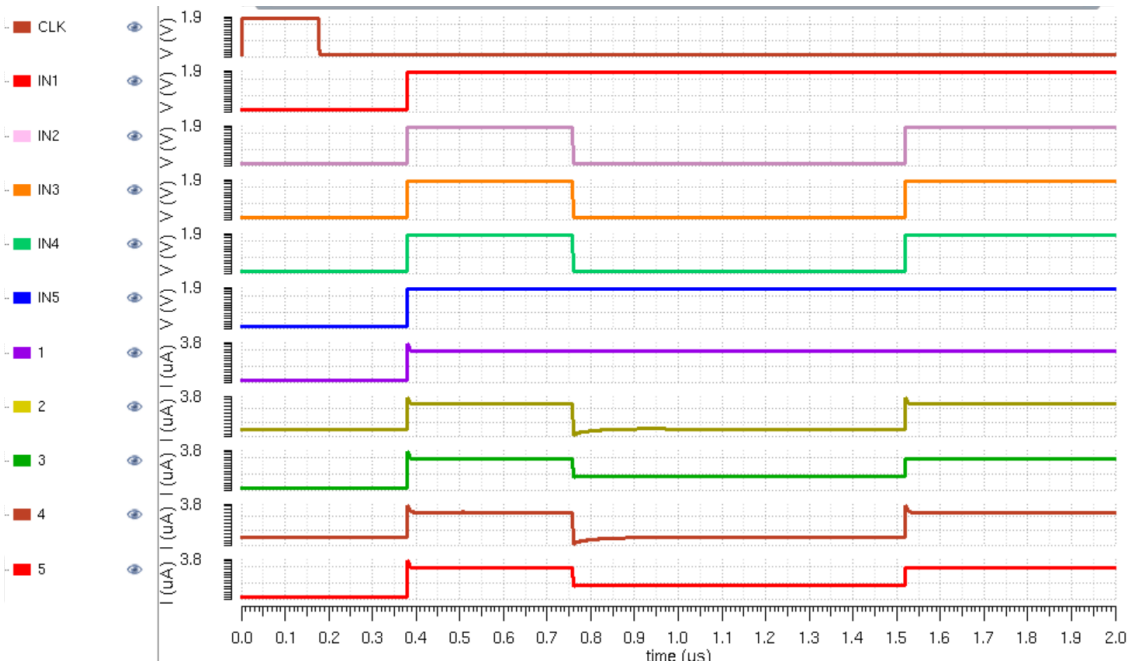

Figure 3-20: Input Stage Plots and Output Plots from Crossbar

The resulting output currents are then applied to the preprocessing stage which is the precision current to current amplifier. This stage splits the current signal into two values of different weights and applies the following signal to the ISI encoding stage. As discussed in the preceding section, the ISI encoding stage then applies the current signal to 2 LIF neurons and then using the extractor generates the pulse-width-modulation signal which is the output from the ISI encoding stage. The resulting plot from the ISI encoding stage is shown below in Figure 3-21.

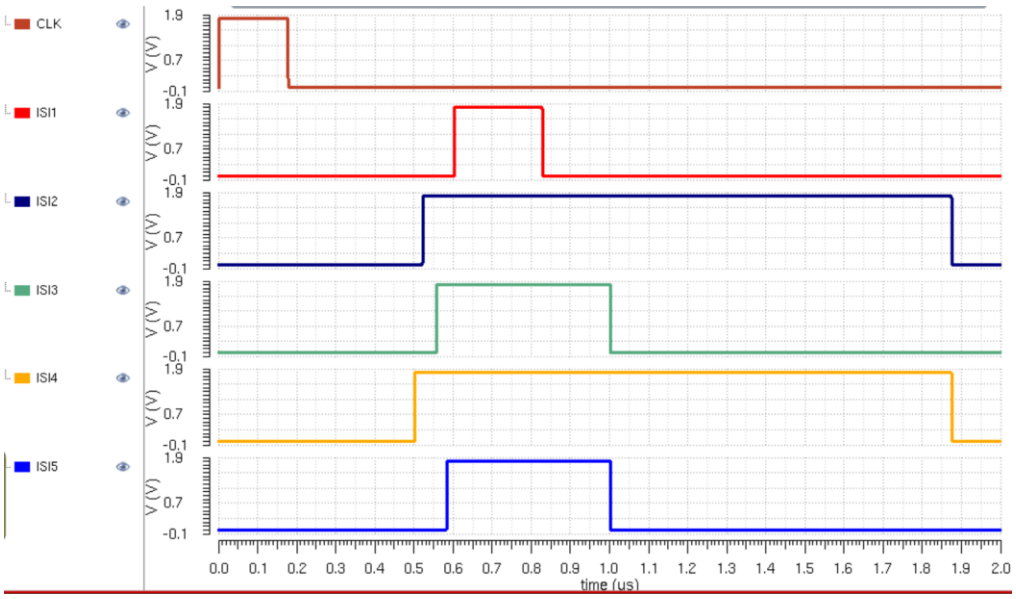

Figure 3-21: ISI Encoder Outputs

The five output signals from the ISI encoding stage are then applied to a 5x4 crossbar with the weights specified in the Figure 3-19. The similar multiply and accumulate operation occurs at this stage where the signals from the ISI encoding stage are multiplied with the crossbar weights and the output currents from this stage are shown below. Due to the multiply and accumulate operation, the four output currents have different values shown in Figure 3-22.

