ENERGY EFFICIENT AND ERROR RESILIENT NEUROMORPHIC COMPUTING IN VLSI

A Dissertation

by

YONGTAE KIM

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2013

Major Subject: Electrical Engineering

ABSTRACT

Realization of the conventional Von Neumann architecture faces increasing challenges due to growing process variations, device reliability and power consumption. As an appealing architectural solution, brain-inspired neuromorphic computing has drawn a great deal of research interest due to its potential improved scalability and power efficiency, and better suitability in processing complex tasks. Moreover, inherit error resilience in neuromorphic computing allows remarkable power and energy savings by exploiting approximate computing. This dissertation focuses on a scalable and energy efficient neurocomputing architecture which leverages emerging memristor nanodevices and a novel approximate arithmetic for cognitive computing.

First, a brain-inspired digital neuromorphic processor (DNP) architecture with memristive synaptic crossbar is presented for large scale spiking neural networks. We leverage memristor nanodevices to build an $N \times N$ crossbar array to store not only multibit synaptic weight values but also the network configuration data with significantly reduced area cost. Additionally, the crossbar array is accessible both column- and row-wise to significantly expedite the synaptic weight update process for on-chip learning. The proposed digital pulse width modulator (PWM) readily creates a binary pulse with various durations to read and write the multilevel memristors with low cost. Our design integrates N digital leaky integrate-and-fire (LIF) silicon neurons to mimic their biological counterparts and the respective on-chip learning circuits for implementing spike timing dependent plasticity (STDP) learning rules. The proposed column based analog-to-digital conversion (ADC) scheme accumulates the pre-synaptic weights of a neuron efficiently and reduces silicon area by using only one shared arithmetic unit for processing LIF operations of all N neurons. With 256 silicon neurons, the learning circuits and $64K$ synapses, the power dissipation and area of our design are evaluated as 6.45 mW and 1.86 mm^2, respectively, in a 90 nm CMOS technology.

Furthermore, arithmetic computations contribute significantly to the overall processing time and power of the proposed architecture. In particular, addition and comparison operations represent 88.5% and 42.9% of processing time and power for digital LIF computation, respectively. Hence, by exploiting the built-in resilience of the presented neuromorphic architecture, we propose novel approximate adder and comparator designs to significantly reduce energy consumption with a very low error rate. The significantly improved error rate and critical path delay stem from a novel carry prediction technique that leverages the information from less significant input bits in a parallel manner. An error magnitude reduction scheme is proposed to further reduce amount of error once detected with low cost in the proposed adder design. Implemented in a commercial 90 nm CMOS process, it is shown that the proposed adder is up to 2.4× faster and 43% more energy efficient over traditional adders while having an error rate of only 0.18%. Additionally, the proposed comparator achieves an error rate of less than 0.1% and an energy reduction of up to 4.9× compared to the conventional ones. The proposed arithmetic has been adopted in a VLSI-based neuromorphic character recognition chip using unsupervised learning. The approximation errors of the proposed arithmetic units have been shown to have negligible impacts on the training process. Moreover, the energy saving of up to 66.5% over traditional arithmetic units is achieved for the neuromorphic chip with scaled supply levels.

DEDICATION

To my wife

ACKNOWLEDGEMENTS

First and foremost, I am very grateful to have had the opportunity to work with a great research advisor Dr. Peng Li and would like to thank him with my deep respect for his valuable advice and consistent support during my doctoral studies at Texas A&M University. Dr. Li has actively encouraged me to move forward with new innovative research ideas and willingly shared his profound knowledge, deep insight and creative inspiration so I could learn the way of research from him. Also, I would like to thank my committee members Dr. Gwan Choi, Dr. Jose Silva-Martinez and Dr. Rabi Mahapatra for their constructive discussions and suggestions on my research, making this dissertation possible.

My appreciation goes to all the members in our research group for their knowledge, discussion and friendship. Particular thanks go to Yong Zhang and Qian Wang for the simulation and layout supports. Many friends in the department and the alumni of Korea University have made my stay of four years in College Station a pleasurable and unforgettable experience. I also want to acknowledge all my other friends who have consistently helped me at A&M for their considerable assistances.

From deep down in my heart, I would like to thank my parents and other family members for their devotion, support and encouragement. In particular, I would like to give special thanks to my wife for her unconditional love, trust, patience and sacrifice, leading me to successfully complete my Ph.D. studies.

Finally, the funding support from the Semiconductor Research Corporation is acknowledged.

NOMENCLATURE

ADC	Analog-to-Digital Conversion/Converter
ANN	Artificial Neural Network
CMOS	Complementary Metal Oxide Semiconductor
CLA/CLC	Carry Lookahead Adder/Comparator
DNP	Digital Neuromorphic Processor
EDAP	Energy-Delay-Area Product
EDC	Error Detection and Correction
EDEP	Energy-Delay-Error Product
EDP	Energy-Delay Product
FSM	Finite State Machine
LIF	Leaky Integrate-and-Fire
LSB	Least Significant Bit
LUT	Look Up Table
MSB	Most Significant Bit
PWM	Pulse Width Modulation/Modulator
RCA/RCC	Ripple Carry Adder/Comparator
SNN	Spiking Neural Network
STDP	Spike Timing Dependent Plasticity
VCO	Voltage Controlled Oscillator
VLSI	Very Large Scale Integration
WTA	Winner-Take-All

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The human brain mediates and produces our thoughts, actions, memory, feelings and other complex tasks. All these, however, are accomplished with great energy and space efficiency by the brain. In contrast, to achieve the same, the man-made conventional Von Neumann machines may require tremendous power, energy and space resources for computation, communication and memory if it is all possible [46]. To date, implementing the Von Neumann architecture faces grand challenges due to growing process variations, device reliability and power consumption. As an appealing architectural solution, brain-inspired neuromorphic computing has emerged as a promising solution to overcome these underlying constraints. It may be well-suited for processing complex tasks, such as character or image recognition, classification and language learning and enjoy greater power efficiency and scalability [48, 58, 46, 59, 4, 43].

Furthermore, brain-inspired architectures may offer inherit error resilience and fault tolerance, which is very appealing for large-scale integration in scaled VLSI technologies. This inherit error resilience in neuromorphic computing allows remarkable power and energy savings by adopting approximate computing, which has drawn a significant research amount of interest in order to remedy the increasing energy efficiency challenges [21, 60, 8]. The key observation of approximate computing is that many applications, such as digital signal processing (DSP) and neuromorphic systems, have inherent error resilience, and hence 100% precision in computation is not required. This provides opportunities for energy saving by relaxing computation accuracy while achieving an acceptable processing quality. Particularly, the core of many DSP and neuromorphic applications lies in processing specific kernel functions, which occupy a significant portion of silicon area and computation time [49, 26]. For instance, MPEG motion estimation heavily performs the L1-norm arithmetic for sum of absolute difference (SAD) calculation [66] and spiking neural networks use the leaky integrate-and-fire (LIF) operation to mimic neuron behavior [58]. Obviously, adders are the primary component for building these arithmetic kernel functions. In addition, comparators are indispensable to determine firing activities in the LIF operation of neuromorphic applications.

To this end, this dissertation proposes two hardware design techniques for scalable and energy efficient neurocomputing applications: 1) reconfigurable digital neuromorphic processor (DNP) with memristive synaptic crossbar and 2) energy efficient approximate arithmetic for neuromorphic VLSI systems.

1.1 Digital Neuromorphic Processor for Cognitive Computing

The first contribution of this dissertation includes a reconfigurable neuromorphic processor with memristive synaptic crossbar for cognitive computing. We propose a reconfigurable digital neuromorphic architecture comprising a memristive crossbar, an array of digital LIF spiking neurons and on-line learning circuits that support spike timing dependent plasticity (STDP) learning mechanisms. We leverage the memristor nanodevice to implement on-chip synaptic weight storage with low-cost since the memristor provides non-volatility, excellent scalability and high density of 10 Gb/cm^2 or more [23, 72]. To implement a multilevel memristor synaptic memory, we systematically analyze the memristor device in terms of programming time and level partitioning. We also investigate memristor readout schemes to more efficiently perform digital LIF operations with the crossbar structure and present a low-cost digital pulse width modulation (PWM) scheme for writing the memristor. While the previous analog-to-digital converter (ADC) design in [36] has a bottleneck of overall power dissipation, we address this limitation by optimizing the VCO based column ADC through the introduction of an asynchronous counter to measure the VCO frequency in digital form. In the proposed DNP, the $N \times N$ memristive synaptic array, which stores both multibit synapse values and network configuration data, can be accessed both column- and row-wise to speed up the synaptic weight update process. The proposed column ADC effectively accumulates pre-synaptic weights and allows a single adder and comparator to be shared among all N neurons to perform LIF operations without degrading throughput. This leads to a considerable silicon area reduction and its digital implementation style is scalable for large scale integration.

When implemented in a commercial 90 nm CMOS technology, a 256 neuron design with a 256×256 synaptic array based on the proposed neuromorphic architecture has an estimated area of 1.86 mm^2 and power consumption of 6.45 mW under the regular supply voltage of 1.2 V, respectively. The proposed neuromorphic architecture is rather flexible and can be configured to various network topologies to support to a range of cognitive learning applications. To demonstrate its potential application, we configure our DNP to realize a two-layer spiking network with over two hundreds silicon neurons for character recognition with unsupervised learning.

1.2 Approximate Arithmetic for Energy Efficient Neurocomputing

Our second contribution is to apply an approximate computing scheme to the neuromorphic hardware design for considerable energy saving (see Figure 1.1). To achieve this, we propose a novel approximate adder with a parallel carry-skip scheme. While reducing the worst-case carry propagation delay, this carry-skip scheme allows for highly accurate carry prediction, making it possible to either speed up addition operations or reduce energy dissipation by lowering the supply voltage. The signif-

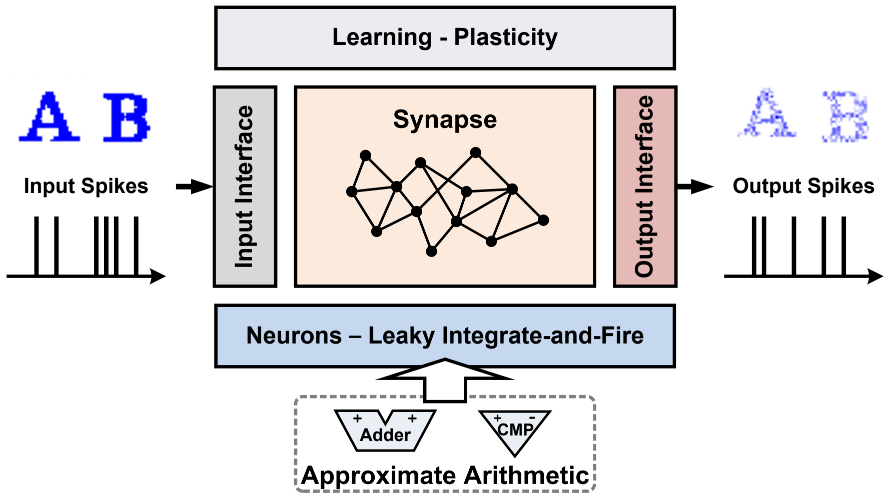


Figure 1.1: Application of approximate arithmetic in neuromorphic computing.

icantly improved error rate and critical path delay stem from the employed carry prediction technique that leverages the information from less significant input bits in a parallel manner. An error magnitude reduction scheme is proposed to further reduce amount of error once detected with low cost. Our adder design is rather flexible in the sense that a low-overhead error correction logic can be readily included to achieve error-free operations at the cost of one additional clock cycle. Additionally, we extend our approximate arithmetic scheme to comparator design and present a complete error rate analysis for the proposed arithmetic units. We extensively compare our approximates designs with a large number of existing accurate and approximate adders and comparators and show the large improvements in area, power, energy, timing and error rate brought by our design technique. To evaluate the performance of our approximate arithmetic units for neurocomputing applications, we present an efficient evaluation methodology to analyze large VLSI-based spiking neural networks with over a thousand silicon neurons for character recognition. We extensively study the error tolerance of the network, the overall energy consumption of digital LIF neurons during the learning process and its dependency on the underlying arithmetic units.

Implemented in a commercial 90 *nm* CMOS process, it is shown that the proposed adder is up to 2.4× faster and 43% more energy efficient over traditional adders while having an error rate of only 0.18%. In addition, the proposed comparator achieves an error rate of less than 0.1% and an energy reduction of up to 4.9× compared to the conventional ones. To evaluate the performance of the proposed arithmetics under neuromorphic applications, we develop a behavioral evaluation approach for a VLSI-based neuromorphic character recognition chip using unsupervised learning. The approximation errors of the proposed adder and comparator have been shown to have negligible impact on the training process while other approximate adders lead to unacceptable level of performance degradation. Furthermore, the proposed adder and comparator enable the overall energy reductions of up to 66.5% over traditional arithmetic units for the digital neuron circuits during the training process with the scaled supply.

1.3 Outline of the Dissertation

The remainder of this dissertation is organized as the follows. Section 2 describes the background of brain-inspired neuromorphic and approximate computing, and the related works in the literature. The memristor nanodevice leveraged for synaptic array is also introduced in Section 2. The reconfigurable digital neuromorphic processor with memristive synaptic crossbar and its application for the character recognition system with unsupervised learning are presented in Section 3. After proposing energy efficient approximate arithmetic units in Section 4, the impacts of the approximation errors on the neurocomputing application and the energy efficiency analysis of the proposed approximate units for the neuromorphic hardware design are presented in Section 5. Finally, we conclude this dissertation and discuss the future works in Section 6.

2. BACKGROUND AND RELATED WORKS

This section describes an overview of neuromorphic and approximate computing paradigms. It begins with the biological motivation of neuromorphic computing, then gives reviews of artificial and spiking neural networks and their learning algorithms. It also deals with the existing designs of silicon neurons and neuromorphic VLSI systems to mimic the biological brain on silicon, and discusses their key design issues and limitations. Next, it introduces a notion of approximate computing and several design approaches of the approximate adder, which is the primary component in approximate computing, are briefly reviewed and their advantages and disadvantages are presented as well. The overview of the memristor nanodevice, which is employed in our digital neuromorphic processor as on-chip storage to maintain a huge number of synaptic weight values, is given. Finally, it clarifies the objective of this dissertation.

2.1 Brain-Inspired Neuromorphic Computing

Today's Von Neumann computers are able to not only deal with very complicated numerical and algorithmic computations and procedural control tasks, such as sorting, but also store huge amount of data. Thus, they have been widely used for solving these complicated problems steadily which may be hard to be handled by humans. On the other hand, traditional machines may be limited by many other kinds of tasks that human beings can process without difficulties, such as character or image recognition, text reading and language learning. Importantly, the humans adapt to new situations and accumulate information and knowledge by an amazing ability of the brain, learning. In other words, when faced with a new situation, they make a proper decision and perform an appropriate behavior based on the acquired knowledge through the learning or training processes. Incredibly, the human brain processes these tasks much more energy efficiently than the conventional computers. In general, biological neurons are $10^6\times$ slower than silicon logic gates [20]. Silicon chips operate with a clock period in the range of the nanoseconds (10^{-9} *sec.*) while neural events happen in the millisecond (10^{-3} *sec.*) range. The slower operating speed of the biological neurons may have contributed to the brain's results in exceptionally good energy efficiency. Specifically, the brain consumes approximately $10^{-16}J$ per operation per second, whereas the traditional computer requires an energy level of about $10^{-6}J$ per operation per second [20].

Historically, from the past years, neuroscientists have devoted intense efforts towards investigating the human brain. As part of these efforts, a landmark work in modeling the dynamics of a biological neuron was conducted by Hodgkin and Huxley [24]. After that, a variety of computational neuron models, such as FitzHugh-Nagumo [17, 52], Hindmarsh-Rose [22], and Morris-Lecar [51], have been proposed. Also, scientists have studied the interactions among neurons through synapses. The brain and neuron modeling are greatly facilitated by the rapid advance of digital computers. However, simulating a large number of these computational neurons is still challenging to date because it requires tremendous computing power and simulation time. Meanwhile, neuromorphic engineers have been trying to reproduce the neuron behaviors by morphing their anatomy and physiology into silicon chips for simulating the human or mammal's brains in real-time [48, 58, 46, 59, 4, 43]. The hardware implementation provides very fast simulation of neural networks with less power consumption.

2.1.1 Biological Motivation

Artificial or spiking neural networks are the core of brain-inspired neuromorphic computing systems. Their development has been motivated in the part by the insights obtained from biological nervous systems (*e.g.* the human brain) which are an extremely intricate interconnection of neurons. As an example, the adult human brain is estimated to contain a densely interconnected network of approximately 10^{11} neurons and more than 10^{14} synapses [13]. Neurons are the primary elements of a nervous system and are specialized types of biological cells that are electrically excitable. Neurons process and transmit information in the form of cellular signals which are either electrical or chemical for long and short distances, respectively. A considerable number of neurons connect to each other to form a neural network via synapses, which are specialized connections among the neurons.

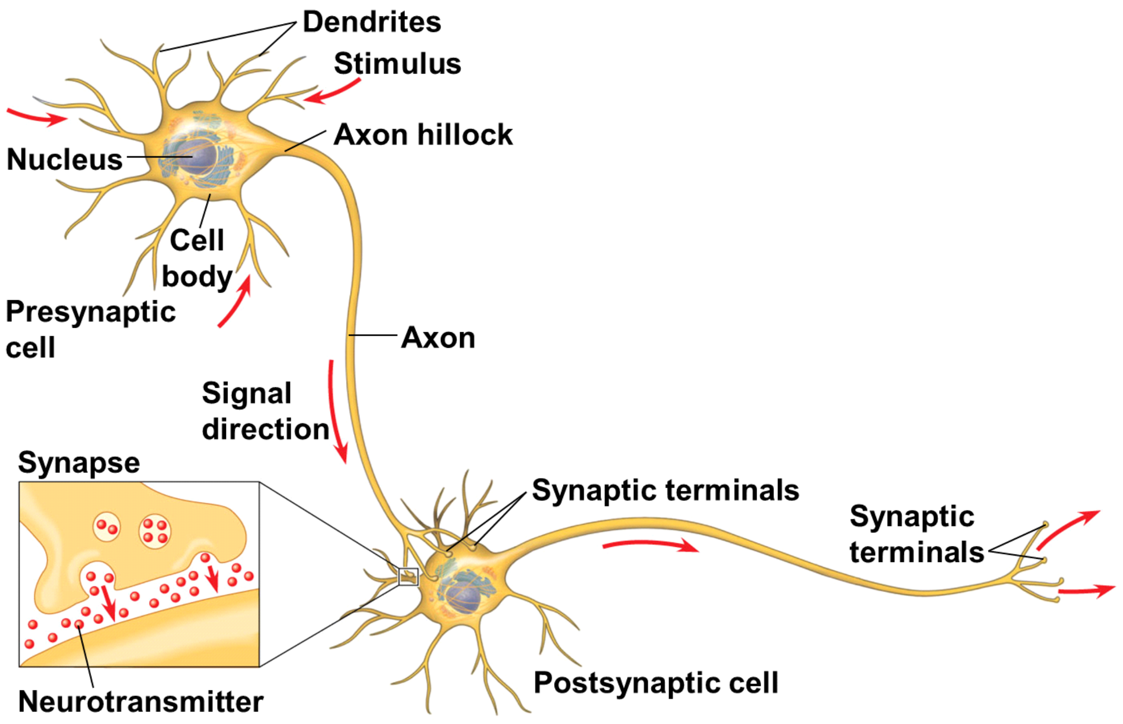


Figure 2.1: Biological neuron anatomy [57].

Figure 2.1 illustrates a typical biological neuron and synapse structure. A neuron mainly consists of three functionally distinct parts, which are the cell body (often called the soma), axon and dendrites. The cell body is the heart of the neuron and includes the nucleus where most protein synthesis occurs. The dendrites of the neuron are highly branched extensions and receive nerve signals from other neurons. A neuron may have numerous dendrites and their overall shape is referred to as a dendritic tree. On the other hand, the neuron has only one axon that is typically thinner and much longer than the dendrites and transmits the signals to other cells via synapses. In short, the dendrites and axon act as the signal receiver and transmitter, respectively. Information of the nervous system is encoded in the form of an electrical impulse which is called the action potential or spike and the pulse is transmitted from a pre-synaptic neuron to a post-synaptic one. The action potentials are created by the axon hillock that is a specialized part of the cell body and connects to the axon. A neuron processes information by integrating the incoming nerve signals that come from its pre-synaptic neurons and the action potential is generated when the membrane potential of the neuron reaches a certain threshold. Briefly, the neuron transmits the information using the action potentials or spikes.

A synapse is basically a junction between two neurons, which are referred to as the pre-synaptic and post-synaptic neurons, respectively. In fact, neurons do not physically touch each other and are separated by a small space called the synaptic cleft. When an action potential arrives at the axon terminal, the pre-synaptic neuron releases chemical neurotransmitter molecules into the synaptic cleft and they diffuse across the synaptic junction, leading to interneuron communication at the synapse. These chemical molecules bind to the receptor which is placed on the opposite side of the cleft (*i.e.* post-synaptic neuron) and cause the membrane potential of the post-synaptic neuron to change. The type of the receptor and neurotransmitter employed at the synapse determines whether the post-synaptic neuron would be either excited or inhibited when a pre-synaptic spike is generated. The resulting effects of excitation and inhibition are to potentiate and depress the post-synaptic neuron's membrane potential. In addition, the strength of a synapse is defined by the amplitude change of the membrane potential as a result of a pre-synaptic action potential. Learning and memory are resulted from the changes in synaptic strength through the mechanism of synaptic plasticity that leads to either decrease or increase in the strength. In this way, the synapses store information.