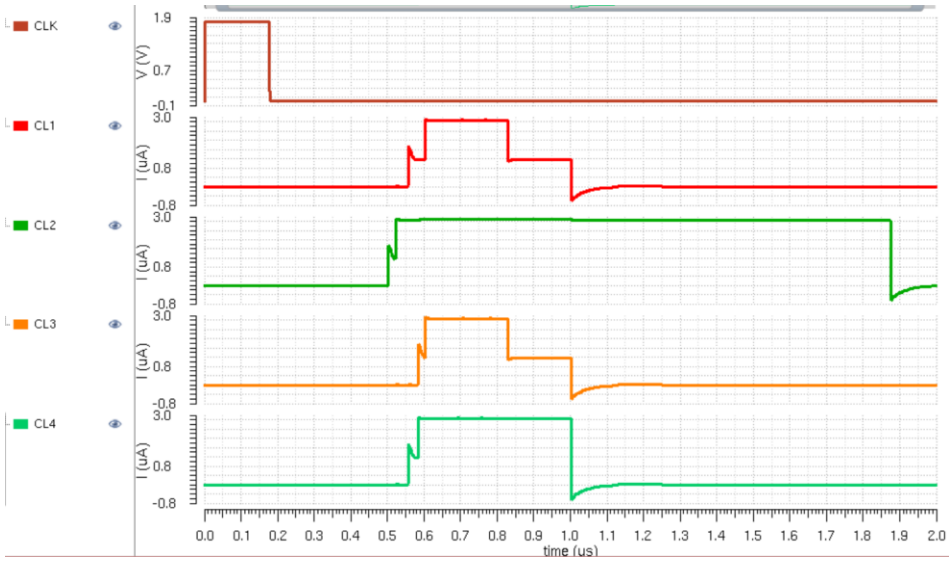

Figure 3-22: Output Currents from Second Crossbar

The resulting current outputs are again applied to the current mirror stage where they are split into two levels and then applied to the ISI encoding stage. In the ISI encoding stage the LIF neurons generate the spikes based on the two current levels and use the extractor to then produce the pulses shown in Figure 3-23.

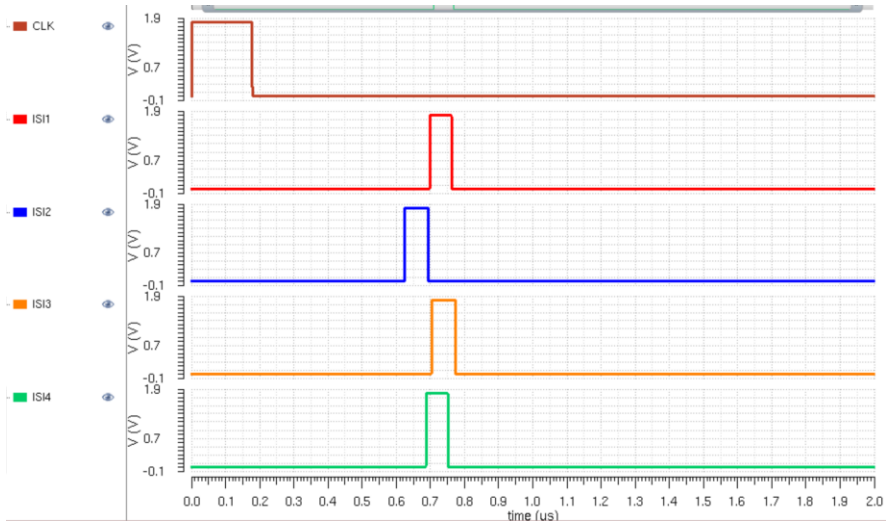

Figure 3-23: ISI Encoder Output Plots from Second Stage

The resulting output pulses are applied to trained output crossbar to compute the multiply and accumulate operation. From the trained output crossbar, the accumulated current outputs are shown below in Figure 3-24.

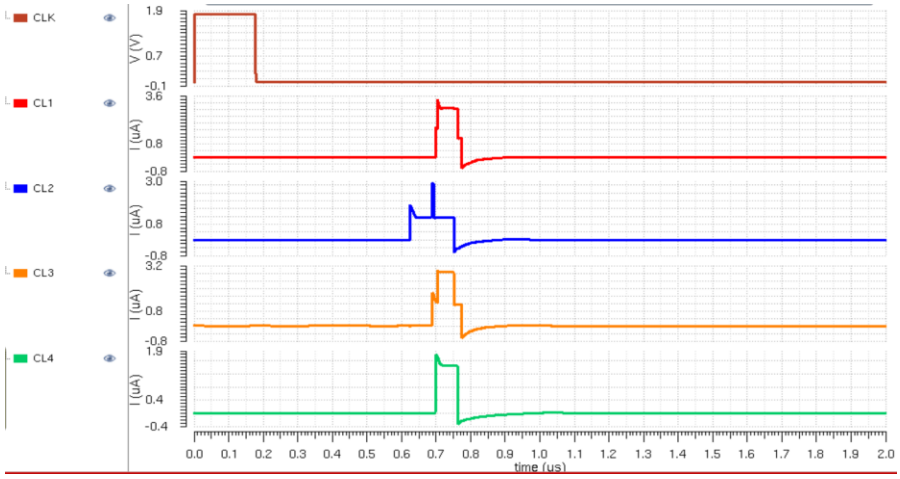

Figure 3-24: Current Outputs from Crossbar

The output current signals are then applied to a current amplifier stage discussed in the preceding sections and then an LIF neuron layer to generate a spike shown below in Figure 3-25.

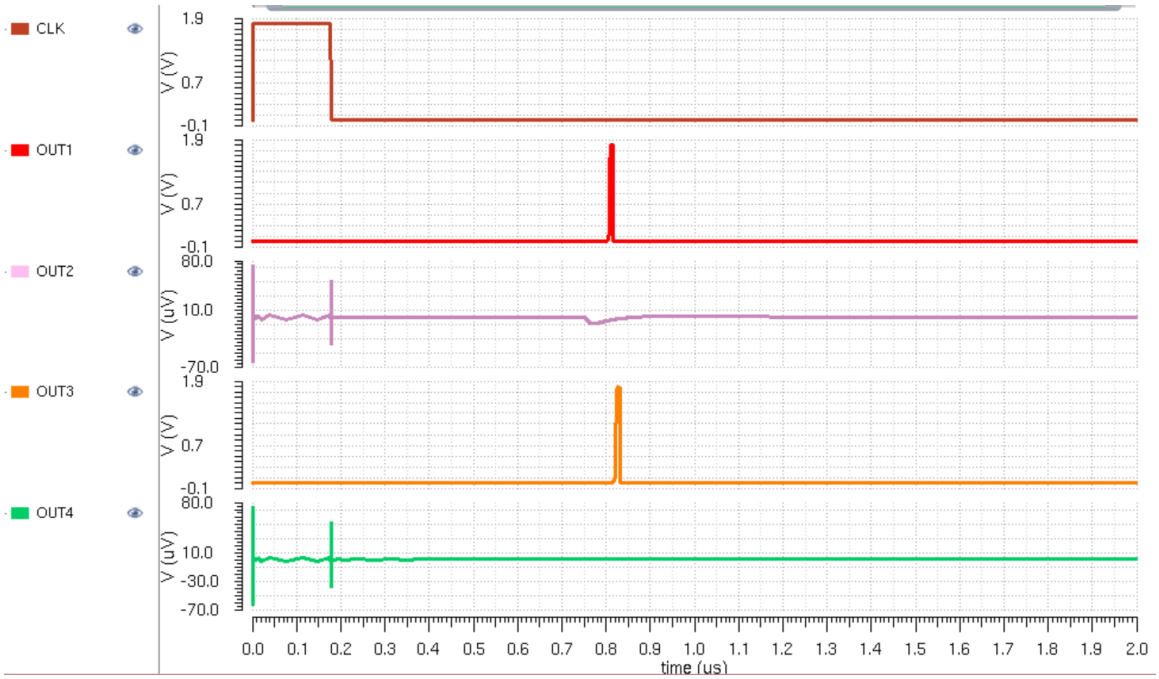

Figure 3-25: Final Output as TTFS Signal

The way the output classification happens is based on the time to first spike method. The output is determined by which neuron and from which column spikes first. Since the hardware design uses a 4x4 crossbar, the output weights are trained to classify images 0 to 3. For image 0 the first, for image 1 the second neuron spikes first, for image 2 the third neuron spikes first and for image 3 the fourth neuron spikes first. Even though the output crossbar is only 4x4 and therefore is able to classify 4 digits, the rest of the outputs from the digits 4-9 are applied to show that the spiking pattern for each neuron is also different which can be used to classify the images. The image pixels used for classification are shown in Figure 3-26.

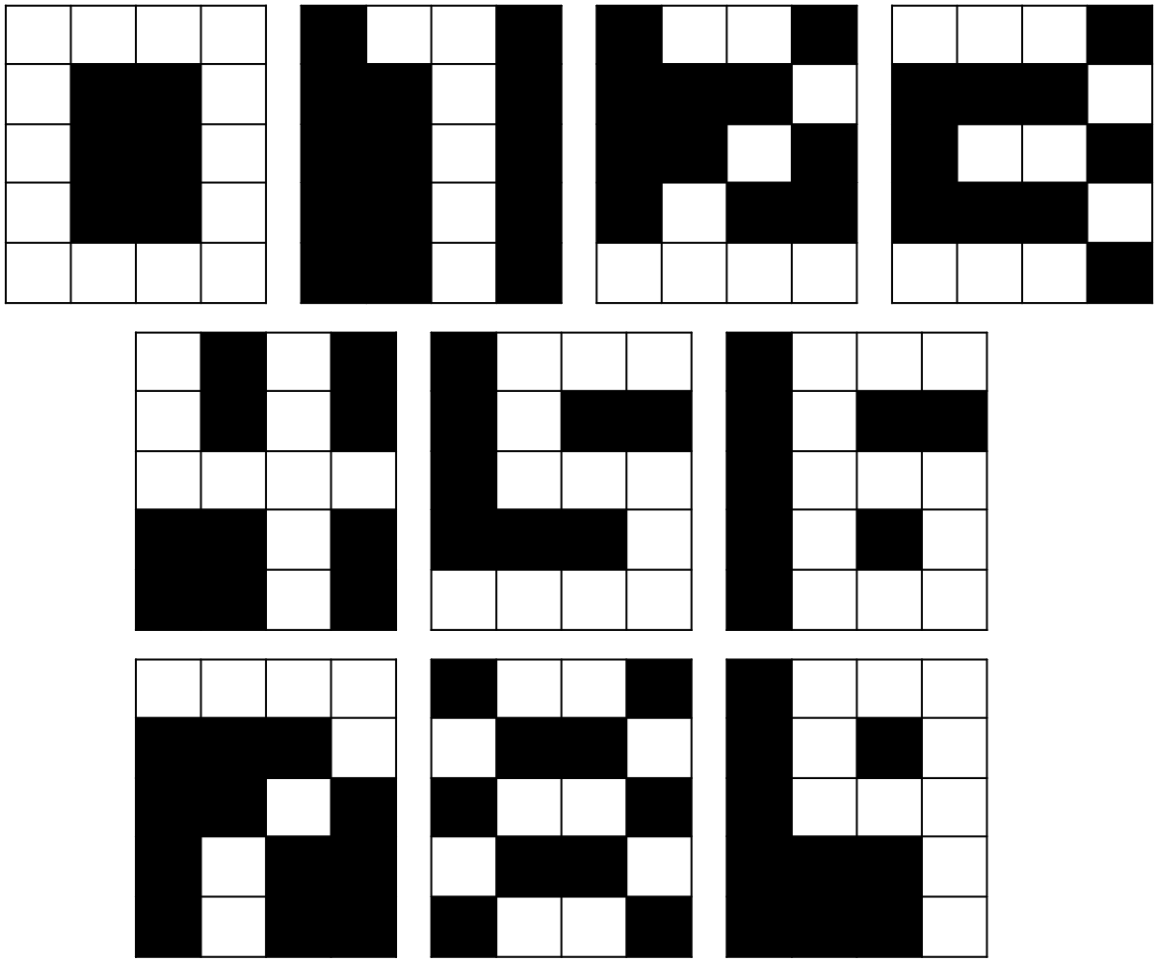

Figure 3-26: Digit Images used for Hardware Simulations

The following Table 1 shows the output from spike time from each digit. It can be observed that for digits 1 to 3 each column spikes first from Column 1 to Column 4 respectively. The spike time from the other digits are shown to demonstrate the difference in spiking pattern and how they can still be classified. From the hardware results the design shows 100% accuracy in recognizing all digits.