2.1.2 Artificial Neural Networks

Artificial neural network (ANN) is a computational model inspired by the biological nervous systems, in particular the brain, and is widely adopted in applications of intelligent information processing, such as machine learning and pattern recognition [20, 30]. An ANN is simply an intricated web of connected artificial neurons (called the processing elements) that processes information in a way to mimic biological neural networks. The signals of the network are passed among the artificial neurons over the connection links called synapses. Each synapse has an associated weight or strength of its own, which typically multiplies the signal transmitted. The weight is a adaptive numerical parameter that can be manipulated by a learning algorithm. Additionally, each neuron accumulates the input signals that are weighted by the respective synapses of the neuron, and applies an activation function that may be either linear or non-linear to its net input (*i.e.* sum of the weighted input signals) to determine its output signal. Furthermore, ANNs are similar to their biological counterparts in the sense that they perform functions dispersively, collectively and in parallel by the processing elements.

Artificial neurons are the basic functional units to build an ANN and are a great simplification of biological neurons. The first computation model of artificial neurons was created by McCulloch and Pitts in [44]. The McCulloch-Pitts model is based on a simplified binary neuron whose state is either active or not-active, and implements a threshold function in discrete time. The state is determined by accumulating weighted incoming signals of activated pre-synaptic neurons at each neural computation step. Namely, it is set to active if the sum of the weighted signals exceeds a given threshold, otherwise it is not. Subsequent neuron models extend the McCulloch-Pitts model by introducing real-valued inputs and outputs and various threshold (activation) functions [20]. Figure 2.2 depicts a typical computation

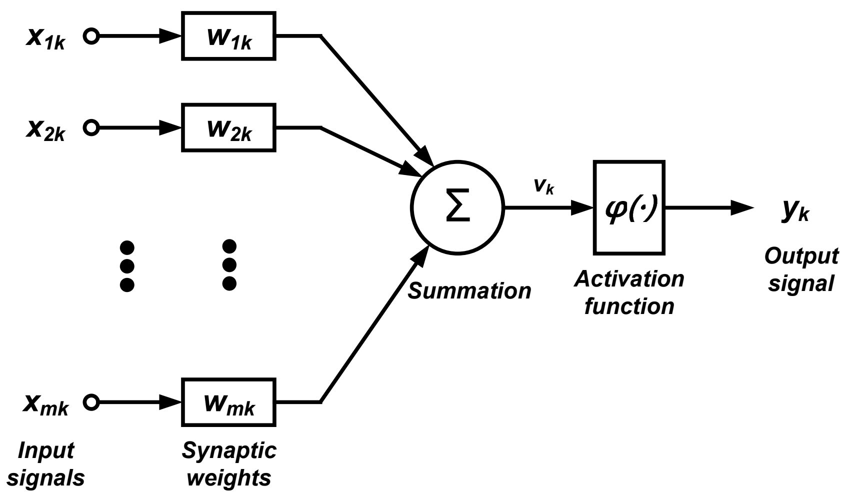


Figure 2.2: Artificial neuron.

model of the artificial neuron. The model consists of three basic elements: 1) a set of synapses which are represented by synaptic weights; 2) an adder for summing the input signals that are multiplied by the respective synaptic weights; and 3) an activation function to bound the amplitude of the output signal. The behavior of the neuron k is mathematically described by the following equations:

$$v_k = \sum_{j=0}^{m} w_{jk} x_j \tag{2.1}$$

$$y_k = \varphi(v_k) \tag{2.2}$$

where m is the number of pre-synaptic neurons, x_j is the input signal coming from the neuron j, w_{jk} is the synaptic weight between the neuron j and k, v_k is the linear summation due to the input signals, $\varphi(\cdot)$ is the activation function and y_k is the output signal of the neuron k. The activation function denoted by $\varphi(\cdot)$ is play a role of defining the neuron output in terms of the summation input v. There are many possible activation functions but we introduce three basic types of functions: 1) step; 2) piecewise linear; and 3) sigmoid. They are plotted in Figure 2.3, respectively.

First, the step function makes a binary decision and produces only two values. The step function in Figure 2.3(a) can be described by

$$\varphi(v) = \begin{cases} 1 & \textit{if } v \geq 0 \\ 0 & \textit{if } v < 0 \end{cases} \tag{2.3}$$

where the threshold value is zero. The output value is "0" if the input v is greater than or equals to a given threshold, otherwise this function generates a value of "1" as the output.

Second, the piecewise linear function is composed of a number of linear segments over an equal number of intervals. The piecewise linear function described in Figure

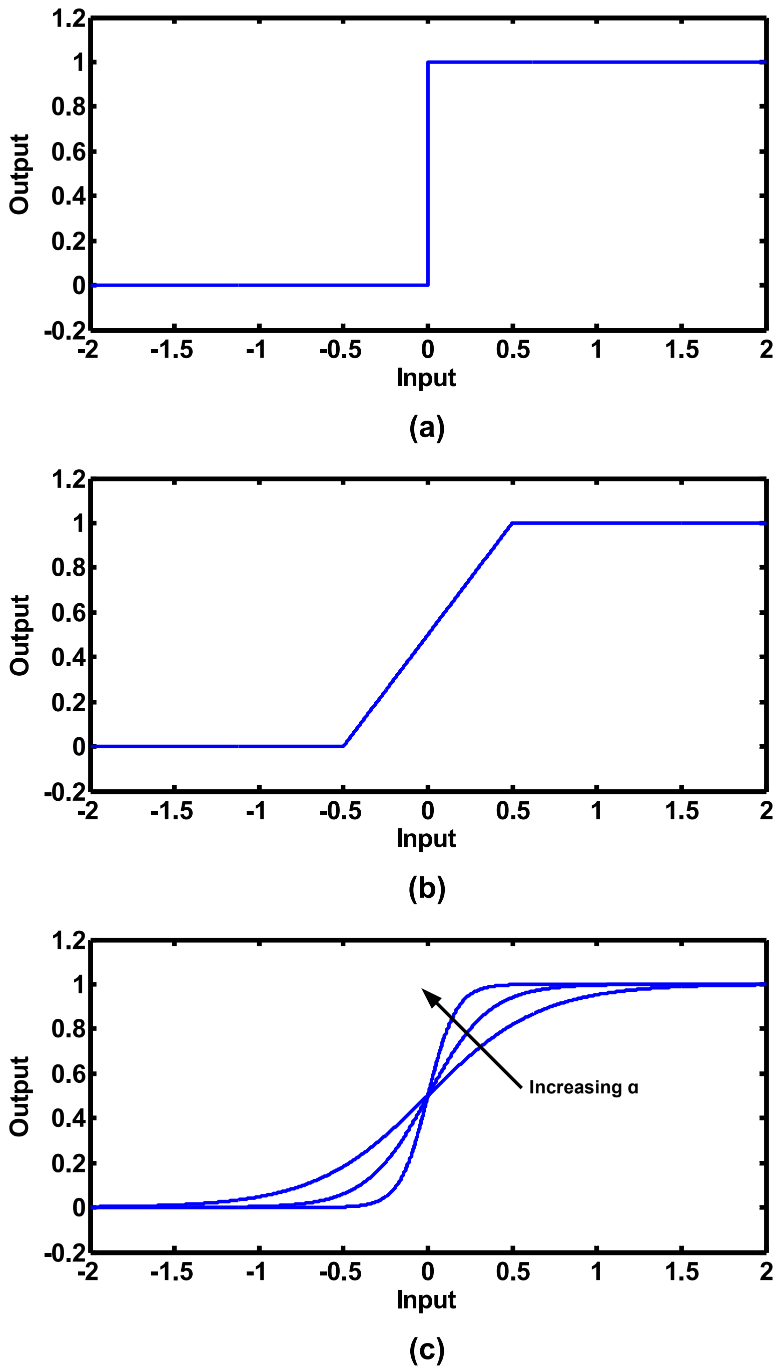


Figure 2.3: Activation functions: (a) step, (b) piecewise linear, and (c) sigmoid with different parameter a.

2.3(b) is expressed by

$$\varphi(v) = \begin{cases} 1 & \mathit{if} \quad v \geq +\frac{1}{2} \\ v & \mathit{if} \quad +\frac{1}{2} > v > -\frac{1}{2} \\ 0 & \mathit{if} \quad v \leq -\frac{1}{2} \end{cases} \tag{2.4}$$

where the amplification factor for the linear region is unity. It has two saturation output levels corresponding an upper and lower bounds (*e.g.* 0 and 1 in (2.4)) and provides a linear response between them.

Third, the sigmoid function is a smooth version of the piecewise linear function in (2.4), and produces an S shaped graph. Moreover, it is most commonly adopted in the construction of ANNs and is mathematically defined by

$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{2.5}$$

where a is the slope parameter. Adjusting the parameter a allows the sigmoid function to generate different slopes as shown in Figure 2.3(c).

The artificial neurons are combined to form a neural network in many different ways. Most of the ANN architectures exhibit the layered structure. Figure 2.4 illustrated a typical feedforward ANN architecture. It has three layers of input, hidden and output neurons where a set of artificial neurons constitutes a layer. Typically, a standard L-layer ANN consists of an input layer, $L-2$ hidden layers and an output layer and they are connected successively. It is worth to note that the hidden layer can be omitted in practice. The network can be connected either fully or partially. The communication proceeds layer by layer from the input to the output layers through the hidden ones. The neuron states of the output layer indicate the

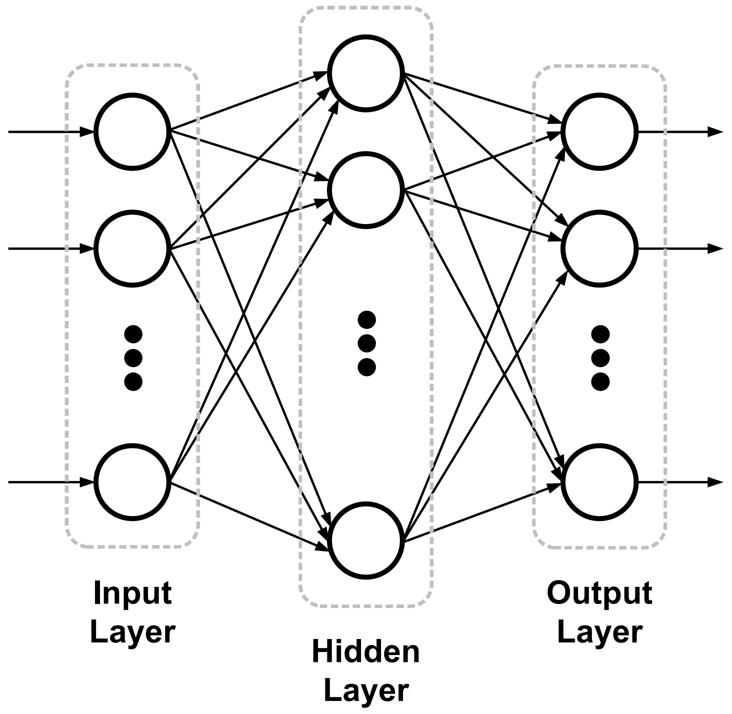


Figure 2.4: Feedforward artificial neural network architecture.

computation result of the network. The neurons in the input layer receives external input signals in the form of activation pattern and projects them onto the next layer (*e.g.* hidden layer in Figure 2.4). The hidden layer plays a role of meditating between the external input and the network output. The addition of more hidden layers allows the network to perform high-order computations, which is particularly valuable when the size of the input layer is large [20]. The neurons in the output layer produce an overall response of the network under the external inputs. In contrast to the feedforward architecture, the recurrent network includes at least one feedback loop, which affects the learning capability of the network and its performance. This network behave like a sequential logic and thus the previous experience may have an impact on the activation of the neurons. The recurrent network is a dynamic system while the feedforward is a static one. When a new input pattern is given, the neuron outputs are computed, then the inputs to each neuron are modified due to the feedback loops, which makes the network enter a new state. Hence, this network topology can be used as associative memory.

Learning makes us to either increase knowledge or enhance understanding, and is archived by studying, experiencing or receiving instructions. For ANNs, learning refers to a process to adjust synaptic weights so that the network is able to perform a specific task efficiently. Many learning algorithms have been presented to appropriately adjust the synaptic weight values of the neural network but they are classified into two main learning paradigms: 1) supervised and 2) unsupervised [30].

Supervised learning is a synaptic weight change process that incorporates the global information. In short, this training process is performed with a teacher that has knowledge or correct answers. The underlying principle of this learning algorithm is the error correction rule that leverages the error signal (*i.e.* difference between the actual output and the correct output) to adjust the synaptic weight values to reduce the error gradually. In supervised learning, every training input is given to the network with each desired output (*i.e.* correct answer). The synaptic weights of the network are modified to produce the outputs as close as possible to the known correct answers. Therefore, the neural network tries to emulate the teacher gradually through the learning process. The backpropagation and perceptron learning algorithms are well known supervised learning methods.

In contrast, unsupervised learning does not have a teacher and utilizes only local information during the learning process. Briefly, it does not require a correct answer associated with each training input for the learning. This learning leverages the properties or correlations of the training inputs, and tries to organize patterns into categories from these correlations. The basic principle of unsupervised learning is that output units (*i.e.* neurons) compete among themselves for activation. Therefore, it allows only one output neuron to be activated at any given time. This phenomenon is referred to as winner-take-all (WTA), which is a common feature of unsupervised learning. Generally, the input patterns form a vector and the network maps the vectors into the synaptic weights. An example of the unsupervised learning algorithms is competitive learning [30].

2.1.3 Spiking Neural Networks

While the conventional ANNs described in Section 2.1.2 are a powerful computational tool to solve complex problems, they still suffer from fundamental limitations in emulating a real biological neural system due to the lack of temporal information of spikes. ANNs have become more powerful and biologically realistic. Spiking neural networks (SNNs), referred to as the third generation of ANNs, have been developed by considering the communication among neurons with precise timing information of the spikes. They more realistically resemble the biological brain than the conventional ANNs [18].

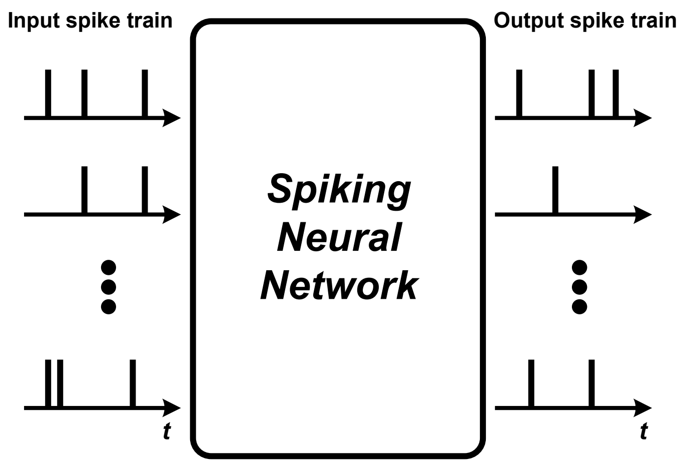


Figure 2.5: Spiking neural network.

SNNs exploit both the presence and timing of individual spikes as the means of communication among the spiking neurons while the conventional ANNs process neural information with real-valued numbers. As in Figure 2.5, an SNN receives the input spike train from the external environment, processes it, and produces the output of the network in the form of another spike train. In an SNN, it is assumed that the amplitude of spikes does not encode any information. Instead, information is encoded in the timing of the spikes that forms a spike train. Therefore, input vectors for an SNN have to be preprocessed to extract the input features that may contain real-valued timing information [18]. Similarly, the output spike train has to be decoded to interpret the result of the network. There are various coding schemes for inputs and outputs for SNNs to interpret a spike train as real-valued numbers, by using either the frequency of the spikes or the timing between the spikes.

Spiking neurons are similar to the conventional artificial ones, but spiking neurons utilize spikes as input and output while the traditional ones have real-valued counterparts. When the spikes from the pre-synaptic neurons arrive at a post-synaptic neuron, the membrane potential of the post-synaptic neuron changes. The membrane potential represents the internal state of the spiking neuron that is induced in the model to respond to pre-synaptic spikes. The membrane potential is affected by the synaptic characteristics such as strength of the synaptic connections. The post-synaptic neuron fires when its potential reaches a specific threshold. The behavior of spiking neurons is illustrated in Figure 2.6. The post-synaptic neuron connects with three pre-synaptic neurons that transmit their output spikes as inputs to the post-synaptic neuron. The pre-synaptic neurons produce the spike trains that consist of a sequence of three, two and one spikes, respectively, and these spikes occur at the different timings. Hence, the post-synaptic neuron receives six input spikes in total. The membrane potential of the post-synaptic neuron increases whenever

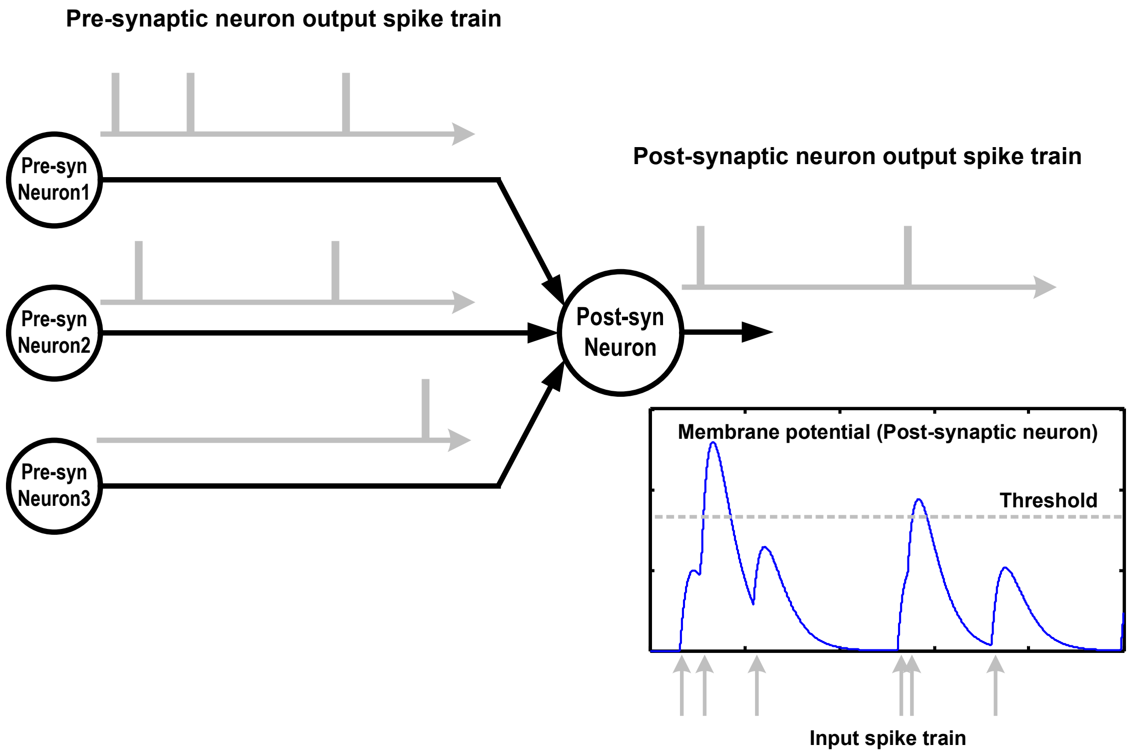

Pre-synaptic neuron output spike train
Post-synaptic neuron output spike train
Pre-syn Neuron1
Pre-syn Neuron2
Pre-syn Neuron3
Post-syn Neuron
Membrane potential (Post-synaptic neuron)
Threshold
Input spike train

Figure 2.6: Spiking neuron behavior.

each spike receives as shown in Figure 2.6. It is important to note that the potential can either increase or decrease according to the type of neurons. In other words, inhibitory pre-synaptic neurons depress the membrane potential of the post-synaptic neuron whereas excitatory ones potentiate. All the pre-synaptic neurons in Figure 2.6 are assumed to be excitatory. The post-synaptic neuron temporally integrates the incoming spike trains to compute the internal state of the neuron (*i.e.* membrane potential) over time. The post-synaptic neuron generates a spike when the potential exceeds the threshold (*e.g.* two output spikes in Figure 2.6). The output spike train of the post-synaptic neuron can be either transmitted to other spiking neurons in the SNN or read off the external environment. Similar neuron behavior can be modeled with many different ways by exploiting the existing spiking neuron models, such as

Hodgkin-Huxley and Leaky Integrate-and-Fire models [54].