Number	Column 1	Column 2	Column 3	Column 4
0	805ns	0ns	821ns	0
1	0	1.5µs	1.51µs	0
2	1.39µs	1.25µs	1.24µs	1.4µs
3	1.02µs	1.47µs	1.1µs	993ns
4	1.02µs	925ns	1.02µs	985µs
5	1.1µs	0	0	0
6	1.19µs	0	1.2µs	0
7	0	0	1.12µs	0
8	1.17µs	0	1.2µs	0
9	1.4µs	1.3µs	1.4µs	1.36µs

Table 1: Output Spike Times for each digit

3.4.2 Power Breakdown

The power breakdown from the circuit components is shown below in Table 2. The power consumption for the neuron is the average power consumption since the neuron on time depends on the clock signal. The ISI encoding stage consists of the two LIF neurons

and an extractor and hence the power consumption reported includes the power consumed by two LIF neurons as well.

Component	Power Consumption
LIF neuron	**4.37µW**
Opamp	**64.62µW**
Current Mirror	**115µW**
ISI Encoding Stage	**13.37µW**

Table 2: Power Consumption for each component

3.4.3 Comparison with other state-of-the-art Neuromorphic Architectures

The following Table 3 summarizes the comparison of the designed SNN architectures with other state-of-the-art neuromorphic architectures. For the hardware testing and evaluation pixels from digits 0 to 9 were used to show the proof of concept. In terms of power consumption of only 2.9mW and an inference speed of 2µs/image, the designed SNN architecture shows competitive results compared to the state-of-the-art architectures.

	[109]	[110]	[111]	[112]	This Work
Technology	180nm	65nm	130nm	130nm	180nm
Algorithm	CNN	CNN	RBM	MLP	SNN
Memory Cell	SRAM	SRAM	ReRAM	ReRAM	ReRAM
Memory Mode	-	-	CIM	CIM	CIM
Weight Precision	-	-	1-bit	3-bit signed	1-bit
Neuron Type	IF	-	IF	-	LIF
Supply Voltage	1.8V	1V	1.8V	5V	1.8V
Latency	15.4ns	-	-	51.1ns	334ns
Power Consumption	4mW	278mW	2.2mW	-	2.9mW

Table 3: Comparison of Results with State-of-the-Art Implementations

3.4.4 Software Results

The designed SNN architecture is then tested for a large scale software model using MNIST. In the software model the structure is defined using the Figure 3-27 shown below. The software model developed from [113] includes an encoding stage where the inputs are initially encoded into either one of Poisson, TTFS or ISI. After the encoding stage the inputs are applied to the memristive crossbar layer that performs the matrix multiplication. Here the memristor weights are trained using ADAM optimizer and weights are quantized to 7 bits. After the crossbar layer the Batch Normalization layer is added which mimics the current mirror stage. In this stage column output currents are converted to voltage and it is clamped between two values. After this layer, a set of LIF neuron layers are added which then generates the spikes. The first set of LIF layer contains 100 neurons and the second output stage of LIF layer contains 10 neurons.

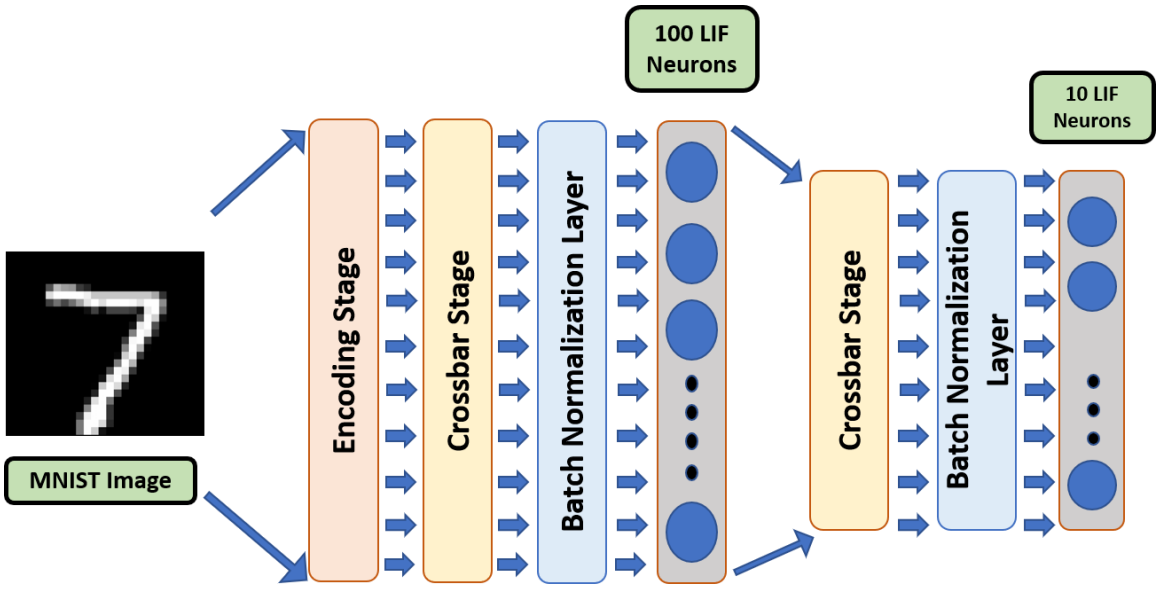

Figure 3-27: Software Architecture of the SNN

With the software model we evaluate our SNN architecture and compare with the different encoding schemes. Using the ISI encoding scheme the SNN architecture can achieve an 86% of accuracy and the results are summarized in the Table 4 below.

Encoding Scheme	Accuracy	Epochs
Poisson	84%	10
TTFS	85%	10
ISI	87%	10

Table 4: Accuracy Results with MNIST for Different Encoding Schemes

Chapter 4: Conclusion and Future Work

In recent years neuromorphic computing has received a lot of attention due to its ability to mimic the human brains. It has great benefits over traditional Von Neumann computing architectures since it can process data much faster using parallel processing and uses much less power. Neuromorphic computing architectures have also shown successful results in the field of machine learning and artificial intelligence.

In this thesis we initially study the two types of ANNs, mainly the RNNs which are more powerful in computing spatio-temporal tasks. We study an energy and power efficient version of RNNs that is Reservoir Computing which simplifies training process by only training the output layer and hence makes way for simpler hardware implementations. We study deep into Reservoir Computing to find out the different types of electronic reservoirs and its applications.

In the second part of this thesis, we cover SNNs which are the third generation of ANNs. SNNs provide significant advantage in terms of power and energy consumption by conveying information through spikes as well as keeping the temporal aspect of the information. To convert the information into spikes, a novel ISI encoding scheme is used which has higher information density compared to its TTFS counterpart and encodes the information in the time between the spikes. A memristive crossbar is used for in-memory computation to carry out the matrix multiplication process for feature extraction. The memristor model is the VT memristor that has a novel heat dissipation capability which reduces the resistance variation. An input preprocessing unit is designed which includes a precision current converter stage to apply the inputs to the crossbar. For the output stage a set of LIF neuron layer is used and the output data is determined by TTFS method. This network has a scalable structure and can be used to create a large scale neural network. Furthermore, it has a spatio-temporal information processing capability since it can take in an entire row of pixels over one clock cycle. This gives the SNN architecture a very high inference speed of 2μs/image. The whole network also has a power consumption of only 2.9mW. Software evaluation of this network in large scale with a 100 LIF neurons in the

input stage and 10 LIF neurons in the output stage reveals an 87% accuracy using the MNIST database with an ISI encoding scheme.

Future work with this SNN architecture involves testing the network with noisy images and exploring more complex datasets to see the accuracy. Training algorithms can be explored in the hardware using backpropagation circuits. Moreover, this architecture will be extended to a tiled one because for large amount of data processing in machine learning applications, it can perform matrix-matrix multiplication much faster with reduced energy. Different in-memory computations can be explored using this architecture such as the addition, truth tables and floating gate operations.

References

[1] C. Mead, "Neuromorphic electronic systems," in Proceedings of the IEEE, vol. 78, no. 10, pp. 1629-1636, Oct. 1990.

[2] S. Walczak and C. Narciso, “Artificial Neural Networks,” in Encyclopedia of Physical Science and Technology, 3rd ed, pp. 631-645, 2003.

[3] L. Appeltant et al., "Information processing using a single dynamical node as complex system", Nature communications, vol. 2, pp. 468, 2011.

[4] J. Schmiduber, “Jürgen Schmidhuber, “Deep learning in neural networks: An overview”, Neural Networks, vol. 61, pp. 85-117, 2015.

[5] H. Jaeger, “The ‘echo state’ approach to analysing and training recurrent neural networks,” German Nat. Res. Cntr. Inf. Technol., Sankt Augustin, Germany, 2001, GMD Report 148.

[6] W. Maass, T. Natschlager, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations,” Neural Comput., vol. 14, no. 11, pp. 2531–2560, Nov. 2002.

[7] Xiaowei Lin, Zehong Yang, Yixu Song, “Intelligent stock trading system based on improved technical analysis and Echo State Network”, Expert Systems with Applications, vol. 38, no 9, 2001, pp. 11347– 11354.

[8] M. D. Skowronski and J. G. Harris, "Automatic speech recognition using a predictive echo state network classifier", *Neural Netw.*, vol. 20, no. 3, pp. 414-423, Apr. 2007.

[9] B. Schrauwen, M. D’Haene, D. Verstraeten, and J. V. Campenhout, “Compact hardware liquid state machines on FPGA for real-time speech recognition,” Neural Networks, vol. 21, no. 2–3, pp. 511–523, 2008.

[10] J. Vreeken, “On real-world temporal pattern recognition using liquid state machines,” M.S. thesis, Utrecht University, Utrecht, Netherlands, 2004.

[11] S. Luo, H. Guan, X. Li, F. Xue and H. Zhou, "Improving liquid state machine in temporal pattern classification," *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, Singapore, 2018, pp. 88-91, doi: 10.1109/ICARCV.2018.8581122.

[12] Grigoryeva, L., J. Henriques, L. Larger, and J.-P. Ortega, "Optimal nonlinear information processing capacity in delay-based reservoir computers." Scientific reports, 2015.

[13] Ortín, S. and L. Pesquera, "Reservoir Computing with an Ensemble of Time-Delay Reservoirs." Cognitive Computation, 2017: p. 1-10.

[14] A. Beuter, J. Bélair, and C. Labrie, “Feedback and delays in neurological diseases: A modeling study using dynamical systems,” Bull. Math. Biol., vol. 55, no. 3, pp. 525–541, May 1993.

[15] K. E. Callan, L. Illing, Z. Gao, D. J. Gauthier, and E. Schöll, “Broadband chaos generated by an optoelectronic oscillator,” Phys. Rev. Lett., vol. 104, no. 11, p. 113901, Mar. 2010.

[16] L. Larger and J. M. Dudley, “Nonlinear dynamics: Optoelectronic chaos,” Nature, vol. 465, pp. 41–42, May 2010.

[17] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, “Photonic information processing beyond turing: An optoelectronic implementation of reservoir computing,” Opt. Exp., vol. 20, no. 3, pp. 3241–3249, Jan. 2012.

[18] D. Verstraeten, B. Schrauwen, M. D’Haene, and D. Stroobandt, “An experimental unification of reservoir computing methods,” Neural Netw., vol. 20, no. 3, pp. 391–403, Apr. 2007.

[19] K. Bai and Y. Yi, “DFR: An energy-efficient analog delay feedback reservoir computing system for brain-inspired computing,” ACM J. Emerg. Technol. Comput. Syst., vol. 14, no. 4, p. 45, 2018.

[20] K. Bai, Q. An, L. Liu and Y. Yi, "A Training-Efficient Hybrid-Structured Deep Neural Network With Reconfigurable Memristive Synapses," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 62-75, Jan. 2020.

[21] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi, "A survey of deep neural network architectures and their applications", *Neurocomput.*, vol. 234, pp. 11-26, 2017.

[22] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks", International journal of neural systems, pp. 295-308, 2009.

[23] W. Maass, “Networks of spiking neurons: the third generation of neural network models,” 1997.