Similar to the traditional ANNs, SNNs also learn through synaptic plasticity which refers to an adaptation process that updates the strength of the synaptic connections among the neurons over time, in response to their increased or decreased activities. Neuroscientific research revealed that the change in the synaptic strength depends on the timings of pre- and post-synaptic spikes [42, 33]. This dependency was experimentally characterized in detail by Bi and Poo [3] and named spike timing dependent plasticity (STDP) [62]. STDP is most commonly utilized in modeling of circuit-level plasticity and learning [16]. Furthermore, the STDP rules have been recognized in a wide variety of tasks including associative memory and pattern recognition. STDP is basically a temporally asymmetric form induced by temporal

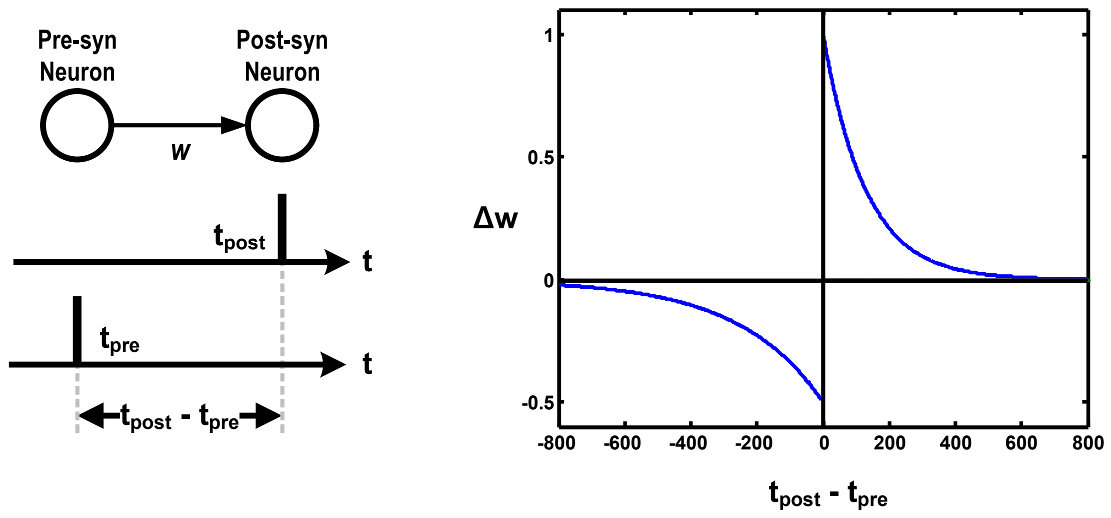


Figure 2.7: Spike timing dependent plasticity.

correlations between the spike firing events between pre- and post-synaptic neurons. Namely, the strength change of synaptic connection is a function of the spike time difference between pre- and post-synaptic firing events and the difference determines the synaptic weight change as illustrated in Figure 2.7. As one example, to achieve the STDP-based learning, the time difference $\Delta t = t_{post} - t_{pre}$ of the firing times between the pre- and post-synaptic neurons needs to be calculated. Then, the synaptic weight update is done by adding the weight change Δw obtained from the STDP curve into the synaptic connection strength w between the pre- and post-synaptic neurons. In this case, the STDP learning is mathematically described by the following equations

$$\Delta w = W(t_{post} - t_{pre}) \tag{2.6}$$

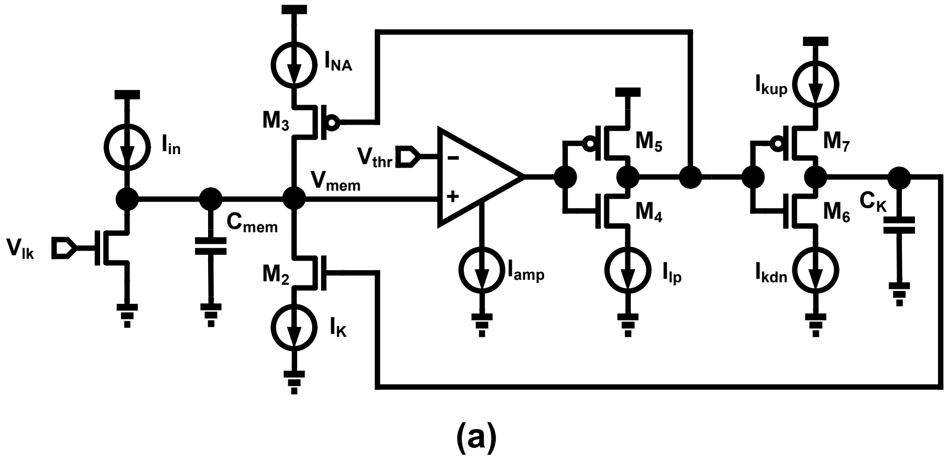
$$w = w + \Delta w \tag{2.7}$$

where t_{pre} and t_{post} are the firing times of the pre- and post-synaptic neurons, respectively, $W(\cdot)$ is the STDP learning function and w is the synaptic weight between the neurons. Since the STDP function $W(\cdot)$ affects the learning performance of the SNN, it should be carefully designed according to the targeted applications.

The rest of this dissertation focuses only on SNNs, which are our primary targeted computational model.

2.1.4 Silicon Neuron Circuits

Silicon neurons are the fundamental building blocks in neuromorphic hardware design [1]. They emulate the electrophysiological behavior of real neurons on a silicon chip rather than on a general purpose computer as software. Many design approaches to implement silicon neurons with analog and digital circuits have been presented [29]. Moreover, several different devices, ranging from conventional CMOS devices to recently developed nano-electro devices such as memristors, are exploited to realize silicon neurons. Considering the presented neuron models so far, the LIF neuron model is widely adopted to implement silicon neurons due to its simplicity in hardware implementation. In contrast, more detailed ordinary differential equation (ODE)-based models such as the Hodgekin-Huxley model can reproduce the behavior of biological neurons more closely, they are less hardware friendly and appropriate for large scale integration.



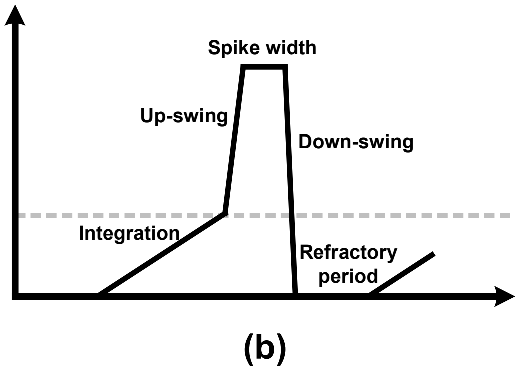


Figure 2.8: Analog silicon neuron: (a) schematic and (b) timing diagram [29].

Traditionally, silicon neurons have been implemented with analog circuits, which utilize the I-V characteristics of the transistors to mimic the biological neurons [48, 65, 69, 11]. Figure 2.8 depicts an analog LIF silicon neuron [29]. The capacitor C_{mem} is used to keep the membrane potential and the leakage current of the membrane is controlled by the gate voltage V_{lk} of M_1 as in Figure 2.8(a). The switches M_2 and M_3 play the role of charging and discharging the membrane capacitor C_{mem}, respectively. The circuit utilizes an analog comparator to compare the threshold V_{thr} with the membrane potential V_{mem}. The emulated membrane voltage over time is illustrated in Figure 2.8(b). If there is no input I_{in} applied, the membrane voltage V_{mem} is drawn to its resting potential, which is $0\ V$ in this configuration, by the leakage current. If an excitatory input by a positive I_{in} is injected to the neuron circuit, then the membrane capacitor C_{mem} is charged while the inhibitory input by a negative I_{in} discharges C_{mem}. When the excitatory current is larger than the leakage current, the membrane potential V_{mem} increases. The potential V_{mem} is compared with the threshold voltage V_{thr} by the comparator which produces a spike when V_{mem} exceeds V_{thr}. The spike turns the switch M_2 on and, as a result, V_{mem} is reset to the resting potential by I_k. The analog circuit based silicon neurons may have a simple structure with a few transistors and consume low power. Unfortunately, they are intrinsically sensitive to process, voltage and temperature (PVT) variations and thus it is essential to design carefully to make the circuits robust. Analog circuits are difficult to scale with technology, reconfigure, and interface with software systems. Importantly, the use of area-consuming capacitors to maintain a considerable number of synaptic weights and membrane potentials hinders large-scale integration of spiking neurons [28, 59].

A digital implementation of silicon neurons is illustrated in Figure 2.9 [6]. The neuron circuit mainly consists of a digital adder, a digital comparing circuit and some control blocks. The accumulator (*i.e.* register) stores an integrated digital

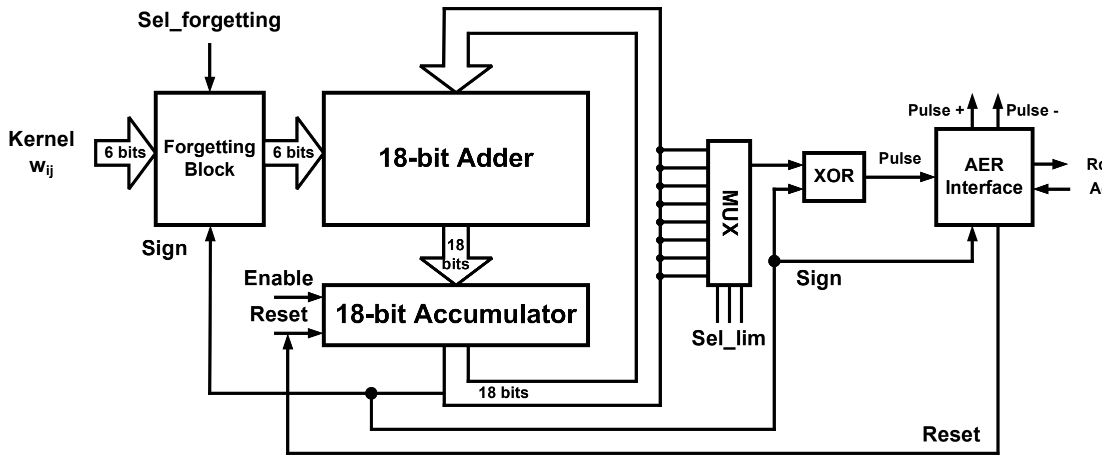


Figure 2.9: Digital silicon neuron [6].

value represented by a 2's complement signed number and its output correspond to a membrane potential. When the event input occurs, *Enable* signal is activated and the accumulator is updated with the corresponding kernel weight w_{ij} (*i.e.* synaptic weight value) through the adder. If the accumulator output reaches a programmed threshold, the neuron generates an event pulse (*i.e.* spike). The threshold value is determined by a 3-bit parameter *Sel_lim* that selects one of the 18 accumulator output bits via the multiplexer. While a lower bit selected leads to a small threshold value, a higher one results in a large threshold. For instance, selecting the $(5)th$ and $(8)th$ bits sets the threshold value to 32 (2^5) and 256 (2^8), respectively. The selected bit is compared with the MSB of the accumulator continuously and the comparator (*i.e.* XOR gate) creates a spike when the MSB and the selected bit is different from each other. Lower thresholds make the neuron fire more frequently, and thus allow fast processing. The forgetting block mimics the leaking behavior of neurons. The periodic forgetting pulse *Sel_forgetting* is applied to the forgetting block. If *Sel_forgetting* signal is asserted, the block outputs a fixed leaky value (*e.g.* -1) so the accumulator output decreases, causing the membrane potential to decrease periodically. The digital neuron circuits are advantageous in terms of robustness against PVT variations, better reconfigurability and ease of implementation. However, it may consume large dynamic power due to digital switching activities.

2.1.5 Neuromorphic VLSI Systems

At the system level, while the conventional computers have become very powerful but they still require a huge amount of capacity, power and computation time to mimic the tasks that humans behave, VLSI-based neuromorphic systems can provide effective ways to emulating the functions of a biological brain in silicon while significantly saving computational power and time. Neuromorphic chips are usually composed of synapse circuits and silicon neurons that store synaptic weight values and emulate neuron dynamics, respectively. In addition, they can also include dedicated on-chip learning circuits according to the application. Recently, two digital reconfigurable neuromorphic chips integrating a large number of spiking neurons and their building blocks have been demonstrated in [58, 46, 27, 2]. These two designs support up to 256 programmable digital silicon neurons and 1024$\times$256 binary synapses by means of an SRAM crossbar array.

Figure 2.10 shows the block diagram of the digital neurosynaptic core in [46]. It consists of 256 digital LIF neurons with an output encoder, 1024 individually addressable axons, which can be either excitatory or inhibitory, with an input decoder and 1024$\times$256 programmable binary synapses implemented with an SRAM crossbar array. This design does not include an on-chip learning mechanism and necessitates loading of synaptic weights into the crossbar after the off-chip learning. It performs neural information processing in an event driven manner to save active power dissipation greatly. Specifically, it adopts an asynchronous design technique where all

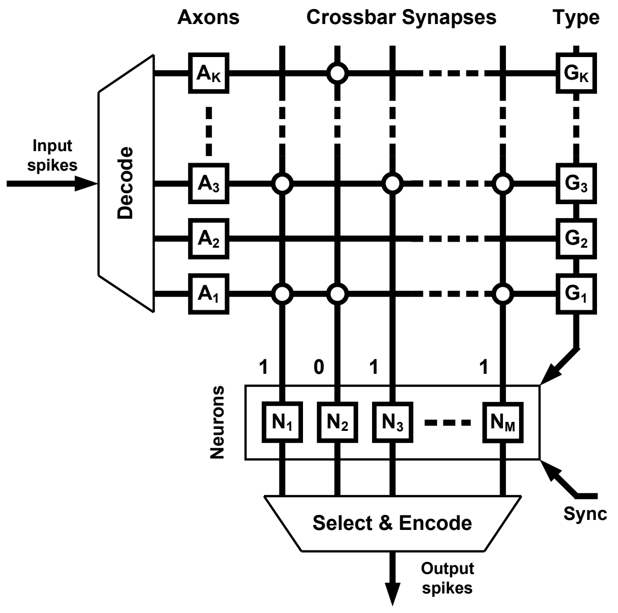


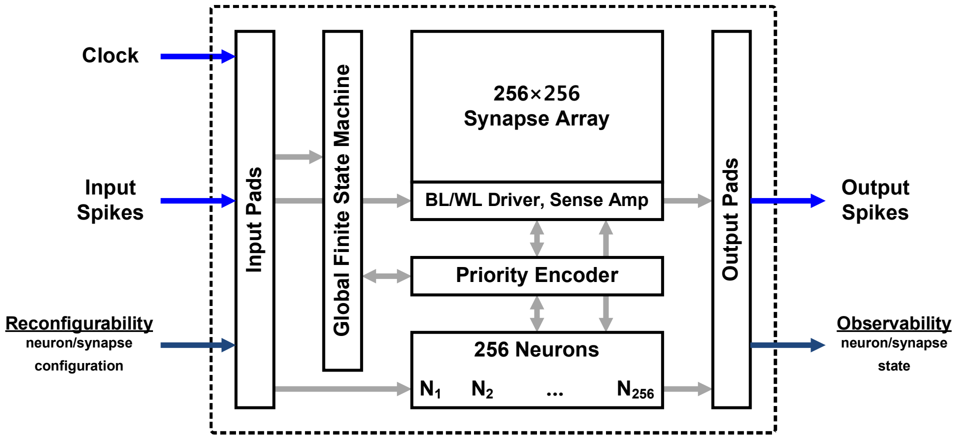
Figure 2.10: Block diagram of digital neurosynaptic core [46].

communication among the blocks requires a request-acknowledge handshake without a global clock signal. The detailed operation in each time step t is divided into two phases. In the first phase, a set of input spike-events A are sent to the neurosynaptic core at a time, and these events are sequentially decoded to the appropriate axon block. The corresponding axon activates the SRAM's row, and reads out all of its connections and type G. If there are synaptic connections W, which are represented as "1", the inputs are sent to the corresponding neurons circuits. Then, they update the membrane potentials V appropriately. After a sequence of neuron updates, the axon block deactivates the SRAM and waits for the new inputs. In the second phase, a synchronization event that occurs in every millisecond period is sent to all the digital neurons. Each neuron checks whether its membrane potential reaches certain threshold. If so, it produces a spike and resets the potential to zero. These spikes of the neurons are encoded and sequentially sent off the chip through the encoder. After that, the leak parameter is applied to the neurons. Throughout the two phases of neural processing, the neurosynaptic core implements the neuron dynamics described by the following mathematical expression.

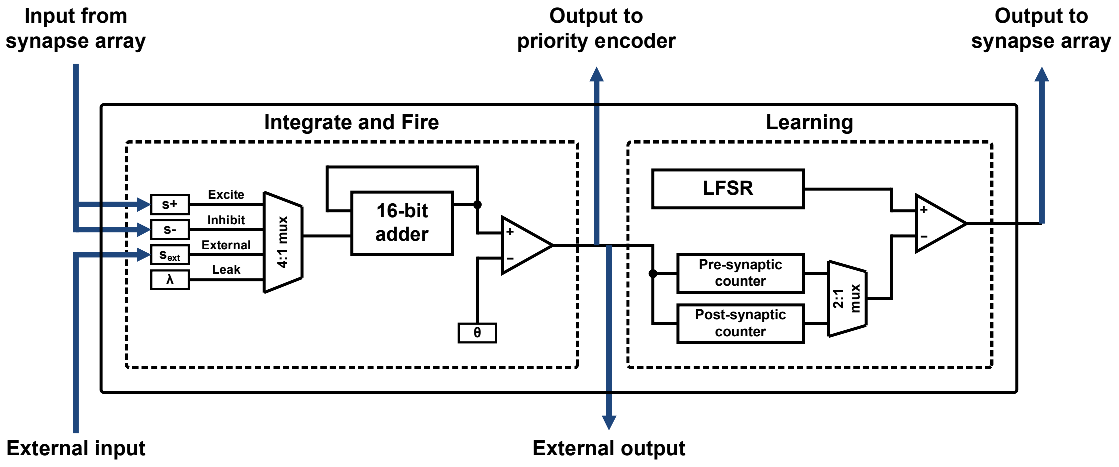
$$V_i(t+1) = V_i(t) + L_i + \sum_{j=1}^{K} \left[A_j(t) \times W_{ji} \times S_i^{G_j} \right] \tag{2.8}$$

where L is the leaky parameter, K is the number of axons, A is the activity bit, W is the synaptic value and G is the axon type.

The digital neuromorphic chip incorporating on-chip learning capability in [58] is illustrated in Figure 2.11. This chip integrates 256 digital spiking neurons with 256×256 binary synapses to implement a fully recurrent network. It operates in a synchronous manner with a global hardware clock signal and each biological step consumes many hardware clock cycles. In each time step, the digital spiking neurons calculate their membrane potential according to the implemented dynamics and produce spikes when the potentials reach the given firing threshold. The spikes created by the firing neurons lead to synaptic integrations in all target neurons (*i.e.* post-synaptic neurons) and synapse weight updates according to certain learning rule. While the input spikes to the system represent external stimuli such as input patterns, the generated spikes (*i.e.* output spikes) indicate the output activities (*e.g.* recognition of a given pattern). The $N \times N$ crossbar architecture is suitable to represent a neural network of N neurons and all N^2 possible synaptic connections among them. Correspondingly, the on-chip storage used to keep the binary synapse values is implemented by a 256×256 array of transposable SRAM cells. While the conventional memory array is accessible only in row-direction, the proposed transposable SRAM array is accessible in both row- and column-fashions for pre-synaptic and post-synaptic weight updates, respectively, leading to a significant speedup of the



(a)



(b)

Figure 2.11: Block diagram of (a) neuromorphic chip and (b) silicon neuron [58].

update process. Moreover, an entire row and column of the crossbar can be accessed simultaneously. Note that each row and column corresponds to a neuron's axon and dendrite, respectively, in the SRAM array. Based on the adopted single-bit memory cells, the binary weight of the synapses are probabilistically set to "1" or "0" according to the implemented learning rule. Each neuron in the Figure 2.11 implements both locally reconfigurable LIF functions and learning rules, and the following LIF neuron dynamics is conducted in each time step

$$V[t] = V[t-1] + s_+ n_+[t] - s_- n_-[t] - \lambda \tag{2.9}$$

where V is the membrane potential, n_+ and n_- are the numbers of excitatory and inhibitory inputs received through "on" synapses, respectively, s_+ and s_- are the input strength parameters and λ is the leak parameter. The membrane potential and parameters are expressed with 8-bit digital values. Additionally, if the membrane potential V exceeds a given threshold θ that is reconfigurable variable, a spike is generated and V is reset to the resting potential. On-chip learning is implemented in each neuron cell, which thus shares the learning circuit across axonal rows and dendritic columns of the synapses. The pre- and post-synaptic counters work with a linear feedback shift register (LFSR) to perform probabilistic synapse weight update.

The two neuromorphic designs both employ an SRAM array to store synaptic weight values, incurring a significant portion of the entire chip area. Furthermore, the learning performance may be degraded due to the adopted binary synapses that are updated by a probabilistic scheme [58]. The lack of an on-chip learning mechanism may limit the design of applications [46].

2.2 Approximate Computing

Aggressive CMOS technology scaling allows modern VLSI systems to integrate many high-performance functional modules, such as multi-media and communication processors. Meanwhile, today's circuit designers are facing grand challenges in managing overall chip power and energy consumptions. To remedy the increasing energy efficiency challenges, a new design paradigm of approximate computing has emerged as one promising solution and has drawn a significant research interest

[7, 67, 21, 60, 8, 9]. Approximate computing can provide great computational and energy efficiencies by relaxing processing precision while maintaining an acceptable overall processing quality for many applications that involve signal processing of multimedia data (*e.g.* audio, video and image), machine learning and speech recognition. Fortunately, these classes of computation do not require perfect accuracy and approximate results with controlled accuracy may be sufficient. For example, while approximation errors in image processing may change the numerical values of the overall output, the users may not be very sensitive to certain amount of error and may still recognize the image. Similarly, there is certain level of error resilience in tasks performed by the human brain. The human brain is often able to fill the missing information and filter out the noisy or redundant information from the received inputs by its natural error compensation mechanisms. In short, the human brain has a certain degree of built-in error or fault resilience. Thus, it makes good sense to system design to reduce the cost in hardware realizations such as power, energy and area with approximate computing from a ranging from the algorithm to circuit levels [21, 60, 8, 9].

Certainly, addition is the fundamental operation in many processing applications and the adder is therefore an essential component to achieve approximate computing. Furthermore, other arithmetic units, such as comparator and multiplier, can be implemented based on the adder. Since approximate adder design is one primary contribution of this dissertation, we briefly review the existing approximate adder design techniques. Lu proposes an approximate adder [39] that leverages a limited number of previous (less significant) input bits for carry speculation to increase the overall speed by cutting down the long carry propagation chain. The critical drawback of this approach is the use of a considerable number of carry generators, which gives rise to large area and high power dissipation. The so-called ETAI [76] and LOA

[40] approximate adders are split into an accurate part for higher order bits and an inaccurate part, which utilizes a modified XOR (ETAI) and OR function (LOA) to approximately compute the remaining lower output bits. Therefore, the approximate errors are concentrated on the lower bits. A few transistors are eliminated from the traditional mirror adder to reduce power and area at the expense of accuracy degradation in [19]. These two approaches are limited by high error rates. The ETAIII [75] improves the error rate of ETAI by introducing a dynamic dividing strategy of the accurate and inaccurate parts by input patterns but still yields a high error rate. The segment based approximate adders are presented in [74] and [14] which are named ETAII and VLCSA-1, respectively. The carry for each k-bit segment is predicted from the lower k-bit inputs to reduce the delay of carry propagation. Similarly, the ACA [32] adopts a number of $2k$-bit sub-adders and leverages only k most significant bit (MSB) outputs of the sub-adders to achieve approximate additions. Unfortunately, these adders have high error rates for the carry generations, particularly for 2's complement signed additions of small numbers. In addition, the use of carry selection in VLCSA-1 and middle sub-adders in ACA result in power consumption and area overhead. The lack of an error magnitude reduction in ETAII degrades the quality of addition. In [47], the approximation errors for less significant bits are reduced by conditional bounding logic with dithering, which causes area and power overheads.