[24] R. Van Rullen and S. J. Thorpe, "Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex", Neural Comput., vol. 13, no. 6, pp. 1255-1283, Jun. 2001.

[25] M. Davies et al., "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," in IEEE Micro, vol. 38, no. 1, pp. 82-99, January/February 2018.

[26] H. Cheng, W. Wen, C. Wu, S. Li, H. H. Li and Y. Chen, "Understanding the design of IBM neurosynaptic system and its tradeoffs: A user perspective," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, 2017, pp. 139-144.

[27] N. K. Upadhyay, H. Jiang, Z. Wang, S. Asapu, Q. Xia and J. J. Yang, "Emerging memory devices for neuromorphic computing", *Adv. Mater. Technol.*, vol. 4, no. 4, Apr. 2019.

[28] L. J. I. T. o. c. t. Chua, "Memristor-the missing circuit element," vol. 18, no. 5, pp. 507-519, 1971.

[29] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” Proc. Natl. Acad. Sci. USA, vol. 79, pp. 2554-2558, 1982.

[30] J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons,” in Proc. Natl. Acad. Sei. USA, vol. 81, pp. 2088-3092, 1984.

[31] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, pp. 855–868, May 2009.

[32] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling,” in Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH), 2014, pp. 338–342.

[33] Xiangang Li and Xihong Wu, “Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition,” in Acoustics,

Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on. IEEE, 2015, pp. 4520–4524.

[34] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735– 1780, 1997.

[35] A. Alalshekmubarak and L. S. Smith, “On improving the classification capability of reservoir computing for arabic speech recognition,” in Proc. Artif. Neural Netw. Mach. Learning, 2014, pp 225–232.

[36] A. Ghani, T. M. McGinnity, L. Maguire, L. McDaid, and A. Belatreche, “Neuro-inspired speech recognition based on reservoir computing,” in Advances in Speech Recognition, N. Shabtai, Ed. Rijeka, Croatia: InTech, 2010.

[37] X. Hinaut and P. Dominey, “On-line processing of grammatical structure using reservoir computing,” in Proc. Int. Conf. Artif. Neural Netw., 2012, pp. 596–603.

[38] Y. Jin and H. Yue, “Handwritten numeral recognition utilizing reservoir computing subject to optoelectronic feedback,” in Proc. Int. Conf. Natural Comput., 2015, pp. 1165–1169.

[39] D. Verstraeten, B. Schrauwen, and D. Stroobandt, “Reservoir computing with stochastic bitstream neurons,” in Proc. 16th Annual Prorisc Workshop, Veldhoven, The Netherlands, 2005, pp. 454–459.

[40] H. Jaeger, “Short Term Memory In Echo State Networks,” GMD-Forschungszentrum Informationstechnik 2001.

[41] F. wyffels, B. Schrauwen, and D. Stroobandt, "Using reservoir computing in a decomposition approach for time series prediction," in European Symposium on Time Series Prediction, 2007, pp. 149-158.

[42] A. Goudarzi, P. Banda, M. R. Lakin, C. Teuscher, and D. Stefanovic. (Jan. 2014). “A comparative study of reservoir computing for temporal signal processing.” [Online]. Available: https://arxiv.org/abs/1401.2224.

[43] R. Legenstein and W. Maass, “Edge of chaos and prediction of computational performance for neural circuit models,” Neural Netw., vol. 20, no. 3, pp. 323–334, 2007.

[44] M. Lukosevicius and H. Jaeger, “Reservoir computing approaches to recurrent neural network training,” Comput. Sci. Rev., vol. 3, no. 3, pp. 127–149, 2009.

[45] B. Hammer, B. Schrauwen, and J. J. Steil, “Recent advances in efficient learning of recurrent networks,” in Proc. Eur. Symp. Artif. Neural Netw., Bruges, Belgium, Apr. 2009, pp. 213–226.

[46] G. Tanaka et al., “Recent advances in physical reservoir computing: A review,” Neural Netw., vol. 115, pp. 100–123, 2019.

[47] H. Jaeger, “A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the ‘echo state network’ approach,” German Nat. Res. Cntr. Inf. Technol., Sankt Augustin, Germany, 2002, GMD Report 159.

[48] H. Jaeger and H. Haas, “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication,” Science, vol. 304, pp. 78–80, 2004.

[49] H. Jaeger, “Discovering multiscale dynamical features with hierarchical echo state networks,” School Eng. Sci., Jacobs Univ., Bremen, Germany, Tech. Rep. 10, Jul. 2007.

[50]] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, “Eventdriven random back-propagation: Enabling neuromorphic deep learning machines,” Frontiers Neurosci., vol. 11, p. 324, Jun. 2017.

[51] W. Maass, “Networks of spiking neurons: the third generation of neural network models,” Neural networks, vol. 10, no. 9, pp. 1659–1671, 1997.

[52] N. Soures and D. Kudithipudi, “Deep liquid state machines with neural plasticity for video activity recognition,” Front. Neurosci., vol. 13, pp. 686, 2019. doi: 10.3389/fnins.2019.00686.

[53] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," Journal of Computer and System Sciences, vol. 69, no. 4, pp. 593-616, 2004.

[54] W. Maass, “Liquid state machines: Motivation, theory, and applications,” in Computability in Context: Computation and Logic in the Real World, B. Cooper and A. Sorbi, Eds. Imperial College Press, 2010, pp. 275–296.

[55] H. Haken, Brain Dynamics, Synchronization, and aCtivity Patterns in Pulse-Coupled Neural Nets with Delays and Noise. New York: Springer, 2002.

[56]] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, “Information processing using a single dynamical node as complex system,” Nature Commun., vol. 2, art. no. 468, pp. 1–6, 2011.

[57] Y. Paquot et al., “Optoelectronic reservoir computing,” Sci. Rep., vol. 2, p. 287, Feb. 2012.

[58] R. Martinenghi, S. Rybalko, M. Jacquot, Y. K. Chembo, and L. Larger, “Photonic non-linear transient computing with multiple-delay wavelength dynamics,” Phys. Rev. Lett., vol. 108, no. 24, p. 244101, 2012.

[59] Soriano, M.C., S. Ortín, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, and G. Van der Sande, "Delay-based reservoir computing: noise effects in a combined analog and digital implementation." IEEE transactions on neural networks and learning systems, 2015. 26(2): p. 388-393.

[60] A. Namaj¯unas, K. Pyragas, and A. Tamaševiˇcius, “An electronic analog of the Mackey-Glass system,” Phys. Lett. A, vol. 201, no. 1, pp. 42–46, 1995.

[61] M. C. Mackey and L. Glass, “Oscillation and chaos in physiological control systems,” Science, vol. 197, no. 4300, pp. 287–289, 1977.

[62] J. Li, K. Bai, L. Liu, and Y. Yi, “A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system,” in Proceedings of 19th International Symposium In Quality Electronic Design (ISQED), 2018.

[63] C. Zhao et al., “Energy efficient spiking temporal encoder design for neuromorphic computing systems,” IEEE Trans. Multi-Scale Comput. Syst., vol. 2, no. 4, pp. 256–276, Sep. 2016.

[64] C. Zhao, J. Li, L. Liu, L. S. Koutha, J. Liu, and Y. Yi, “Novel spike based reservoir node design with high performance spike delay loop,” in Proc. 3rd ACM Int. Conf. Nanoscale Comput. Commun., Sep. 2016, pp. 1–5.

[65] C. Zhao, Y. Yi, J. Li, X. Fu, and L. Liu, “Interspike-interval-based analog spike-time-dependent encoder for neuromorphic processors,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 8, pp. 2193–2205, Aug. 2017.

[66] Miquel L. Alomar, Vincent Canals, Nicolas Perez-Mora, Víctor Martínez-Moll, and J. L. Rosselló “FPGA-based Stochastic Echo State Networks for Time-Series Forecasting,” Comput. Intell. Neurosci., vol. ND, no. ND, p. Article ID 537267, 2015.

[67] B. Penkovsky, L. Larger, and D. Brunner, “Efficient design of hardware-enabled reservoir computing in FPGAs,” J. Appl. Phys. 124, 162101, 2018.

[68] Yi, Y., Liao, Y., Wang, B., et al.: ‘FPGA based spike-time dependent encoder and reservoir design in neuromorphic computing processors’, Microprocess. Microsyst., 2016, 46, Part B, pp. 175–183.

[69] P. Petre and J. Cruz-Albrecht, “Neuromorphic mixed-signal circuitry for asynchronous pulse processing,” in Rebooting Computing (ICRC), IEEE International Conference on. IEEE, 2016, pp. 1–4.

[70] S. Roy, A. Banerjee, and A. Basu, “Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations,” IEEE Trans. Biomed. Circuits Syst., vol. 8, no. 5, pp. 681–695, Oct. 2014.

[71] A. Polepalli and D. Kudithipudi, “Reconfigurable digital design of a liquid state machine for spatio-temporal data,” presented at the Third ACM Int. Conf. Nanoscale Computing and Communication, 2016.

[72] Y. Zhang, P. Li, Y. Jin, and Y. Choe. A digital liquid state machine with biologically inspired learning and its application to speech recognition IEEE Trans. on Neural Networks and Learning Systems 2015.

[73] Q. Xia and J. J. Yang, “Memristive crossbar arrays for brain-inspired computing,” Nature Mater., vol. 18, no. 4, pp. 309–323, 2019.

[74] V. Keshmiri, “A study of the memristor models and applications,” Linköping Univ., Linköping, Sweden, Tech. Rep. LITH-EX-11/4455, 2014.

[75] T. Chang, Y. Yang, and W. Lu, “Building neuromorphic circuits with memristive devices,” IEEE Circ. Syst. Mag., vol. 13, no. 2, pp. 56–73, 2013.

[76] H. An, M. S. Al-Mamun, M. K. Orlowski, L. Liu and Y. Yi, "Robust Deep Reservoir Computing through Reliable Memristor with Improved Heat Dissipation Capability," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2020.3002539.

[77] C. E. A. Du, “Reservoir computing using dynamic memristors for temporal information processing,” Nature Commun., vol. 8, 2017, Art. no. 2204.

[78]] G. Van der Sande, D. Brunner, and M. C. Soriano, “Advances in photonic reservoir computing,” Nanophotonics, vol. 6, no. 3, pp. 561–576, 2017.

[79] D. Brunner, Miguel C. Soriano, and G. Van der Sande, “Photonic reservoir Computing,” 1st ed., De Gruyter, 08.07.19.

[80] K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman, and J. V. Campenhout, “Toward optical signal processing using photonic reservoir computing,” Opt. Exp., vol. 16, no. 15, pp. 11 182– 11 192, Jul. 2008.

[81] K. Vandoorne et al., “Experimental demonstration of reservoir computing on a silicon photonics chip,” Nature Commun., vol. 5, 2014, Art. no. 3541.

[82] H. Hauser, A. Ijspeert, R. Fuechslin, R. Pfeifer, and W. Maass, “Towards a theoretical foundation for morphological computation with compliant bodies,” Biological Cybernetics, vol. 105, pp. 355–370, 2011.

[83] J. Torrejon et al., “Neuromorphic computing with nanoscale spintronic oscillators,” Nature, vol. 547, pp. 428–431, Jul. 2017.

[84] R. Nakane, G. Tanaka, and A. Hirose, ‘‘Reservoir computing with spin waves excited in a garnet film,’’ IEEE Access, vol. 6, pp. 4462–4469, Jan. 2018.

[85] D. Prychynenko et al., “Magnetic skyrmion as a nonlinear resistive element: A potential building block for reservoir computing,” Phys. Rev. A, Gen. Phys., vol. 9, no. 1, 2018, Art. no. 014034, doi: 10.1103/ PhysRevApplied.9.014034.

[86] A. Jalalvand, G. Van Wallendael and R. Van De Walle, "Real-Time Reservoir Computing Network-Based Systems for Detection Tasks on Visual Contents," 2015 7th International Conference on Computational Intelligence, Communication Systems and Networks, Riga, 2015, pp. 146-151, doi: 10.1109/CICSyN.2015.35.