2.3 Emerging Memory Technologies

To date, various new memory technologies have emerged to replace the traditional memories such as SRAM and DRAM. Among these new memory technologies researched so far, spin-torque-transfer random access memory (STT-RAM), phase-change memory (PCRAM) and memristor based resistive random-access memory

(ReRAM) are considered the most promising candidates for the future. STT-RAMs exploit a magnetic tunnel junction to store information and the difference in magnetic directions is used to represent a bit of information [12]. PCRAMs leverage chalcogenide materials for memory storage which can be switched between a crystalline phase (SET state) and an amorphous phase (RESET state) by heat [12]. ReRAMs are typically implemented with a memristor, as known as memory resistor, whose existence was theoretically predicted by Chua in 1971 as the fourth fundamental passive circuit element [10]. More recently, TiO_2 thin-film based memristors have been demonstrated at the nanoscale [63]. The memristive nanodevice has gained increasing research interest and becomes a promising solution for low-cost on-chip storage thanks to its non-volatile nature, excellent scalability and high density of 10 Gb/cm^2 or more [23, 72]. A number of multibit hybrid CMOS/memristor memory architectures targeting high integration density and low power dissipation have been proposed to substitute the conventional SRAM and flash memories that are confronted with the fundamental technology scaling limits [45, 41]. In addition, several recent studies have suggested leveraging memristive nanodevices for building synaptic arrays [31, 61, 55, 25].

We briefly review the memristor device model which is used for implementing the on-chip synaptic weight storage in our neuromorphic processor design. A TiO_2 thin-film based memristor is a two terminal electrical device and is a titanium oxide film sandwiched by two metal contacts. Conceptually, there are two layers in the film: a doped and an undoped ones as shown in Figure 2.12. The undoped layer is a highly resistive pure TiO_2 region (TiO_2 layer) while the doped one is filled with oxygen vacancy that makes it highly conductive (TiO_{2-x} layer) [23]. The memristor

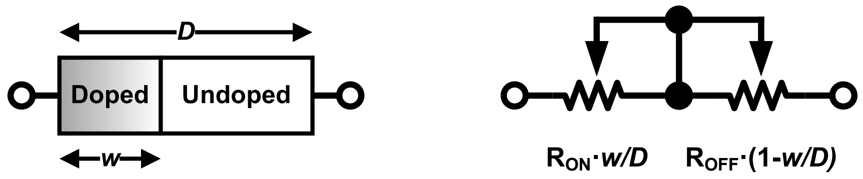


Figure 2.12: Memristive device structure (left) and variable resistance model (right) [63].

device model can be mathematically expressed by

$$\begin{gathered} R(w) = \frac{w}{D} \cdot R_{ON} + \left(1 - \frac{w}{D}\right) \cdot R_{OFF} \\ where\ \ 0 \le w \le D \end{gathered} \tag{2.10}$$

In (2.10), R_{ON} and R_{OFF} are the fully doped (lowest) and fully undoped (highest) resistances of the memristor, respectively, and w is the length of the doped region and thus physically bounded by the range between 0 and the total device length D. Moreover, w represents the internal state of the memristor. The memristor state is controlled by the incoming flux across the device for flux-controlled memristive devices where the voltage source is utilized as the input. The memristance is determined by the state w that can be mathematically described by

$$\frac{w}{D} = \begin{cases} 1 & if\ \varphi \ge \Phi_{UP} \\ \frac{\beta}{\beta-1}\left[1 - \sqrt{\left(\frac{R(w_0)}{R_{OFF}}\right)^2 - \frac{\varphi}{\Phi_D}}\right] & if\ \Phi_{LOW} < \varphi < \Phi_{UP} \\ 0 & if\ \varphi \le \Phi_{LOW} \end{cases} \tag{2.11}$$

$$\Phi_{UP} = \frac{\Phi_D}{R_{OFF}^2}\left(R(w_0)^2 - R_{ON}^2\right) \tag{2.12}$$

$$\Phi_{LOW} = -\frac{\Phi_D}{R_{OFF}^2}\left(R_{OFF}^2 - R(w_0)^2\right) \tag{2.13}$$

where the ratio between the resistances of the on and off states is denoted by β (*i.e.* $R_{OFF} = \beta R_{ON}$), w_0 is the initial state of the memristor, D is the total length of the memristor and φ is the injected flux. In addition, Φ_{UP} and Φ_{LOW} are the upper and lower limits of the effective flux injection, respectively, and

$$\Phi_D = \frac{(\beta D)^2}{2\mu_v(\beta - 1)} \tag{2.14}$$

where μ_v is the average ion mobility. Also, the memristor internal state w varies dynamically with the external input. A recent experimental study has shown that the conductance of the memristive device can be incrementally adjusted by altering the pulse width of the constant input voltage [31]. In other words, a longer positive pulse duration leads to a larger increase of memductance that is given by the inverse of memristance. Suppose that the state of the memristor is moved from the initial state w_0 to a feasible state w by a square-wave voltage pulse with an amplitude of V_A. Then, the required pulse width T_W is can be derived to be [23]

$$T_W = \frac{\Phi_D}{V_A R_{OFF}^2}\left[R(w_0)^2 - R(w)^2\right] \tag{2.15}$$

Furthermore, the pulse duration required to move the memristor state from $w = 0$ to $w = D$ is the same as what is needed to change the state from $w = D$ to $w = 0$, and the pulse width T_W for these moves is given by

$$T_W = \left|\frac{\Phi_D}{V_A R_{OFF}^2}\left(R_{OFF}^2 - R_{ON}^2\right)\right| \tag{2.16}$$

Note that the polarity of the input pulse would be different from each other for these moves. This equation also indicates that the programming time (*e.g.* pulse width) needed to write a specific value to the memristor would be a function of the current and target resistances of the memristor.

2.4 Objective of the Dissertation

The objective of the dissertation is to realize energy efficient and error resilient neuromorphic computing in VLSI. To achieve this goal, we first introduce a general digital neuromorphic VLSI architecture. The block diagram of a this digital neu-

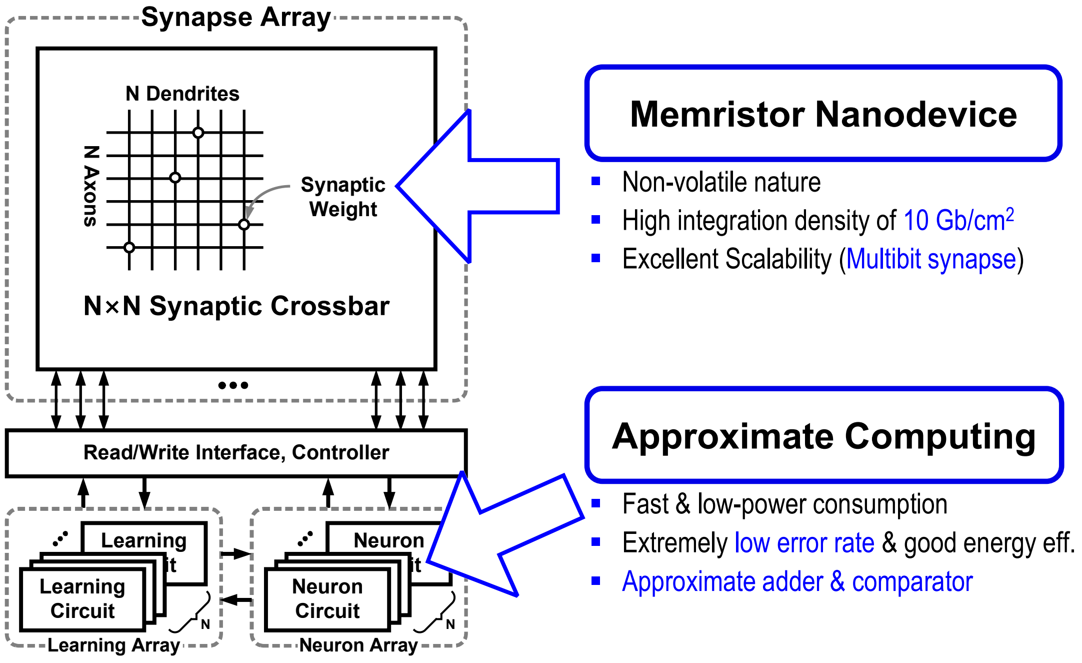

Synapse Array
N Dendrites
N Axons
Synaptic Weight
N×N Synaptic Crossbar
Read/Write Interface, Controller
Learning Circuit
Learning Array
Neuron Circuit
Neuron Array
N
Memristor Nanodevice
Non-volatile nature
High integration density of 10 Gb/cm²
Excellent Scalability (Multibit synapse)
Approximate Computing
Fast & low-power consumption
Extremely low error rate & good energy eff.
Approximate adder & comparator

Figure 2.13: Energy efficient and error resilient neuromorphic computing in VLSI.

romorphic architecture with N spiking neuron networks is depicted in Figure 2.13. The design consists of three arrays of synapse, learning, and neuron circuits as well as a control and an interface circuits for them. The $N \times N$ crossbar synapse array represents a fully recurrent network topology and can store all possible N^2 synaptic weights among the N neurons. The resolution of the synaptic weights is determined according to the targeted applications [56], and can be either binary [58, 46] or multibit [50]. To mimic the behavior of a biological neuron, each neuron circuit emulates the neuron dynamics (*e.g.* according to the LIF or Hodgkin-Huxley models) and generates spikes when it fires. The learning circuits cooperate with the respective neurons and update the synaptic weights according to a learning rule such as STDP.

To realize energy efficient and error resilient neuromorphic VLSI systems (see Figure 2.13), we adopt one of the emerging nanodevices, memristor, to implement a low-cost $N \times N$ synaptic crossbar array. While the conventional SRAM-based synaptic array suffers from significant area overhead, in particular, for multibit synapses, the memristor-based nonvolatile arrays provide an excellent scalability and high integration density of 10 Gb/cm^2 or more. We systematically analyze the memristor device characteristics in terms of access time and level partitioning to realize multibit synapses and present a low-cost digital PWM scheme for programming the memristor. Furthermore, we investigate memristor readout schemes for the crossbar array and propose a column based analog-to-digital conversion technique to more efficiently carry out the digital LIF neuron dynamics. The details of the design of the memristive crossbar is presented in Section 3.

As the second key contribution which will be presented in Section 4 and 5, we apply approximate computing to the digital neuromorphic VLSI system to considerably reduce energy dissipation since approximate computing allows for fast computation with low power consumption, leading to good energy efficiency with an acceptable computation quality. To do this, a novel approximate arithmetic scheme, referred to as parallel carry-skip, is proposed for adders and comparators in Section 4. By cutting down the worst-case carry propagation chain, we reduce the critical path delay and leverage the information from less significant input bits to speculate the carry in a parallel manner, which allows for highly accurate carry prediction. Thus, it makes it possible to either speed up addition and comparison operations or reduce energy dissipation by lowering the supply voltage level. We adopt the proposed approximate arithmetic units for the digital LIF computations in the neuron circuit and show the impacts of the approximation errors on the VLSI-based neurocomputing applications. Moreover, we systematically analyze the energy efficiency of the neuromorphic hardware adopting the approximate components with supply voltage scaling in Section 5.

Finally, Section 6 summarizes our key contributions and discusses the future works.

3. RECONFIGURABLE DIGITAL NEUROMORPHIC PROCESSOR WITH MEMRISTIVE CROSSBAR ARRAY

3.1 Digital Neuromorphic Processor Architecture

3.1.1 Overall Processor Architecture

Figure 3.1 depicts the overall block diagram of the proposed neuromorphic processor architecture for an N spiking neuron network. It consists of a synapse unit (SU), a learning unit (LU) with a global timer, a neuron unit (NU), a LIF arithmetic unit (LAU) and a system controller. The SU employs the proposed $N \times N$ memristive crossbar array since the crossbar structure effectively implements biological synapse connections [58, 46]. It can represent a fully recurrent network topology and store N^2 possible synaptic weights among N neurons. A biological neuron has multiple dendrites and a single axon, which receive input spikes from the pre-synaptic neurons and transmits output spikes to the post-synaptic neurons, respectively. One axon can connect with the dendrites of multiple post-synaptic neurons. In the crossbar array, a row and a column corresponds to an axon and a dendrite, respectively, of a biological neuron. The connection between the $(j)th$ row (axon) and $(i)th$ column (dendrite) is represented by the synaptic weight w_{ji} between the $(j)th$ and $(i)th$ neurons. The employed memristor device for the array keeps not only a multibit synapse value but also the network connectivity information. The proposed crossbar is fully reconfigurable in sense that the connectivity can be programmed with respect to the topology for any N-neuron network. The detailed memristor cell utilization of the crossbar array will be explained in Section 3.2. In order to achieve the parallel synaptic weight updates, we design the array to be accessible in both row and column directions and this improves the update performance greatly. Among the

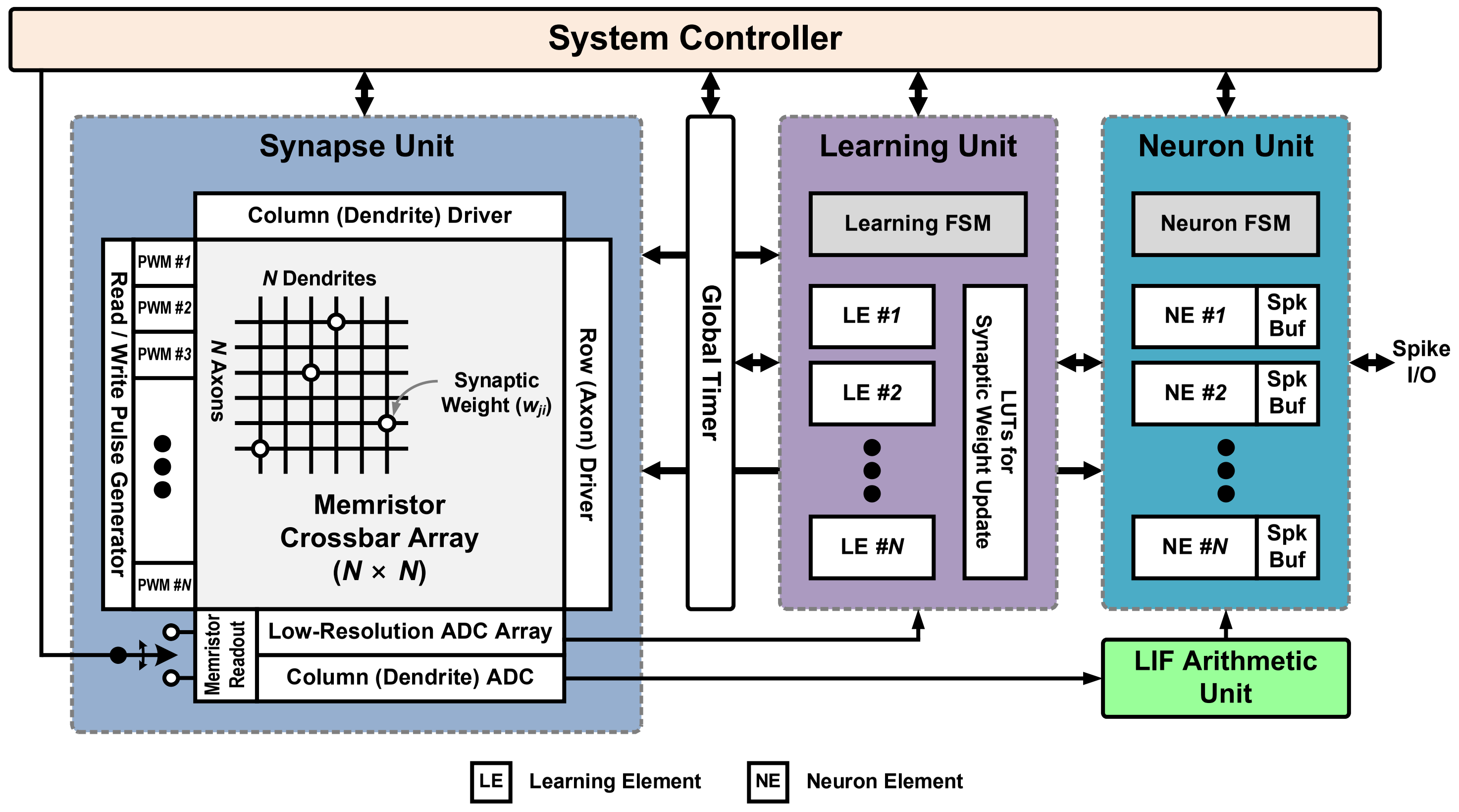


Figure 3.1: Block diagram of the proposed digital neuromorphic processor architecture.

various neuron models, we adopt the LIF model for the silicon neurons to mimic the biological counterparts. LIF models have been shown to be effective for a number of learning applications and are suitable for digital implementation due to its moderate hardware overhead that includes a few arithmetic components, such as an adder and comparator [29].

The NU, which consists of a finite state machine (FSM) and N neuron elements (NEs), emulates the LIF neuron dynamics while interfacing with the column (dendrite) ADC and LAU. In fact, our DNP has two different memristor readout circuits, which are a low-resolution ADC array and a column ADC, which will be described in Section 3.3. Each NE keeps the membrane potential and has spike buffers to store both the external spike that is fed by the off-chip environment and the output spike that is generated when the potential is greater than the given threshold voltage. The NE can be made either excitatory or inhibitory, which potentiates or depresses the membrane potential of the post-synaptic neurons, respectively.

The LU is responsible for performing an on-chip learning. It contains N learning elements (LEs) that cooperate with the respective NEs as well as an FSM to control the overall synaptic weight update process. Each LE has a register to maintain the corresponding neuron's spike timing, which is used to calculate the spike time difference between a pre-synaptic and a post-synaptic neurons. The LU updates the synapse values in the crossbar based on the time differences to realize the STDP learning mechanism. The STDP rule is programmable through the use of look-up tables (LUTs) where synaptic weight change as a function of timing difference is stored. These LUTs are shared by all N LEs and thus able to reduce the silicon-area significantly and our design allows for parallel STDP updates of weights of all neurons in a parallel manner The communications between the proposed neuromorphic processor and the external environment is performed through input and output spikes. External stimulus are applied in the form of input spikes while output results in the form of the generated spikes of the output neurons are outputted. In a character recognition system, for example, input letters are encoded into sequences of input spikes that are applied to the input neurons and recognition (classification) results are identified through spiking activities of output neurons.

3.1.2 Flow Control of the Neuromorphic Processor

The system controller manages the overall operations of the processor through a clocking based synchronous control and the proposed DNP operates in a synchronous manner as shown in Figure 3.2. Each step corresponds to a biological time unit and consumes many hardware clock cycles. It includes three processing stages: 1) spike input/output (I/O); 2) neuron and 3) learning. These stages are executed in a pipelined manner in that the spike I/O and learning stages can work simultaneously because there is no data and control hazards between them. During the spike I/O stage, the input spike buffers in NEs store the spikes from the external environment. Meanwhile, the output spikes can be read off the chip to observe the output activities.

After receiving/transmitting all the input/output spikes, the neuron stage starts,

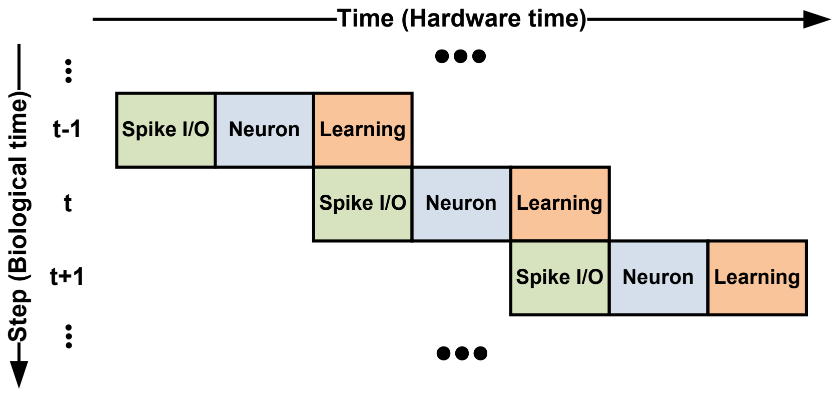


Figure 3.2: Flow diagram of the proposed neuromorphic processor.

where the following LIF neuron dynamics is implemented for each neuron.

$$V_i[t] = V_i[t-1] + K_{SYN} \sum_{j=1}^{M} w_{ji} S_j[t-1] + K_{EXT} E_i[t-1] - V_{LEAK} \tag{3.1}$$

where V_i is the membrane potential of $(i)th$ neuron, M is the number of pre-synaptic neurons, K_{SYN} is the synaptic weight parameter, w_{ji} is the synaptic weight between the $(j)th$ and $(i)th$ neurons, S_j is the activity bit that indicates whether the $(j)th$ neuron fired, K_{EXT} is the external input spike parameter, E_i is the activity bit for the input spike of the $(i)th$ neuron and V_{LEAK} is the leaky potential. At each hardware time step in the neuron stage, through the column driver, an NE activates the corresponding column word line to access all its pre-synaptic neurons. The read/write (R/W) pulse generator, which contains N digital pulse width modulators (PWMs), produces parallel pulses for reading all pre-synaptic weight values from the memristor cells in the column and these values are sent to the column ADC. The ADC accumulates these pre-synaptic weights and converts the sum into a digital quantity. Note that synaptic weights are stored as an analog quantity of memristors' resistance or conductance (*i.e.* memristance or memductance). This reading process will be detailed in Section 3.2. Finally, the NE updates its membrane potential by adding up the accumulated pre-synaptic weights, the external spike weight and the leaky potential through LAU. If the membrane potential exceeds the given threshold voltage, the NE generates a spike event and its potential is reset to the resting potential. The spiking activity bit of the $(i)th$ neuron S_i is set according to

$$S_i[t] = \begin{cases} 1 & if \;\; V_i[t] > V_{TH} \\ 0 & otherwise \end{cases} \tag{3.2}$$

where V_{TH} is the threshold voltage. These spikes are read off the DNP to monitor the output results during the spike I/O stage. The aforementioned process is repeated N times to achieve the LIF operations of all N neurons during the neuron stage.

After the neuron stage, the processing continues onto the third learning phase, where the synaptic weights are updated according to the STDP learning rule. In this rule, the time difference between a pre-synaptic and a post-synaptic spike event is measured and utilized to determine the synaptic weight change. To do this, each LE has a time register to keep track the neuron's spike event time that is stamped by the global timer. For each fired neuron, the LU conducts both a pre-synaptic and a post-synaptic weight updates in a row. If a neuron fires,

1) all its pre- (post-) synaptic neurons' time registers are compared with the global timer and the corresponding LEs determine the amounts of the synaptic weight change using the pre-computed LUT;

2) the column (row) driver activates the memristor crossbar array's column (row) word line that is associated with the dendrites (axons) of the fired neuron;

3) the R/W pulse generater provides a read pulse word to the corresponding column (row) to sense each memristor's current internal state (*i.e.* current synaptic weight) in the column (row) through the low-resolution ADC array;

4) the LEs calculate the pulse durations to write the desired synaptic weight values, which are evaluated in step 2), into the respective memristor cells using the memristor's states obtained in step 3);

5) all the pre- (post-) synaptic weights are updated by means of the R/W pulse generator with the durations determined in step 4).

The generator produces parallel write pulses that have different widths according to not only the amount of the synaptic weight change but also the memristor's current internal state (due to the non-linear device characteristics for write time) as will be described in Section 3.2. This update process works only when the corresponding neuron fired at the neuron stage. In other words, if the membrane potential of the $(i)th$ neuron, for instance, does not exceed the threshold voltage at the neuron stage, then both pre- and post-synaptic weight update processes for the $(i)th$ neuron are skipped. It is worth to note that the entire learning stage is omitted when no neurons fired at the neuron stage. In addition, the proposed architecture is able to process spiking I/O tasks and the learning stage simultaneously since LIF operations are processed in the previous stage, resulting no conflict of spiking event data.

3.2 Memristive Synaptic Crossbar Array

In this section, we first briefly introduce the memristor model and two readout schemes that are suitable for processing LIF operations of silicon neurons and synaptic weight update process. Then, the proposed memristive synaptic crossbar array and CMOS/memristor hybrid cell are presented. Additionally, we propose a new digital PWM scheme for both reading and STDP update of memristive synapses. While an analog PWM scheme has been conceptualized for implementing the STDP earning rule [61], the presented digital design is more amenable to large scale integration in digital system architectures.

3.2.1 Memristor Readout Schemes

Two different ways to read the memristor internal state have been presented based on the sensing device for the memristor: 1) load resistor and 2) summing amplifier (*i.e.* current-to-voltage converter), as depicted in Figure 3.3. The load resistor based sensing scheme is commonly adopted for both binary and multilevel

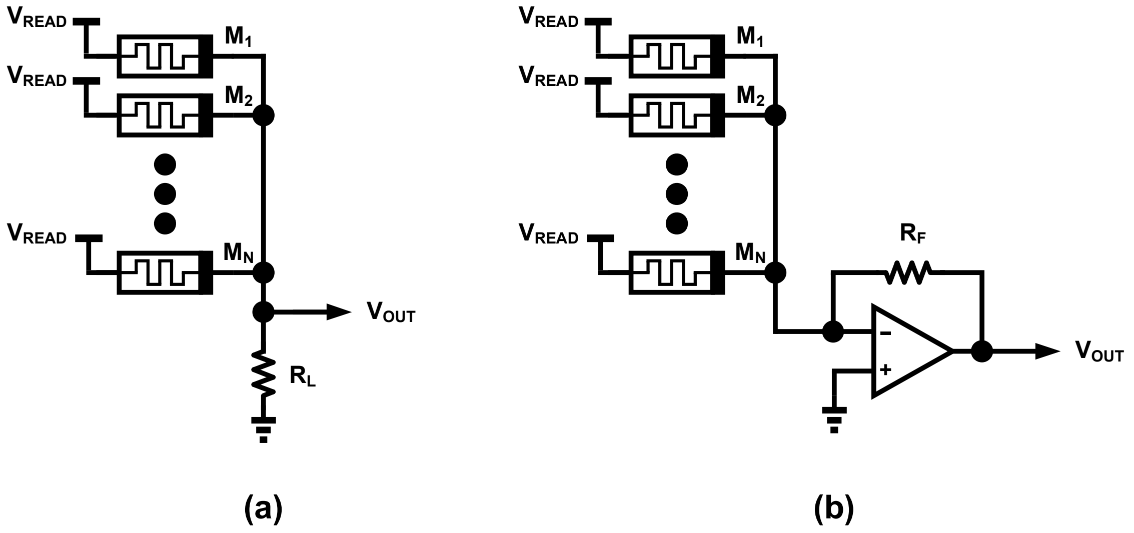


Figure 3.3: Memristor sensing schemes by (a) load resistor and (b) summing amplifier.

memristor memories due to the ease of implementation [23, 45, 41]. This scheme leverages a load resistor R_L, which is connected in series with the memristor, and forms a voltage divider. Hence, the output voltage V_{OUT} of Figure 3.3(a) under a given read voltage V_{READ} is calculated by

$$V_{OUT} = \frac{\sum_{i=1}^{N} \frac{1}{R_i}}{\frac{1}{R_L} + \sum_{i=1}^{N} \frac{1}{R_i}} V_{READ} = \frac{\sum_{i=1}^{N} G_i}{G_L + \sum_{i=1}^{N} G_i} V_{READ} \tag{3.3}$$

where N is the number of memristors attached to the sensing node, R_i and G_i are the resistance and conductance of the memristor M_i, respectively, and R_L and G_L are those of the load resistor. With respect to this particular neuromorphic application, the result shows that this scheme is unable to integrate multiple memristor internal states associate with a STDP update since it can not have the output voltage V_{OUT} represent a linear summation of either memristor resistance or conductance under a fixed read voltage V_{READ}. What is possible with this scheme, though, is the accumulation of all pre-synaptic weights for a neuron in N iterations with use of an additional adder. In other words, each memristor state can be readout and accumulated by the adder at a time, resulting in a total of N^2 iterations to complete all N neurons' LIF tasks. Additionally, to concurrently integrate pre-synaptic weights of N neurons to expedite the tasks, it necessitates N adders and N iterations to finish them [58]. On the other hand, the summing amplifier based sensing scheme provides a linear summation of conductance of memristors such that it is able to integrate all pre-synaptic weights for each neuron. This scheme forms a virtual ground at the negative input terminal of the amplifier and each current from the memristor flows out into the ground. Thus, the output voltage V_{OUT} of Figure 3.3(b) with the input voltage V_{READ} is expressed by

$$V_{OUT} = R_F \sum_{i=1}^{N} \frac{V_{READ}}{R_i} = R_F \sum_{i=1}^{N} G_i V_{READ} \tag{3.4}$$

where R_F is the feedback resistor of the amplifier. This scheme allows the adder and comparator (LAU) to be shared for the LIF task for N neurons, leading to great area and power savings. It is worth to note that both schemes can be equally employed to sense the state of a single memristor while the amplifier based scheme is more suitable to accumulate the internal states of multiple memristors. Hence, we leverage the summing amplifier based sensing scheme to effectively accumulate all pre-synaptic weights of each neuron for LIF operations at the expense of an amplifier, whereas the resistor based sensing is exploited to detect each memristor's current state for the synaptic weight update process.

3.2.2 Memristive Synaptic Cell Partition

Since the summing amplifier based sensing provides the linear summation of conductance of memristors, we make the memristors of the crossbar array have equally partitioned 9 conductance levels to represent a multilevel synaptic weight as shown in Figure 3.4. Importantly, the programming times to write a value to the memristor are drastically different according to the memristor internal state (*i.e.* current conductance value) in spite of writing the same value [41]. Table 3.1 shows the write times to change the conductance value by one level under different internal state according to our memristor model [71]. They are normalized against the time to change the level between 7 and 8. Note that the times are symmetric with respect to

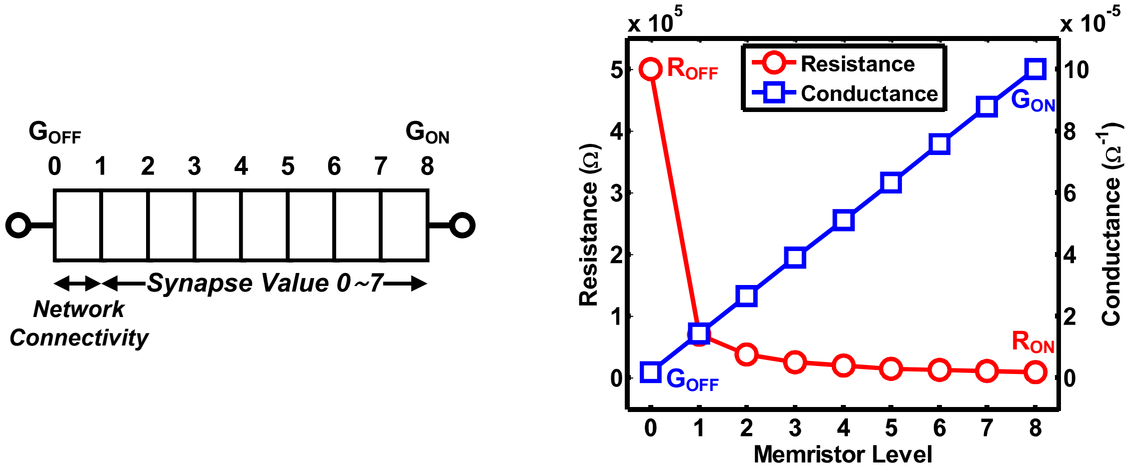


Figure 3.4: Memristor level partitions by equal conductance.

Table 3.1: Normalized write times to change one level of memristor conductance (R_{ON}=10$K\Omega$, R_{OFF}=500$K\Omega$, V_{WRITE}=1.2V).

Level change	$0 \Leftrightarrow 1$	$1 \Leftrightarrow 2$	$2 \Leftrightarrow 3$	$3 \Leftrightarrow 4$	$4 \Leftrightarrow 5$	$5 \Leftrightarrow 6$	$6 \Leftrightarrow 7$	$7 \Leftrightarrow 8$
Time	8205	117	25	10	5	3	2	1

the direction (*e.g.* write time for changing from level 1 to level 0 and vice versa are the same). Also, we have modeled the memristors with the parameters $R_{ON} = 10K\Omega$ and $R_{OFF} = 500K\Omega$ and the read voltage V_{READ} and write voltage V_{WRITE} are considered as 1.2 V for both [71]. The programming time to change level between 0 and 1 is over 8000$\times$ slower than that between 7 and 8 for our memristor. The excessive programming time required to change the state between levels 0 and 1 may notably slow down the overall on-chip learning speed during the training phase. Therefore, we utilize the lowest level to indicate the network connectivity of the crossbar, which allows the network to be fully reconfigurable, and other higher 8 levels to represent the actual synapse value (*i.e.* 3-bit synapse) for a significant training speedup. As an example, if the memristor in the $(i)th$ column and the $(j)th$ row of the crossbar is at level of 0, it means that no connection between the $(j)th$ and the $(i)th$ neurons exist. On the other hand, the memristor level of 4 indicates that the $(j)th$ and $(i)th$ neurons are connected with the synaptic weight of 3.

3.2.3 Memristive Crossbar Array and Cells

Figure 3.5 exhibits the proposed synaptic crossbar array and the CMOS/memristor hybrid synaptic cell. The two switches S_1 and S_2 in the cell are introduced to allow each memristor to be accessible in both the column and row fashion. When the row (column) driver activates a word line, S_1 (S_2) of all cells that lie in the same row (column) are turned on and ready to be accessed. Parallel voltage pulses are generated by the R/W pulse generator and applied to read or write all cells in the row (column) as shown in Figure 3.5.

To read the value from a cell, a fixed positive voltage pulse is applied to the memristor cell, when S_3 and S_4 connect to the pulse generator and the ADC (*i.e.* memristor readout circuit) lines, respectively. The current generated by the cell due

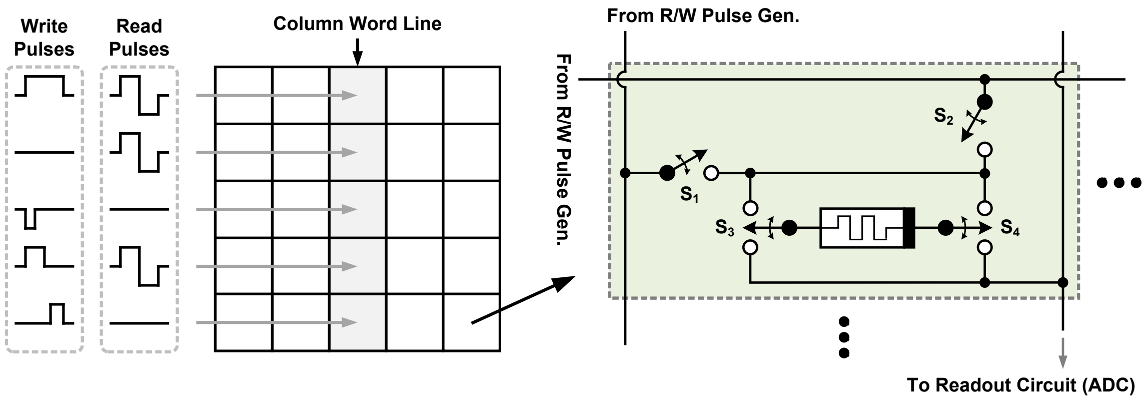


Figure 3.5: Proposed synaptic crossbar array and CMOS/memristor hybrid synaptic cell.

to the applied positive voltage pulse flow out into the ADC line and is converted to a digital value, reflecting the memductance of the cell. Unfortunately, the applied positive pulse disturbs each memductance [23]. Therefore, a flipped (*i.e.* negative) voltage pulse following the positive one is injected to each memristor to restore its memductance, resulting zero net flux injection for the memristor. This is effectively done by connecting S_3 and S_4 to the ADC and the generator lines, respectively. In the write operation, the cells in either one row or column are accessed and incrementally updated in parallel. A write voltage pulse is injected to each memristor cell and its memductance is altered depending on the pulse duration. The write operation latency varies with respect to the value to be written into the cell. It is possible to either increase or decrease the memductance. For the latter, S_3 and S_4 are connected to the ADC and the generator lines, respectively, to effectively apply a negative voltage pulse to the memristor cell.

3.2.4 Digital Pulse Width Modulation for Memristive Synaptic Cell

The proposed DNP uses a parallel write pulse word, consisting of N binary pulses whose durations are different from each other to update either all the pre-synaptic or post-synaptic weights of a given neuron during the training process. The delay line based digital PWM requires both of a large number of delay cells to realize many different pulse widths and a large multiplexer to select one output from these cells, leading to remarkable area and power overheads [64]. Also, the delay cells can be sensitive to PVT variations. This affects the pulse durations and may incur failures in writing the desired values to the memristors. Thus, we leverage a low-cost counter based digital PWM to readily generate a binary pulse with various durations as illustrated in Figure 3.6. The counter records the number of cycles of the clock

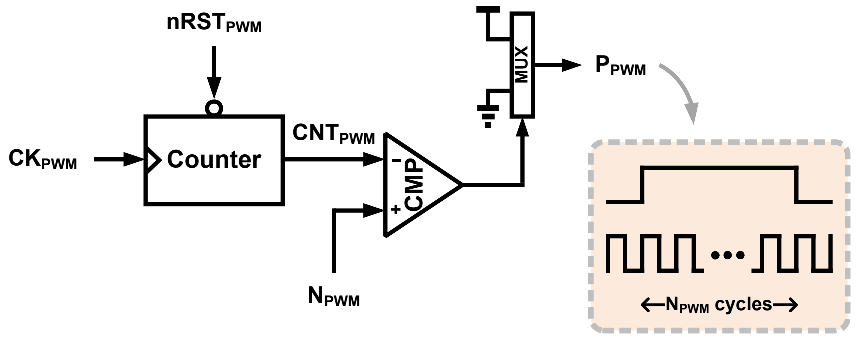


Figure 3.6: Digital pulse width modulator.

signal CK_{PWM} and its output CNT_{PWM} is compared with the desired number of cycles N_{PWM} by the digital comparator. The multiplexer outputs “1” until CNT_{PWM} reaches N_{PWM} and chooses “0” after then. The pulse duration is given by

$$t_{PWM} = N_{PWM} \cdot t_{CKPWM} \tag{3.5}$$

where N_{PWM} and t_{CKPWM} are the desired number cycles and the period, respectively, of the PWM clock CK_{PWM}. Note that CK_{PWM} does not have to be identical to the

DNP operating clock. N_{PWM} is provided by LU, where the amount of synaptic weight change is calculated by the time difference between a pre- and a post-synaptic firing events in accordance with the STDP rule, during the learning stage. CK_{PWM} and N_{PWM} can be straightforwardly configured according to the range of level change and device characteristics of the memristor. The R/W pulse generator includes N digital PWMs to create a read or write pulse word to simultaneously access the memristive synaptic cells in either one column or row.

3.3 Building Block Implementations

3.3.1 Memristor Readout

Figure 3.7 illustrates the proposed memristor readout block that includes a column ADC and a low-resolution ADC array. The column ADC works only for the neuron stage to conduct LIF operations while the low-resolution ADC array is activated during the learning stage to sense each memristor's internal state. In [58], each neuron circuit has its own adder and comparator to integrate all pre-synaptic weights and determine firing activity. It requires N iterations to complete all neu-

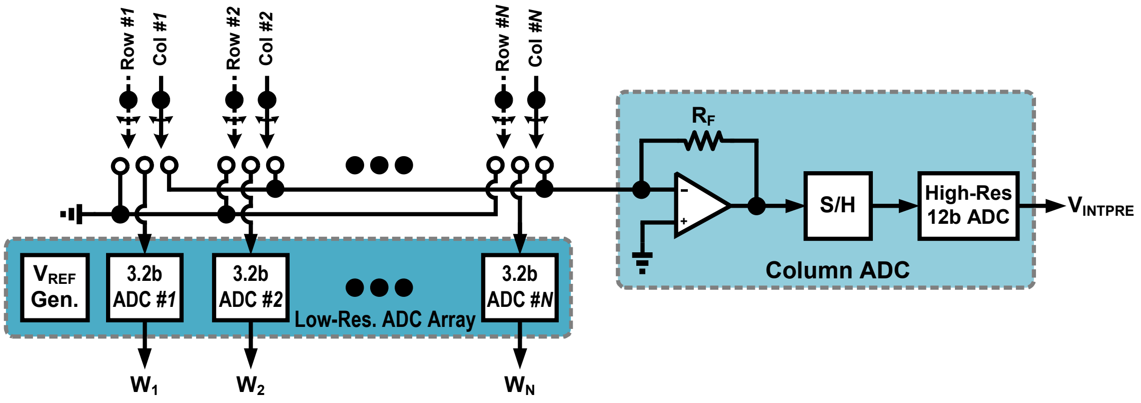


Figure 3.7: Proposed memristor readout block consisting of column ADC and low-resolution ADC array.

rons' LIF tasks. The proposed column ADC, which contains a summing amplifier, a sample-and-hold circuit and a high-resolution ADC, provides significant power and area reductions without degrading overall throughput since it allows a single adder and comparator (LAU) to be shared by all N neurons.

During a LIF operation, the R/W pulse generator injects a pulse word into the corresponding column to read all pre-synaptic weights from the memristor cells. Each current from the cell flows out into the virtual ground and is summated at the negative terminal of the amplifier. The total current is converted to a voltage quantity through the feedback amplifier. The sample-and-hold circuit keeps the voltage and the high-resolution ADC transforms it into a digital value that corresponds to the term $\sum_{j=1}^{M} w_{ji}S_j[t-1]$ in (3.1). In this way, N iterations are enough to complete the LIF operations for all N neurons with only one adder and comparator (LAU).

During the learning stage, before performing a synaptic weight update with a desired value (*i.e.* writing the value into the memristor), we should know the corresponding memristor's current internal state to determine the pulse duration to write the desired synaptic weight. This is because that the required pulse width varies with respect to the current state, notwithstanding writing the same amount of value into the memristor, due to the non-linear device characteristics as demonstrated in Table 3.1. To do this, we consider the array of N low-resolution flash ADCs to read all pre- (post-) synaptic weights of each column (row) in parallel. Also, we leverage the load resistor based sensing scheme, as in Figure 3.3(a), to eliminate the need of involving amplifiers to reduce area and power dissipation. Each flash ADC has 8 comparators to detect one of 9 levels in our memristor cell and a digital logic to encode the output of the comparators. The reference voltage generator shared by N flash ADCs is a resistor string and creates 8 reference voltages for the comparators of each ADC. Importantly, this string does not have equally spaced resistor values since our memristor cell is equally sliced not by resistance but by conductance (see Figure 3.4).