[87] K. Bai, Y. Yi, Z. Zhou, S. Jere and L. Liu, "Moving Toward Intelligence: Detecting Symbols on 5G Systems Through Deep Echo State Network," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 10, no. 2, pp. 253-263, June 2020, doi: 10.1109/JETCAS.2020.2992238.

[88] S. S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng and Y. Yi, "Brain-Inspired Wireless Communications: Where Reservoir Computing Meets MIMO-OFDM," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 10, pp. 4694-4708, Oct. 2018, doi: 10.1109/TNNLS.2017.2766162.

[89] Z. Zhou, L. Liu, V. Chandrasekhar, J. Zhang, and Y. Yi, “Deep Reservoir Computing Meets 5G MIMO-OFDM Systems in Symbol Detection,” in 34th AAAI Conf. Artificial Intell., 2020.

[90] H. Chang, H. Song, Y. Yi, J. Zhang, H. He and L. Liu, "Distributive Dynamic Spectrum Access Through Deep Reinforcement Learning: A Reservoir Computing-Based Approach," in IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1938-1948, April 2019, doi: 10.1109/JIOT.2018.2872441.

[91] K. Hamedani, L. Liu, R. Atat, J. Wu and Y. Yi, "Reservoir Computing Meets Smart Grids: Attack Detection Using Delayed Feedback Networks," in IEEE Transactions on Industrial Informatics, vol. 14, no. 2, pp. 734-743, Feb. 2018, doi: 10.1109/TII.2017.2769106.

[92] T. Yamazaki and S. Tanaka, “The cerebellum as a liquid state machine,” Neural Networks, vol. 20, no. 3, pp. 290–297, 2007.

[93] E. A. Antonelo, et al., "Unsupervised Learning in Reservoir Computing: Modeling Hippocampal Place Cells for Small Mobile Robots," presented at the International Conference on Artificial Neural Networks (ICANN), 2009.

[94] F. Hadaeghi, “Reservoir computing models for patientadaptable ECG monitoring in wearable devices,” arXiv preprint arXiv:1907.09504, 2019.

[95] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," nature, vol. 453, no. 7191, pp. 80-83, 2008.

[96] S. R. Williams, "How we found the missing memristor," Spectrum, IEEE, vol. 45, no. 12, pp. 28-35, 2008.

[97] V. Keshmiri, "A Study of the Memristor Models and Applications," 2014.

[98] Y. Xie and J. Zhao, "Emerging Memory Technologies," in IEEE Micro, vol. 39, no. 1, pp. 6-7, Jan.-Feb. 2019, doi: 10.1109/MM.2019.2892165.

[99] J. Park, “Neuromorphic Computing Using Emerging Synaptic Devices: A Retrospective Summary and an Outlook,” *Electronics*, vol. 9, no. 9, p. 1414, Sep. 2020.

[100] N. K. Upadhyay, H. Jiang, Z. Wang, S. Asapu, Q. Xia and J. J. Yang, "Emerging memory devices for neuromorphic computing", *Adv. Mater. Technol.*, vol. 4, no. 4, Apr. 2019.

[101] D. Niu, Yang Xiao and Yuan Xie, "Low power memristor-based ReRAM design with Error Correcting Code," *17th Asia and South Pacific Design Automation Conference*, 2012, pp. 79-84, doi: 10.1109/ASPDAC.2012.6165062.

[102] H.-S. P. Wong et al., "Metal-oxide RRAM", *Proc. IEEE*, vol. 100, no. 6, pp. 1951-1970, Jun. 2012.

[103] A. Huang et al., "Memristor neural network design", *Memristor and Memristive Neural Networks*, pp. 1-35, 2018.

[104] M. Shevgoor, N. Muralimanohar, R. Balasubramonian and Y. Jeon, "Improving memristor memory with sneak current sharing," *2015 33rd IEEE International Conference on Computer Design (ICCD)*, 2015, pp. 549-556, doi: 10.1109/ICCD.2015.7357164.

[105] Liu B. Neuromorphic System Design and Application. University of Pittsburgh; ProQuest Dissertations Publishing, 2016.

[106] B. Govoreanu et al., "10× 10nm 2 Hf/HfO x crossbar resistive RAM with excellent performance, reliability and low-energy operation," in Electron Devices Meeting (IEDM), 2011 IEEE International, 2011: IEEE, pp. 31.6. 1-31.6. 4.

[107] M. Al-Mamun, S. W. King, S. Meda, and M. K. Orlowski, "Impact of the Heat Conductivity of the Inert Electrode on ReRAM Performance and Endurance," ECS Transactions, vol. 85, no. 8, pp. 207-212, 2018.

[108] H. An, M. S. Al-Mamun, M. K. Orlowski, L. Liu and Y. Yi, "Robust Deep Reservoir Computing through Reliable Memristor with Improved Heat Dissipation Capability," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2020.3002539.

[109]G. Indiveri, F. Corradi, and N. Qiao, “Neuromorphic architectures for spiking deep neural networks,” in 2015 IEEE International Electron Devices Meeting (IEDM). IEEE, 2015, pp. 4–2.

[110]Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” IEEE journal of solid-state circuits, vol. 52, no. 1, pp. 127–138, 2016.

[111]W. Wan, R. Kubendran, S. B. Eryilmaz, W. Zhang, Y. Liao, D. Wu, S. Deiss, B. Gao, P. Raina, S. Joshi et al., “33.1 a 74 tmacs/w cmos-rram neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models,” in 2020 IEEE International Solid-State Circuits Conference-(ISSCC). IEEE, 2020, pp. 498–500.

[112]Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen et al., “33.2 a fully integrated analog reram based 78.4 tops/w compute-in-memory chip with fully parallel mac computing,” in 2020 IEEE International Solid-State Circuits Conference- (ISSCC). IEEE, 2020, pp. 500–502.

[113]Q. Duan, Z. Jing, X. Zou, Y. Wang, K. Yang, T. Zhang, S. Wu, R. Huang, and Y. Yang, “Spiking neurons with spatiotemporal dynamics and gain modulation for monolithically integrated memristive neural networks,” Nature News, 07-Jul-2020. [Online]. Available: https://www.nature.com/articles/s41467-020-17215-3.