The desired resolution of the column ADC is derived by

$$resolution = \lceil log_2 N + log_2 L \rceil \tag{3.6}$$

where N and L are the numbers of neurons and conductance levels of the memristor cell in the array, respectively. Obviously, the flash ADC architecture is not suitable for a high-resolution ADC because it requires $2^K - 1$ comparators to implement K-bit analog-to-digital conversion, leading to considerable power and area consumptions as K increases. The successive approximation register (SAR) and pipeline ADCs occupy large silicon area stemming from area-consuming passive components. Moreover, the SAR and delta-sigma ($\Delta\Sigma$) ADCs are able to achieve a high-resolution and they may be, unfortunately, limited by a relatively slow conversion rate that is the KHz range. Therefore, to realize a low-cost high-resolution ADC with a moderate conversion speed, we adopt a multiphase voltage controlled oscillator (VCO) based ADC where an analog input alters the VCO frequency and the frequency is measured in digital value by counters [73]. The VCO based ADC is readily implemented with a few digital components such as counters, resulting in a small area overhead. Figure 3.8 shows the block diagram of the VCO based ADC, which consists of a ring VCO, counters, and a tree adder, and the proposed delay cell. We employ a 12-stage ring VCO with the proposed pseudo differential delay cell that is based on an inverter structure with a NMOS current source. The back-to-back inverter in the cell ensures the differential outputs. The VCO operating frequency is adjusted by control the current source (*i.e.* a higher current leads to a higher frequency). The clock phases of each delay cell of the VCO can be exploited to enhance the ADC resolution [34].

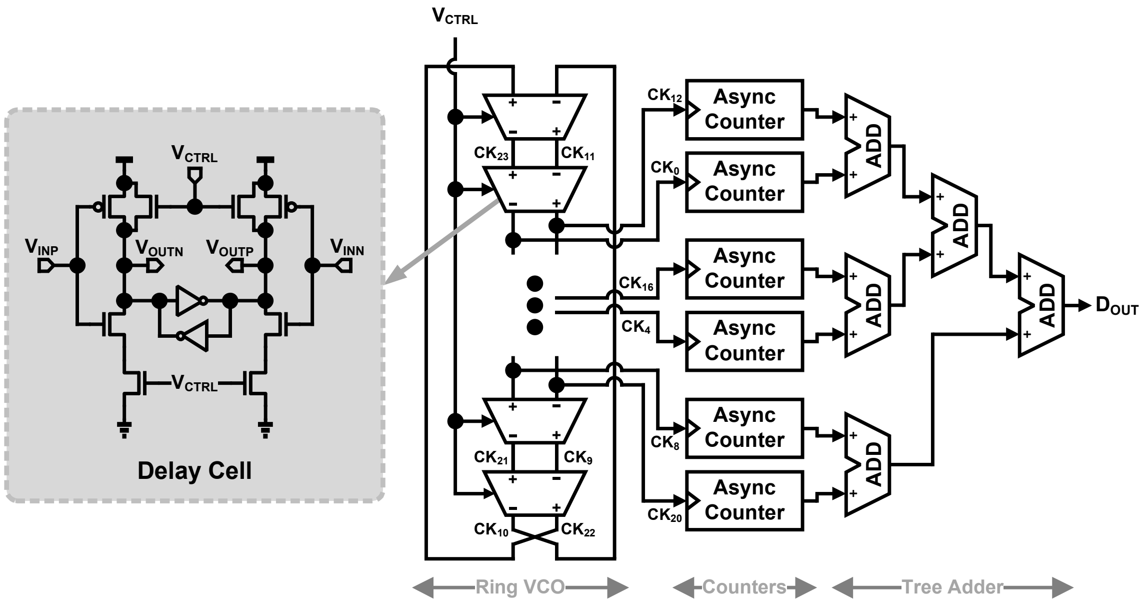


Figure 3.8: Block diagram of VCO based ADC and proposed delay cell.

Generally, the use of more phases results in the higher resolution with higher power and area overheads. In our DNP, six phases (see Figure 3.8) are leveraged to achieve a 12-bit ADC resolution for 256 neurons and 9 levels of the memristor cells under 1 *MHz* conversion rate as will be described in Section 3.4. These clock phases are connected to the respective digital counters whose outputs are summed by the tree adder to obtain the final digital output. The frequency of our VCO is up to 1016 *MHz* and thus 10-bit counter is used to measure the frequency of each clock phase under 1 *MHz* sampling frequency. Also, the VCO usually operates at a few hundred *MHz* and the counters that operate at such a high frequency contribute a large portion of the overall ADC power dissipation [36]. Hence, we employ asynchronous counters to significantly reduce the ADC power.

3.3.2 Neuron and LIF Arithmetic Units

The NU cooperates with LAU to emulate the LIF neuron behavior of (3.1) during the neuron stage. The block diagram of NEs with LAU and the flowchart of NU processing are described in Figure 3.9 and Figure 3.10, respectively. For each

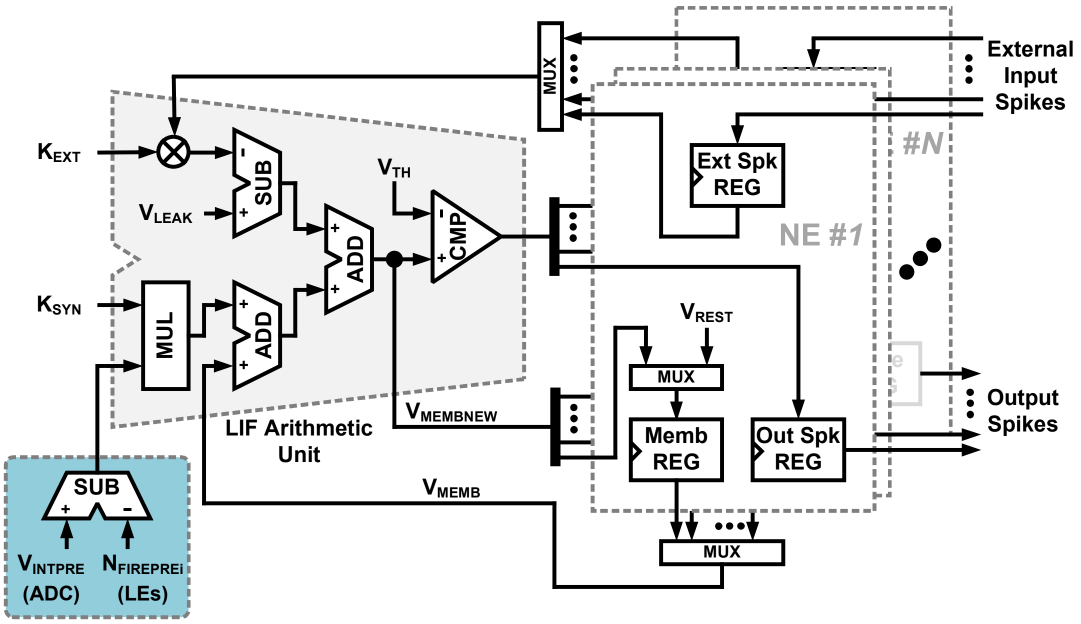


Figure 3.9: Neuron elements with the LIF arithmetic unit.

neuron, the NU makes the R/W pulse generator to create a read pulse word for the corresponding column (dendrite) of the crossbar and sums up all the pre-synaptic weights of the neuron V_{INTPRE} through the column ADC. Importantly, the word contains the read pulses for only fired neurons at the previous time step $t-1$ to save power. Consder a network of 10 neurons as an example. If the $(3)rd$, $(4)th$ and $(5)th$ neurons fired and the other 7 neurons did not fire at $t-1$, only the $(3)rd$, $(4)th$ and $(5)th$ PWMs in the generator produce the read pulse and the other PWMs output

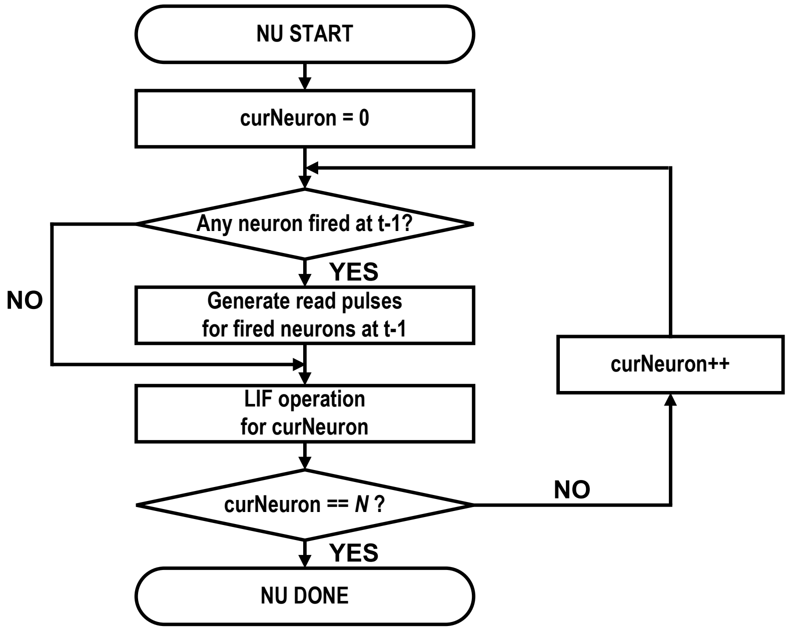


Figure 3.10: Flowchart of the processing of the neuron unit.

"0". Similarly, this column reading and pre-synaptic weight accumulating process is omitted to reduce power dissipation when there is no firing activity in the previous neuron stage (*i.e.* $\forall j,_{0\leq j<N}: S_j[t-1] = 0$ in (3.1)). In this case, V_{INTPRE} is forced to zero due to the term $\sum_{j=1}^{M} w_{ji}S_j[t-1] = 0$ in (3.1) and only the leaky potential V_{LEAK} and the external input spike weight K_{EXT} are considered to obtain the membrane potential.

More importantly, the column ADC output V_{INTPRE} should be adjusted since our synaptic cell includes the connectivity information (*e.g.* memristor level of 4 in the $(j)th$ row and $(i)th$ column of the crossbar indicates the synapse value of 3 between the $(j)th$ and $(i)th$ neurons). Therefore, V_{INTPRE} is subtracted by the number of fired pre-synaptic neurons $N_{FIREPREi}$ (for the $(i)th$ neuron), which is evaluated by the corresponding LE during the synaptic weight update process (*i.e.* the learning stage) at the previous time step $t-1$. The subtractor output $V_{INTPRE} - N_{FIREPREi}$ is added to the corresponding membrane potential V_{MEMB}, and the external input spike weight K_{EXT} and leaky potential V_{LEAK} are added as well. The adder's output $V_{MEMBNEW}$ is compared to the threshold voltage V_{TH} by the digital comparator and the result is sent to the respective NE through the demultiplexer and captured by its output spike register. Meanwhile, $V_{MEMBNEW}$ is sent to the NE and stored in the membrane register when it does not exceed V_{TH}. Otherwise, the register is reset to the resting potential V_{REST} via the multiplexer.

3.3.3 Learning Unit

The LU is designed to conduct the STDP on-chip learning by calculating the amounts of pre- and post-synaptic weight changes, determining the pulse durations to write the desired weights and updating the memristive crossbar array with the values through the R/W pulse generator. Figure 3.11 exhibits the block diagram of the learning elements with the global timer and the two programmable LUTs. The flowchart of LU is shown in Figure 3.12 as well. All LEs contain their own time register to keep the neuron's spike event time and share the register-based LUTs for STDP learning curve and write pulse width for the memristor. The STDP LUT holds several pairs of the spike time difference Δt and synaptic weight change ΔW while the LUT stores the required number of cycles of PWM clock CK_{PWM} to write desired values for the memristor. The design of the pulse width LUT involves important area considerations. For K-bit synapses, a brute-force implementation would require a large number of $\frac{2^K(2^K-1)}{2}$ LUT entries for all possible pairs of the current and target memristor levels. Instead, to reduce area and complexity of the selection logic, we design the LUT in such a way that only the desired cycles of CK_{PWM} for altering the memristor level from the lowest to each target (*i.e.* from level 1 to levels 2, 3, 4, 5, 6, 7 and 8) are stored in $2^K - 1$ entries. With this area-efficient design, the actual

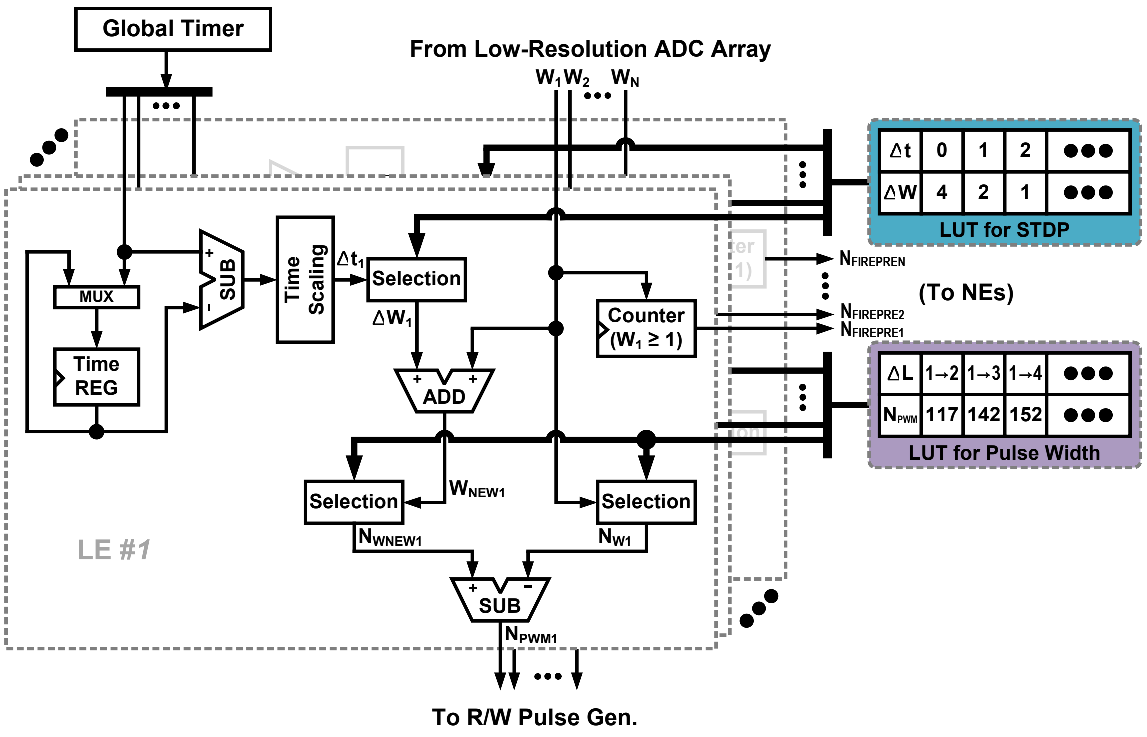


Figure 3.11: Learning elements with global timer and shared LUTs.

pulse duration for a given update is determined by finding the difference between the entry values for the current and target levels. For instance, the number of cycles for increasing the memristor level from 3 to 5 is obtained by subtracting the number of cycles needed for changing the memristor level from 1 to 3 from that for changing from level 1 to 5. As a result, we attain $2^{K-1}\times$ area reduction of the pulse width LUT (*e.g.* 4× reduction in the proposed DNP). In addition, the LU supports a time scaling feature to provide additional programmability in scaling the stored STDP rule. It is implemented with a shift operation of the time differences.

The processing of the learning stage for the entire network is done as follows. The synaptic weight updates are processed by iterating over all fired neurons. To check each neuron's firing activity, the LU checks the output spike buffer, which is

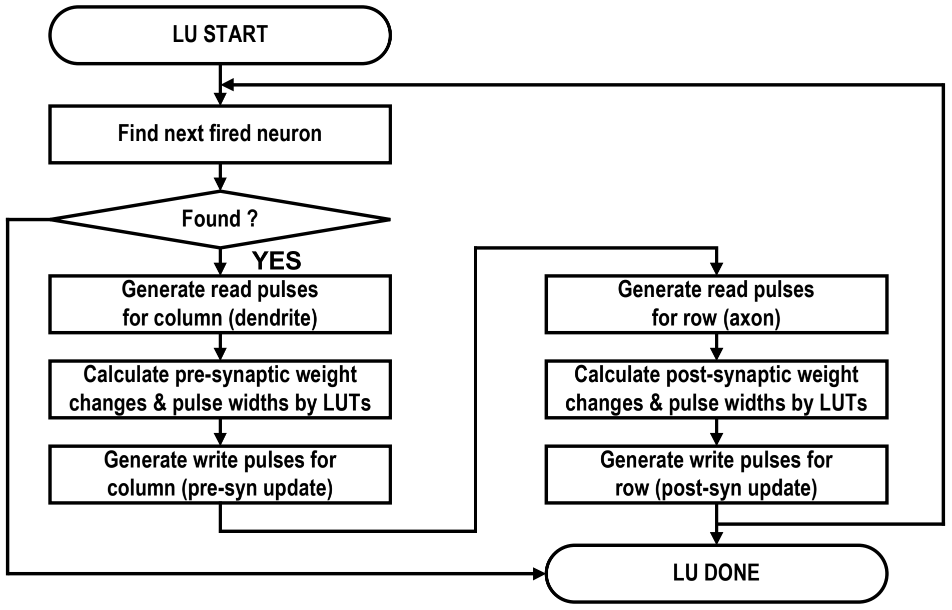

LU START
Find next fired neuron
Found ?
YES
Generate read pulses for column (dendrite)
Calculate pre-synaptic weight changes & pulse widths by LUTs
Generate write pulses for column (pre-syn update)
Generate read pulses for row (axon)
Calculate post-synaptic weight changes & pulse widths by LUTs
Generate write pulses for row (post-syn update)
LU DONE

Figure 3.12: Flowchart of the processing of the learning unit.

filled at the neuron stage, in the corresponding NEs. For each fired neuron, the LU runs the following two back-to-back parallel processes, one for pre-synaptic weights and the other for post-synaptic weight updates. Note that in the absence of neuron firing, the learning stage is skipped. The respective LE for every fired neuron updates its time register with the global timer. Simultaneously, all LEs calculate the scaled time differences $\Delta t_1, \Delta t_2, \cdots, \Delta t_N$ between the global timer and their time register values. This step basically determines the firing time differences between this fired neuron and all other neurons in the network. The synaptic weight changes $\Delta W_1, \Delta W_2, \cdots, \Delta W_N$ for $\Delta t_1, \Delta t_2, \cdots, \Delta t_N$ are selected from the STDP LUT in parallel. Meanwhile, all pre- (post-) synaptic weights $W_1, W_2, \cdots, W_N$ (before update) are obtained from the corresponding column (row) through the R/W pulse generator and low-resolution ADC array and fed into the respective LEs, also in parallel. The pre- (post-) synaptic weights to be written into the respective memristor cells

$W_{NEW1}, W_{NEW2}, \cdots, W_{NEWN}$ are computed by the adder (*i.e.* $\forall i : W_{NEWi} = W_i + \Delta W_i$). Now, W_i and W_{NEWi} correspond to the current and target levels of the $(i)th$ memristor, respectively. Then, each LE concurrently looks up the cycle counts N_{Wi} and N_{WNEWi} from entries associated with W_i and W_{NEWi} from the pulse width LUT. The desired numbers of cycles $N_{PWM1}, N_{PWM2}, \cdots, N_{PWMN}$ to update the pre- (post-) synaptic weight values by $W_{NEW1}, W_{NEW2}, \cdots, W_{NEWN}$, respectively, are determined by subtracting each N_{Wi} from N_{WNEWi} (*i.e.* $\forall i : N_{PWMi} = N_{WNEWi} - N_{Wi}$). Finally, the pulse generator produces the parallel write pulse word with $N_{PWM1}, N_{PWM2}, \cdots, N_{PWMN}$ as in Figure 3.6. When creating the word, N_{PWMi} is set to "0" for $W_i = 0$ since no pre- (post-) synaptic connection exists. Additionally, for negative N_{PWMi} values, the generator inverts the polarity of the pulses and it is effectively done by manipulating the switches of the proposed cell as shown in Figure 3.5. Also, all LEs record the numbers of fired pre-synaptic neurons $N_{FIREPRE1}, N_{FIREPRE2}, \cdots, N_{FIREPREN}$ during the post-synaptic weight update process for the following LIF operations as detailed in Section 3.3.2.

3.4 Implementation of the Neuromorphic Processor and Simulation Results

Except for the memristor nanodevices, the proposed neuromorphic processor has been implemented with a commercial 90 *nm* CMOS technology under the regular supply voltage of 1.2 V. All the digital circuits, which exclude the analog components of the column ADC (*e.g.* ring VCO and summing amplifier) and the low-resolution ADC array, are synthesized with standard cells. The layout that measures 1.45 *mm* × 1.28 *mm* is shown in Figure 3.13. It is worth to note that the memristor crossbar array is defined as an empty macro based on the estimated area. The main clock frequency is 1 *MHz* while the R/W pulse generator operates at 50 *MHz*. Table 3.2 summarizes the key features of the implementation. All the results are obtained from

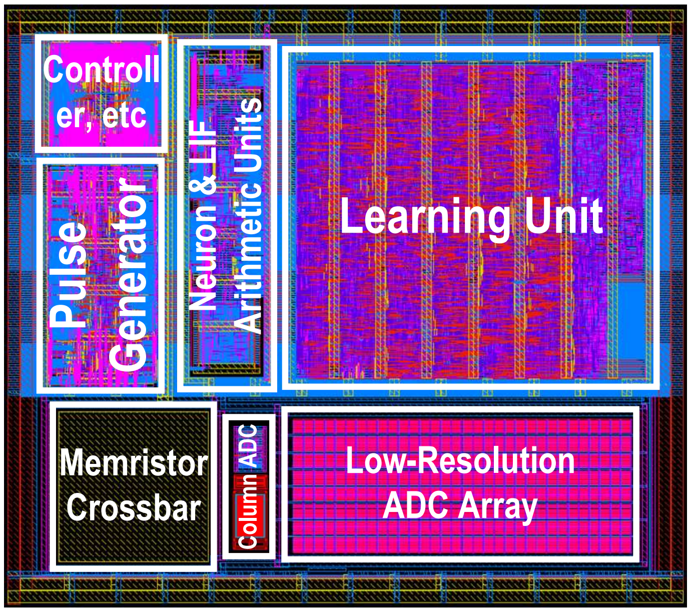


Figure 3.13: Layout of the neuromorphic processor with 256 neurons and 65,536 synapses.

the pre-layout simulations.

3.4.1 Column ADC Performance

To show the performance of the VCO based ADC as proposed in Figure 3.8, we sweep the input voltage from 0.45 V to 1.15 V with a 0.05 V step under a sampling frequency of 1 MHz. The simulated input-to-output characteristic is shown in Figure 3.14(a). The digital output spreads over the range from 1,440 to 5,862 (*i.e.* 12.1-bit resolution) and exhibits a great linear behavior with respect to the input ($R^2 = 0.9964$). Thus, it satisfies the resolution required in (3.6) to serve as a column ADC for the 256 silicon neurons and 256×256 memristive crossbar array with 3-bit synapses in the proposed architecture. As in Figure 3.14(b), we compare the power and area of the ADC with the asynchronous and synchronous counters under various ADC resolutions. Our 12-stage differential ring VCO in Figure 3.8 can

Table 3.2: Neuromorphic processor implementation summary.

Item	**Specification**
Technology	90 *nm* CMOS
Synapse Storage	Memristor
Supply Voltage	1.2 *V*
Main/PWM Operating Frequency	1 / 50 *MHz*
# of Neurons	256
# of Synapses	65,536
Synapse Resolution	3-bits (8-levels)
Neuron Model Parameter Resolution	5-bits
Membrane Potential Resolution	16-bits
Neuron Model	Digital LIF neuron
Learning Rule	On-chip STDP learning
Synaptic Connection Scheme	Fully reconfigurable crossbar
Power Dissipation	6.45 *mW*
Area	1.86 mm^2

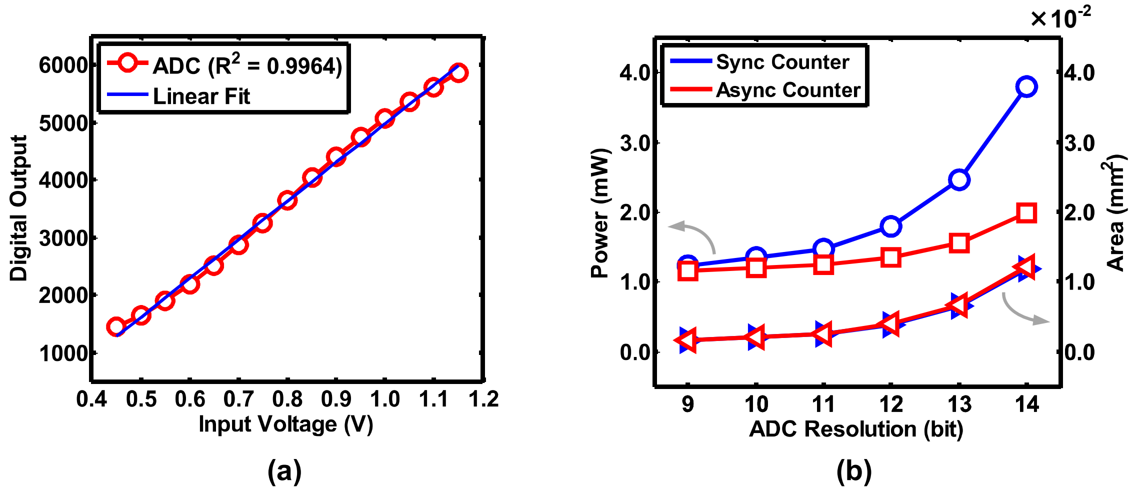


Figure 3.14: Column ADC performance: (a) input-to-output characteristics and (b) power and area as functions of counter type and resolution.

attach up to 24 counters and hence can be leveraged to realize ADCs with a resolution between 9- and 14-bits. Note that the tree adder size also varies according to the number of counters. The power consumptions are measured at the input voltage of 0.8 V, which is the median input of Figure 3.14. The ADC with asynchronous counters is more power efficient than that with synchronous ones while the area remains almost the same as the resolution increases. In the 12-bit resolution, the ADC adopting the asynchronous counter dissipates 24.8% less power with merely 1.6% area overhead compared with the synchronous counter based ADC. Hence, the asynchronous counter is appealing for high-resolution VCO based ADC designs.

3.4.2 Overall Processor Performance

Figure 3.15 demonstrates the overall performance of the proposed neuromorphic processor. The power consumption as functions of the network size, which are eval-

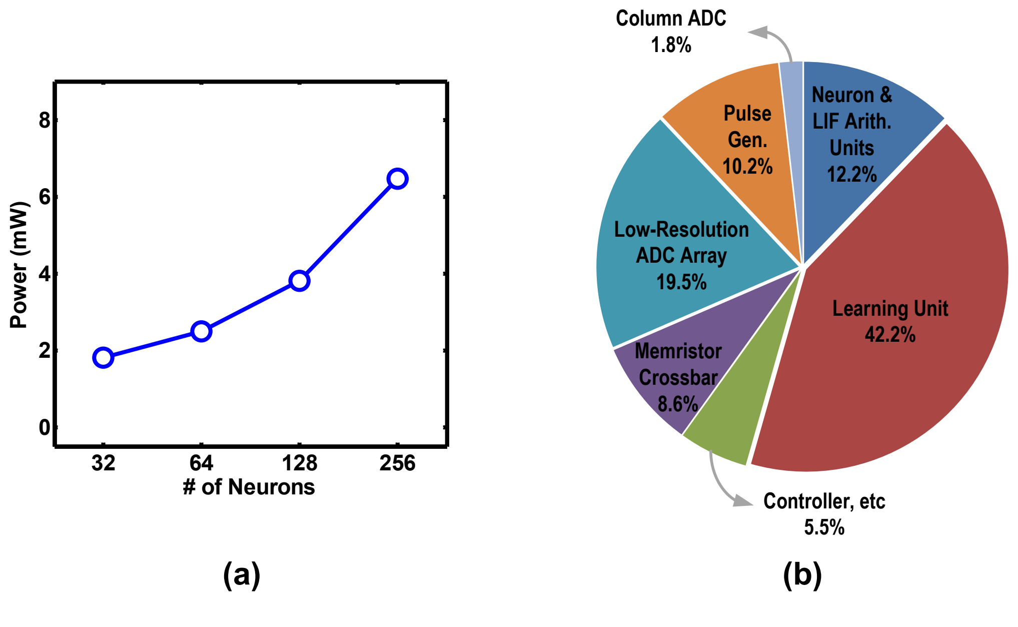


Figure 3.15: Neuromorphic processor performance: (a) power and (b) area breakdown.

uated based on a 90 nm CMOS technology and memristor parameters in [71], is depicted in Figure 3.15(a). The required column ADC resolution is a function of the network size as given in (3.6). (*e.g.* 9-bit column ADC for a 32 neuron design). As the number of integrated neurons N doubles, the chip power increases more slowly than twice. The asynchronous counter based column ADC consumes over 21% of the overall power dissipation but its power does not increase much as the resolution increases (see Figure 3.14(b)). The area breakdown analysis for the neuromorphic processor for a 256 neuron network is illustrated in Figure 3.15(b) as well. The SU, which includes the column ADC, low-resolution ADC array, memristive crossbar, and pulse generator, occupies about 40% of the chip area. Despite of the relatively small area of the memristive crossbar array (8.6%), realizing the parallel access scheme for the multibit memristive crossbar requires integration of several peripherals such as the array of low-resolution ADCs and multiple PWMs. In addition, the concurrent synaptic weight update scheme makes LU to occupy a large area portion (42.2%). Nevertheless, as a return, this parallel scheme expedites the synaptic weight update process significantly. Furthermore, the power consumed by SU reaches 5.16 mW, which is 80% of the entire processor power. Therefore, further optimized low-overhead access scheme can be appealing, as will be explored in the future work.

3.4.3 Application of the Neuromorphic Processor for Character Recognition System

Finally, we conduct a behavior-level digital simulation of the chip to demonstrate the functionality of the neuromorphic processor designed in this section. The behavioral simulation is necessary as gate or transistor level simulation of long training processes requires huge CPU times, making it practically infeasible. To realistically capture the functionality of the designed processor and its dependencies on key hardware design choices, the key network and design features including the digital LIF neuron dynamics, the STDP learning rules, bit-widths to represent the neuron model and synapses in Table 3.2 are fully captured in the behavioral simulation. We specifically consider the case where the proposed DNP is configured to be a two-layer learning network for character recognition as illustrated in Figure 3.16 [15]. The net-

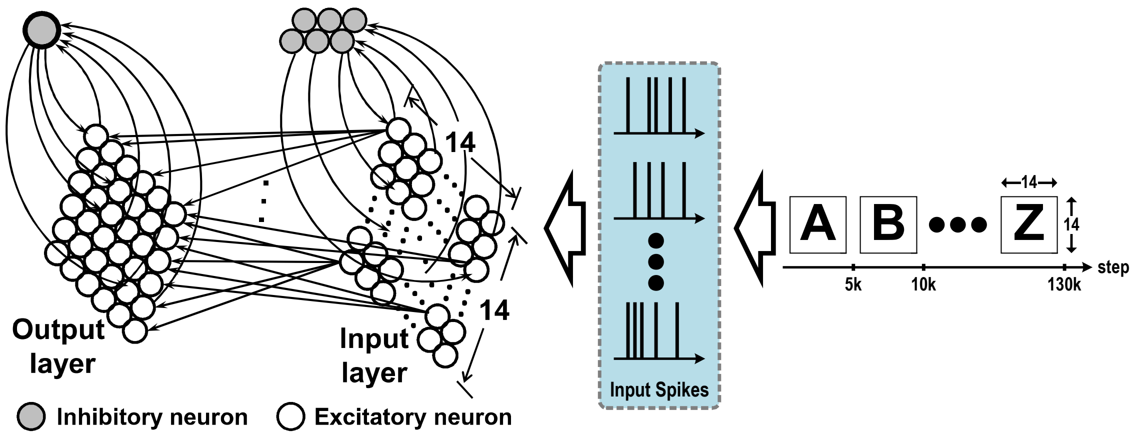


Figure 3.16: Network for character recognition and training for alphabets.

work has an input-and-output layer structure with 232 excitatory and 7 inhibitory neurons and is designed to recognize the alphabets "A" – "Z" by unsupervised learning. The input layer has 196 excitatory neurons, which form a 2 dimensional array. Each excitatory input neuron receives a binary input representing a pixel value in the 14×14 pixel input pattern and projects its output to all excitatory output neurons through plastic synapses. In the input layer, the excitatory neurons project signals to 6 inhibitory neurons which provide negative feedback to modulate the firing frequencies of all excitatory neurons. The output layer consists of 36 excitatory neurons where each of which receives input from all the input excitatory neurons.

Structurally similar to the input layer, one inhibitory neuron is also employed in the output layer to provide strong negative feedback. The inclusion of this inhibitory neuron and these connections implements the winner-take-all (WTA) mechanism, where any firing output neuron activates the inhibitory neuron and thereby prevents other output neurons from firing through negative feedback.

We clock the chip at the frequency of 1 MHz under a fixed supply level of 1.2 V. To train the network, we first convert the training letters, which are composed of a 14×14 pixel map each, to 196 (=14^2) parallel input spike trains to inject into the network as in Figure 3.16. The corresponding input neuron is either silent or active to encode a binary pixel. Then, for each alphabet from "A" to "Z", the corresponding input spike trains are applied to the respective input neurons for 5000 biological time steps. As described here, the network connectivity can be configured by properly programming the memristive crossbar. Note that our reconfigurable processor has 256 neurons. For the configured character recognition chip, Figure 3.17 illustrates

Neuron Index Mapping

Neuron Type	Neuron Index
Input Excitatory	1 ~ 196
Output Excitatory	197 ~ 232
Input Inhibitory	233 ~ 238
Output Inhibitory	239

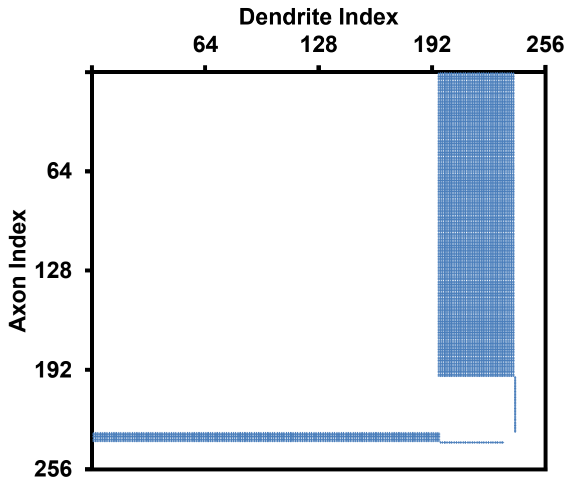


Figure 3.17: Neuron index mapping and synaptic connections of the crossbar array.

the index mapping for the neurons and the synaptic connections of the 256×256 memristive crossbar array (*i.e.* dot plot for memristor levels > 1). The weights of all plastic synapses are random values before the training. With any input pattern, the net input received by each excitatory output neuron can be thought as the inner product of a random weight vector and a signal vector representing the activities of the excitatory input neurons. The weight vector of each output neuron corresponds to its receptive field, which describes the input pattern whose presence leads to excitation of the corresponding output neuron. The network reshapes receptive fields of some excitatory output neurons to memorize each alphabet during the training such that they receive strong excitatory signal and emit spikes with the presence of corresponding input pattern.

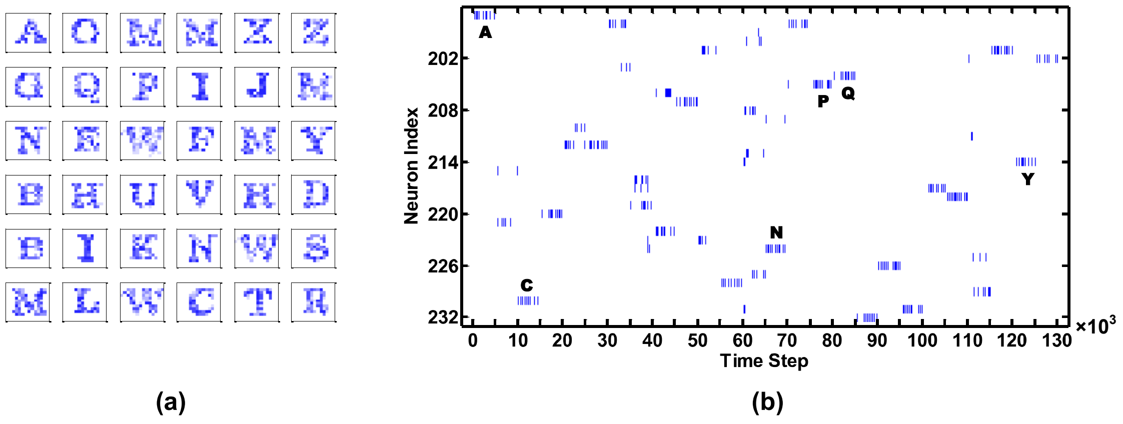


Figure 3.18: Learning results for network: (a) receptive fields after training and (b) spike rasters for output neurons.

To demonstrate the function of the proposed processor, we show the simulation results of the learning network for the alphabet training in Figure 3.18. The receptive fields of the network after the training are shown in Figure 3.18(a). As can be seen, the receptive fields are well shaped by the training in the sense that every letter from "A" to "Z" appears once at least in the fields. This implies that during the recognition phase the presence of a letter is expected to excite at least one output neuron whose receptive field closely reassembles the presented letter, signifying the correct recognition of the letter. The spike rasters for the 36 output excitatory neurons, which correspond to the neuron indices from 197 to 232, respectively, during the training process is plotted in Figure 3.18(b). Due to the WTA network configuration, each input pattern has the tendency to one or few output neurons to fire and all other output neurons are inhibited through the negative feedback. For instance, during the biological time steps from 1 to 5000, the letter "A" is presented in training. The (197)th neuron's receptive field is trained to resemble "A" and this neuron is the only output neuron who actively fires in this period. In short, the (197)th neuron is the winner when the alphabet "A" is presented. For the training of some letters such as "B", there may be a small number of winners in the WTA mechanism and, as a result, more than one output neuron are trained as seen two "B" shaped letters in the receptive fields. The results shown in Figure 3.18(b) verify the correct functioning of the designed WTA scheme. In Figure 3.18(b), we mark the spike rasters of a few output neurons which should be active with the presence of the corresponding training alphabet. For ease of visualization, only a subset of these neurons are marked in the figure.

3.5 Summary

In this section, we have proposed a scalable digital neuromorphic processor architecture for large scale integration of spiking neurons. A novel multilevel memristive synaptic crossbar design is presented to allow for high-density synaptic storage and flexible access. Moreover, the lowest conductance level of each memristor is used to arbitrarily configure the network connectivity with very low overhead, which also significantly improves the synaptic weight update performance by reducing the write time of the memristive crossbar. The proposed VCO based column ADC design reduces the silicon overhead required for LIF operations and is amenable to integration due to its digital implementation style. Implemented in a 90 nm CMOS process, our design with 256 digital neurons with learning circuits and $64K$ synapses is evaluated to occupy an area of 1.86 mm^2 and dissipate a power of 6.45 mW under a supply voltage of 1.2 V. Furthermore, we have validated the functionality of the proposed architecture through the behavioral digital simulation for the case of a character recognition system with unsupervised learning.

4. ENERGY EFFICIENT APPROXIMATE ARITHMETIC

4.1 Proposed Approximate Adder

Our main focuses and key contributions in the design of approximate arithmetic units include 1) a significant reduction of the error rate by the carry-skip scheme enabling carry speculation in a parallel manner and its application to the adder and comparator design, 2) complete error rate analysis of the proposed arithmetic units and 3) a very low-cost error magnitude reduction scheme without additional clock cycle scheme for the proposed adder. In this section, the proposed approximate adder design is presented while the proposed approximate comparator is discussed in the next section.

4.1.1 Approximate Adder Architecture

Denote the two inputs of the adder A and B, and the (i)th least significant bits (LSBs) by a_i and b_i, respectively. In addition, the propagate (p_i), generate (g_i), kill (k_i), and carry (c_i) signals of the (i)th bit position are defined by

$$g_i = a_i b_i,\;\; k_i = \bar{a}_i \bar{b}_i,\;\; p_i = a_i \oplus b_i$$
$$c_i = \begin{cases} 1 & \textit{if}\;\; g_i = 1 \\ 0 & \textit{if}\;\; k_i = 1 \\ c_{i-1} & \textit{if}\;\; p_i = 1 \end{cases} \tag{4.1}$$

where c_{i-1} is the carry of the (i-1)th bit position. Briefly, the adder outputs the carry c_i when g_i=1 or k_i=1 independently of c_{i-1}, otherwise, it propagates c_{i-1} to c_i.

Figure 4.1 shows the block diagram of the proposed approximate n-bit adder, which is divided into several k-bit sized blocks. Each block contains a k-bit sub-

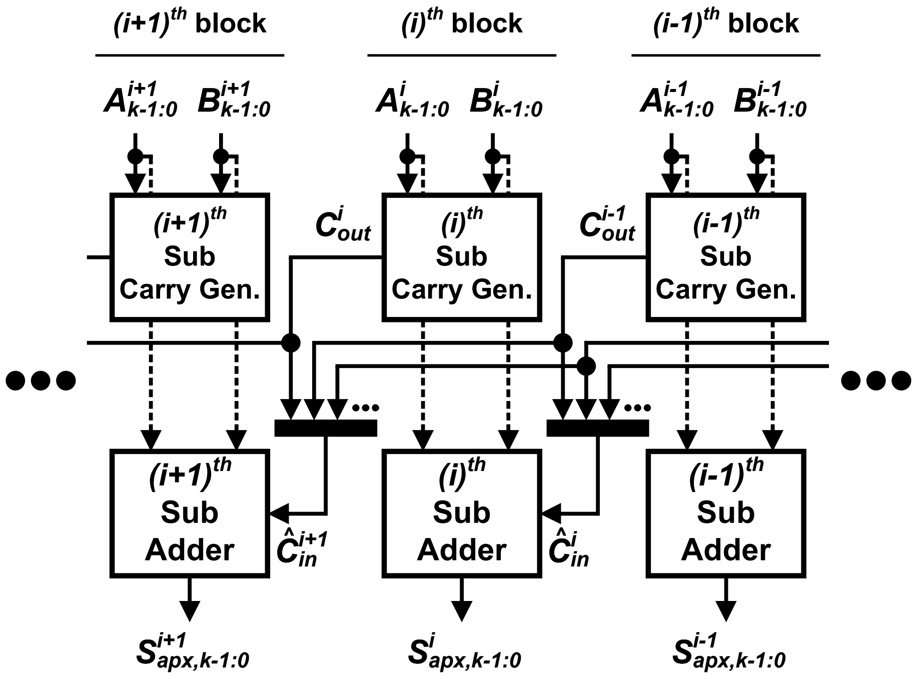


Figure 4.1: Block diagram of the proposed approximate adder.

adder and a k-bit sub-carry generator, which create a partial summation and a partial carry-out signal, respectively. The n-bit adder has $m = \lceil \frac{n}{k} \rceil$ blocks. Also, as in Figure 4.1, the k-bit inputs of the (i)th block are represented by $A^i_{k-1:0}$ and $B^i_{k-1:0}$, and the partial summation result is indicated by $S^i_{apx,k-1:0}$. Note that the sub-adders could be implemented by any traditional accurate adders such as ripple-carry adder (RCA) and carry-lookahead adder (CLA). At the beginning of an addition operation, all the sub-carry generators simultaneously create the partial carry-out signals ($\cdots$, C^{i+1}_{out}, C^{i}_{out}, C^{i-1}_{out}, $\cdots$) using only their k-bit inputs. Then, the sub-adders' carry-in signals ($\cdots$, $\widehat{C}^{i+1}_{in}$, $\widehat{C}^{i}_{in}$, $\widehat{C}^{i-1}_{in}$, $\cdots$) are also concurrently speculated from the v ($\geq$2) preceding k-bit sub-carry generators with a multiplexer. Finally, the sub-adders work with the speculated carries and produce the partial summations ($\cdots$, $S^{i+1}_{apx,k-1:0}$, $S^{i}_{apx,k-1:0}$, $S^{i-1}_{apx,k-1:0}$, $\cdots$). Therefore, the critical path delay of the proposed approximate adder t_{apx} is derived by

$$t_{apx} = t_{sa} + \lceil log_2 v \rceil t_{mux} + t_{scg} \tag{4.2}$$

where t_{sa}, t_{mux} and t_{scg} are the delays of the sub-adder, a two-input multiplexer, and the sub-carry generator, respectively. The delay is based on a multilevel tree structure of two-input multiplexers. Note that the multiplexer delay is negligible if k is large.

The proposed carry prediction works as follows. When all the propagate signals of the (i-1)th block are true, the carry-out of a large number of preceding blocks are required for more accurate carry prediction for the (i)th sub-adder. Thus, we utilize carry-skip to speculate the carry as depicted in Figure 4.2, which is an example of the adder with k=6 and v=3. This carry-skip scheme is particularly more advantageous

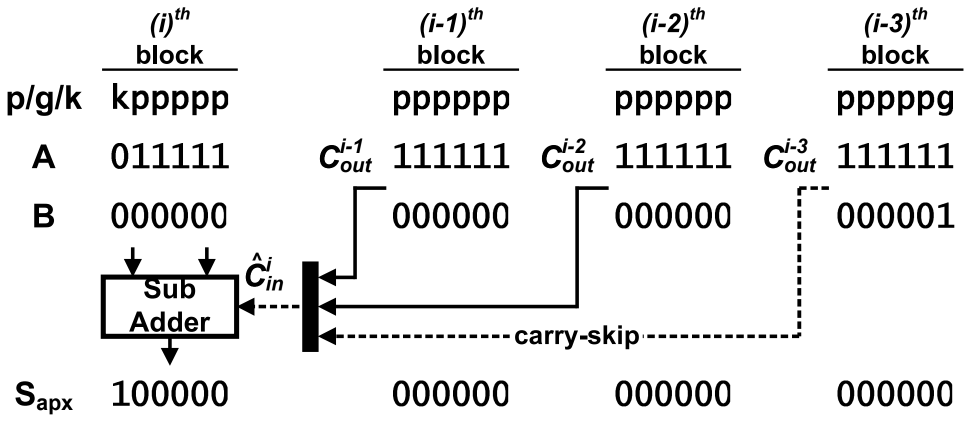


Figure 4.2: Proposed carry prediction using parallel carry-skip (k=6, v=3).

over the alternative approach of cascading several sub-carry generators [74], which could appreciably increase the critical path delay when k is large. In order to obtain the (i)th carry-in $\widehat{C}_{in}^{i}$, the multiplexer selects C_{out}^{u} where $i-v \leq u < i$ if any propagate signal of the (u)th block is false as in Figure 4.2. If all the propagate signals of the v preceding blocks are true, it chooses C_{out}^{i-v}. Hence, $\widehat{C}_{in}^{i}$ is expressed by

$$\begin{aligned}\widehat{C}_{in}^{i} =& \overline{P_{k-1:0}^{i-1}} C_{out}^{i-1} + P_{k-1:0}^{i-1} \overline{P_{k-1:0}^{i-2}} C_{out}^{i-2} + \cdots + \\ & \prod_{j=i-1}^{i-v+2} P_{k-1:0}^{j} \overline{P_{k-1:0}^{i-v+1}} C_{out}^{i-v+1} + \prod_{j=i-1}^{i-v+1} P_{k-1:0}^{j} C_{out}^{i-v} \\ & where \quad P_{k-1:0}^{i} = \prod_{j=0}^{k-1} p_{j}^{i}\end{aligned} \tag{4.3}$$

In (4.3), C_{out}^{i} is the carry-out signals of the (i)th block and p_{j}^{i} is the propagate signal of the (j)th bit position of the (i)th block. Additionally, the carry-out of the (i)th block is given by

$$C_{out}^{i} = g_{k-1}^{i} + g_{k-2}^{i} p_{k-1}^{i} + \ldots + g_{0}^{i} \prod_{j=1}^{k-1} p_{j}^{i} \triangleq G_{k-1:0}^{i} \tag{4.4}$$

where g_{j}^{i} is the generate signal at (j)th bit position of the (i)th block.

By adopting the carry-skip scheme, the proposed adder is able to enhance the carry prediction accuracy at the cost of multiplexer delay. Generally, a larger number of preceding sub-carry generators can be used to further improve the accuracy of carry prediction, at a low-cost of one multiplexer delay per each included generator.

4.1.2 Error Rate Analysis

The carry prediction error of the proposed adder occurs when a carry propagation chain has a length greater than kv. In other words, if all the propagate signals of more than v consecutive blocks are true and a carry is generated in the preceding block, then the carry prediction is incorrect. Assuming that the adder inputs A and B are bitwise independent, then the propagate and generate signals are bitwise independent as well. We denote the event that the carry-in prediction of the (i)th sub-adder is mistaken due to a carry propagation path of a length between kv and $k(v+1)-1$ by E^i_{cin}

$$E^i_{cin} = P^{i-1}_{k-1:0} P^{i-2}_{k-1:0} \cdots P^{i-v}_{k-1:0} G^{i-v-1}_{k-1:0} \tag{4.5}$$

where $P^i_{k-1:0}$ and $G^i_{k-1:0}$ is defined in (4.3) and (4.4), respectively and the probability of the event is given by

$$\begin{aligned}\mathbf{P}(E^i_{cin}) =& \mathbf{P}(P^{i-1}_{k-1:0} P^{i-2}_{k-1:0} \cdots P^{i-v}_{k-1:0} G^{i-v-1}_{k-1:0}) \\ =& \mathbf{P}(P^{i-1}_{k-1:0})\mathbf{P}(P^{i-2}_{k-1:0}) \cdots \mathbf{P}(P^{i-v}_{k-1:0})\mathbf{P}(G^{i-v-1}_{k-1:0})\end{aligned} \tag{4.6}$$

In (4.6), $\mathbf{P}(P^i_{k-1:0})$ [$= \mathbf{P}(P^{i-1}_{k-1:0}) = \cdots$] and $\mathbf{P}(G^i_{k-1:0})$ [$= \mathbf{P}(G^{i-1}_{k-1:0}) = \cdots$] are given by

$$\begin{aligned}\mathbf{P}(P^i_{k-1:0}) &= \mathbf{P}(\prod_{j=0}^{k-1} p^i_j) = \prod_{j=0}^{k-1} \mathbf{P}(p^i_j) = \frac{1}{2^k} \\ \mathbf{P}(G^i_{k-1:0}) &= \mathbf{P}(g^i_{k-1} + g^i_{k-2}p^i_{k-1} + \cdots + g^i_0 \prod_{j=1}^{k-1} p^i_j) \\ &= \mathbf{P}(g^i_{k-1}) + \mathbf{P}(g^i_{k-2}p^i_{k-1}) + \cdots + \mathbf{P}(g^i_0 \prod_{j=1}^{k-1} p^i_j) \\ &= \frac{1}{4} + \frac{1}{4}\cdot\frac{1}{2} + \cdots + \frac{1}{4}\cdot\frac{1}{2^{k-1}} = \frac{1}{2}\left(1 - \frac{1}{2^k}\right)\end{aligned} \tag{4.7}$$

where g^i_{k-1}, $g^i_{k-2}p^i_{k-1}$, $\cdots$, $g^i_0 \prod_{j=1}^{k-1} p^i_j$ are mutually exclusive. The proposed adder produces an error if any error event E^i_{cin} occurs for any of the sub-adders except for the v+1 least significant ones. Note that the (0)th, (1)st, $\cdots$, (v)th sub-adders always have the correct carry-in signals in our design. Thus, the overall error rate of the proposed adder under random inputs is expressed by

$$\mathbf{P}_{\mathbf{err}}(n,k,v) = \mathbf{P}(E_{cin}^{\lceil \frac{n}{k} \rceil - 1} + E_{cin}^{\lceil \frac{n}{k} \rceil - 2} + \cdots + E_{cin}^{v+2} + E_{cin}^{v+1}) \tag{4.8}$$

By the inclusion-exclusion principle [5], it is given by

$$\begin{aligned}\mathbf{P}_{\mathbf{err}}(n,k,v) = & \sum_{v<i<m} \mathbf{P}(E_{cin}^{i}) \\ & - \sum_{v<i_1<i_2<m} \mathbf{P}(E_{cin}^{i_2} E_{cin}^{i_1}) \\ & + \sum_{v<i_1<i_2<i_3<m} \mathbf{P}(E_{cin}^{i_3} E_{cin}^{i_2} E_{cin}^{i_1}) \\ & - \cdots + (-1)^{m-v} \mathbf{P}(E_{cin}^{m-1} E_{cin}^{m-2} \cdots E_{cin}^{v+1}) \\ & \textit{where} \;\; m = \lceil n/k \rceil \end{aligned} \tag{4.9}$$

Once E_{cin}^{i} occurs, E_{cin}^{i-1}, E_{cin}^{i-2}, $\cdots$, E_{cin}^{i-v} can not do. This is because that under this case the carry propagate chain lengths for the (i-1)th, $\cdots$, (i-v)th sub-adders become less than kv due to $P_{k-1:0}^{i-2} = \cdots = P_{k-1:0}^{i-v} = G_{k-1:0}^{i-v-1} = 1$ and thus the carry speculations for these sub-adders are always correct. In short, $\mathbf{P}(E_{cin}^{i_r} \cdots E_{cin}^{i_1}) = 0$ if $\exists q : i_q - i_{q-1} \leq v$ where $v < i_1 < \cdots < i_r < \lceil \frac{n}{k} \rceil$. Then, we can rewrite (4.9) to yield

$$\begin{aligned}& \mathbf{P}_{\mathbf{err}}(n,k,v) \\ & = \sum_{r=1}^{m-v-1} (-1)^{r+1} \left(\sum_{\substack{v<i_1<\cdots<i_r<m, \\ \forall q: i_q - i_{q-1} > v}} \mathbf{P}(E_{cin}^{i_r} \cdots E_{cin}^{i_1}) \right) \\ & \quad \textit{where} \;\; m = \lceil n/k \rceil \end{aligned} \tag{4.10}$$

$E_{cin}^{i_r}, E_{cin}^{i_{r-1}}, \cdots E_{cin}^{i_1}$ are independent if $\forall q : i_q - i_{q-1} > v$. Therefore, by putting (4.6), (4.7) and (4.10) together, the overall error rate for the adder under random inputs is

$$\begin{aligned}
&\mathbf{P_{err}}(n,k,v)\\
&= \sum_{r=1}^{m-v-1} (-1)^{r+1} \left(\sum_{\substack{v<i_1<\cdots<i_r<m,\\ \forall q: i_q - i_{q-1}>v}} \mathbf{P}(E_{cin}^{i_r}) \cdots \mathbf{P}(E_{cin}^{i_1}) \right)\\
&= \sum_{r=1}^{m-v-1} (-1)^{r+1} \left(\sum_{\substack{v<i_1<\cdots<i_r<m,\\ \forall q: i_q - i_{q-1}>v}} \left(\frac{1}{2^{kv+1}} \left(1 - \frac{1}{2^k}\right)\right)^r \right)\\
&\qquad \qquad \textit{where} \quad m = \lceil n/k \rceil
\end{aligned} \tag{4.11}$$

4.1.3 Error Magnitude Reduction Scheme

In addition to error rate, another important metric to evaluate approximate adders is error significance, which should be minimized and is defined by the ratio of the error magnitude to the correct summation result as follows [53]

$$\textit{error significance} = \left| \frac{S_{apx} - S_{cor}}{S_{cor}} \right| \tag{4.12}$$

where S_{apx} and S_{cor} are the approximate and correct outputs for given inputs.

Figure 4.3 depicts the block diagram of the proposed error magnitude reduction and one example of its operation in case of k=8 and v=2. In the example, since all the propagate signals of the (i)th and (i-1)th blocks are true, the carry-in for the (i+1)th sub-adder is speculated to "0" although the correct one is "1" due to $C_{out}^{i-2} = 1$. Then, the error significance is $\frac{1}{2^7}$. Note that it could reach $\frac{1}{2}$ for the worst case inputs of $A_{7:0}^{i+1} = 00000001$ and $B_{7:0}^{i+1} = 00000000$. To reduce the amount of error, the proposed adder forces all the output bits of the (i)th and the (i-1)th

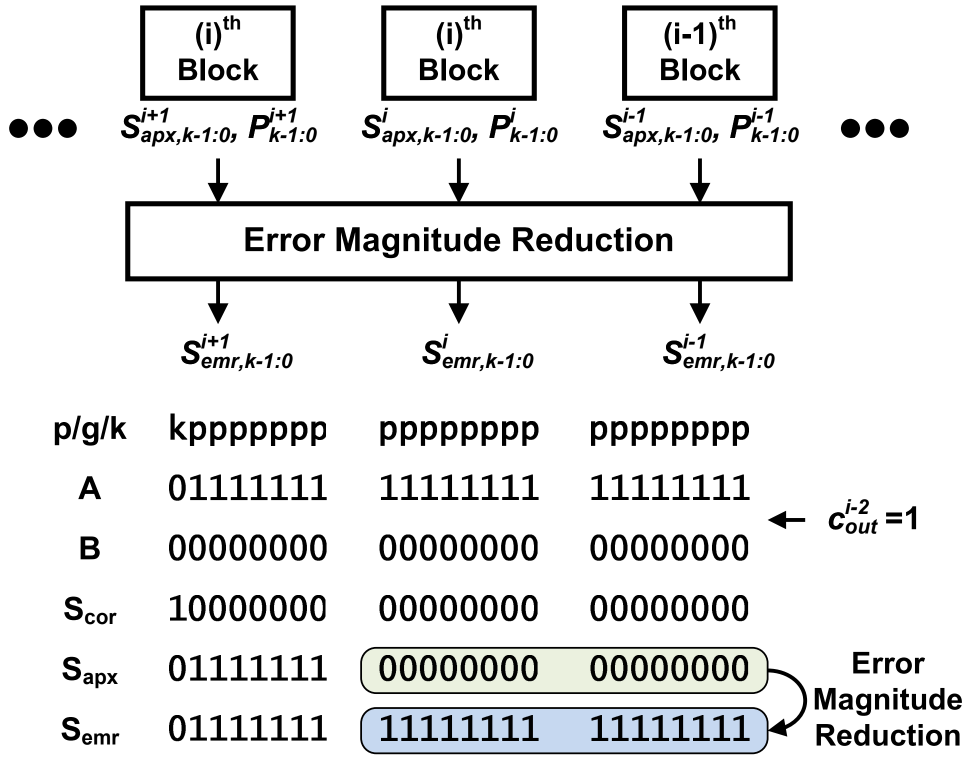

(i)th Block
(i)th Block
(i-1)th Block
$S^{i+1}_{apx,k-1:0}$, $P^{i+1}_{k-1:0}$
$S^{i}_{apx,k-1:0}$, $P^{i}_{k-1:0}$
$S^{i-1}_{apx,k-1:0}$, $P^{i-1}_{k-1:0}$
Error Magnitude Reduction
$S^{i+1}_{emr,k-1:0}$
$S^{i}_{emr,k-1:0}$
$S^{i-1}_{emr,k-1:0}$
p/g/k kppppppp pppppppp pppppppp
A 01111111 11111111 11111111
B 00000000 00000000 00000000
S_{cor} 10000000 00000000 00000000
S_{apx} 01111111 00000000 00000000
S_{emr} 01111111 11111111 11111111
c^{i-2}_{out} =1
Error Magnitude Reduction

Figure 4.3: Block diagram of the error magnitude reduction and an example of its operation (k=8, v=2).

sub-adders to “1” when $P^i_{k-1:0} = P^{i-1}_{k-1:0} = 1$. The reduction can be implemented by ORing each partial summation (*i.e.* $S^i_{apx,k-1:0}$ and $S^{i-1}_{apx,k-1:0}$) and the product of the propagate signals (*i.e.* $P^i_{k-1:0}P^{i-1}_{k-1:0}$). It allows the error significance to be reduced by $\frac{1}{2^{2k}}$. As a result of the reduction, the adder finally produces the error reduced output of $S^i_{emr,k-1:0} = S^{i-1}_{emr,k-1:0} = 11111111$ and the error significance decreases from $\frac{1}{2^7}$ to $\frac{1}{2^{23}}$. It is worth mentioning that the error magnitude reduction always produces the exact right results when $C^{i-2}_{out} = 0$. Consequently, the worst case error magnitude is reduced from 2^{n-k} to $2^{n-k(v+1)}$ through the use of error magnitude reduction.

4.2 Proposed Approximate Comparator

4.2.1 Approximate Comparator Architecture

Basically, a comparator determines the larger of two inputs A and B, and can be implemented by using a subtraction. After subtracting two inputs $A - B$, a comparison is readily done by checking the sign bit (*i.e.* MSB) of the result. In short, $A < B$ when the MSB = 1, otherwise $A \geq B$. Note that subtraction is achieved by addition of 2's complement (*i.e.* $A - B = A + \overline{B} + 1$) and requires an additional inverting operation for one of two inputs. However, the use of traditional adders such as an RCA may incur timing and energy overheads since the MSB of the addition would be needed to produce the final comparator output. Again, due to the fact that the targeted neuromorphic computing applications have built-in resilience to arithmetic errors as shown in the later part of the section, we exploit the same idea of the parallel carry-skip scheme to improve the timing and energy efficiency of the comparator.

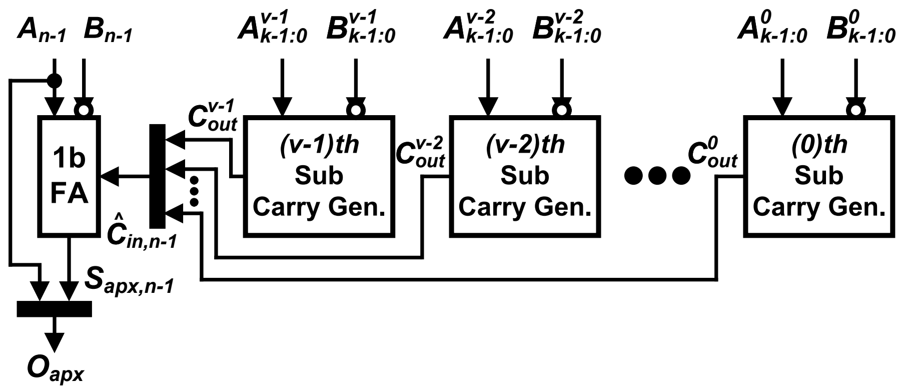


Figure 4.4: Block diagram of the proposed approximate comparator.

Figure 4.4 illustrates the block diagram of the proposed approximate comparator. The n-bit comparator consists of a 1-bit full adder and v ($\geq$2) k-bit sub-carry generators that are identical to the ones in the proposed adder. In Figure 4.4, the k-bit inputs of the (i)th sub-carry generator $A^i_{k-1:0}$ and $B^i_{k-1:0}$ correspond to $A_{n-k(v-i-1)-2:n-k(v-i)-1}$ and $B_{n-k(v-i-1)-2:n-k(v-i)-1}$, respectively. It is worth to note that the proposed approximate comparator exploits only kv+1 MSBs of the n-bit inputs, resulting in area and power reductions. Importantly, the input B is inverted to achieve subtraction operation. Since implementing 2's complement necessitates an additional incrementor, we employ 1's complement to further reduce area and energy with sacrificing an error rate, but still achieving a very low error rate. The full adder generates the sign bit $S_{apx,n-1}$ (MSB output) of the subtraction between the two inputs by leveraging the speculated carry-in signal $\widehat{C}_{in,n-1}$ and the MSB of the two inputs. The speculated carry-in signal is obtained in the same parallel way by the v sub-carry generators according to (4.3) and (4.4). When the two inputs have the different signs ($i.e.$ $A_{n-1} \oplus B_{n-1} = 1$), the comparison result is readily obtained by the input MSB without the full adder. Therefore, the output multiplexer selects the MSB of the input A_{n-1} if the signs of two inputs are different from each other, otherwise, it chooses the full adder output $S_{apx,n-1}$. Therefore, the comparator output O_{apx} is expressed by

$$\begin{aligned} O_{apx} &= \left(A_{n-1} \oplus B_{n-1}\right) A_{n-1} + \left(\overline{A_{n-1} \oplus B_{n-1}}\right) S_{apx,n-1} \\ where \quad S_{apx,n-1} &= A_{n-1} + \overline{B_{n-1}} + \widehat{C}_{in,n-1} \end{aligned} \tag{4.13}$$

Then, the critical path delay of the proposed approximate comparator $t_{apx,cmp}$ with the multilevel tree based multiplexer is derived to be

$$t_{apx,cmp} = t_{fa} + (\lceil log_2 v \rceil + 1)\, t_{mux} + t_{scg} \tag{4.14}$$

where t_{fa}, t_{mux} and t_{scg} are the delays of the full adder, the two-input multiplexer, and the sub-carry generator, respectively. In the proposed comparator, the 1-bit full adder delay is negligible and the multiplexer delay is also negligible if k is large.

4.2.2 Error Rate Analysis

The proposed comparator fails when the signs of the inputs are different from each other as well as the carry prediction for the full adder is incorrect. Figure 4.5 illustrates an example of the proposed comparator configuration with n=16, k=4, v=2 to effectively explain the carry prediction error for the full adder. The input B is inverted and it includes an unused sub-carry generator block that has the 7 LSBs as the inputs and its carry-in signal $C_{in,0}$ is always "1" due to the 2's complement (*i.e.* $-B = \overline{B} + 1$). Note that this block is never used for the comparisons in our design. The carry speculation is incorrect when all the propagate signals in the v (in this case two) sub-carry generators used for carry prediction are true and $C_{out}^{un} = 1$. The latter condition is true if a carry is generated by the unused block or $C_{in,0}(= 1)$ is

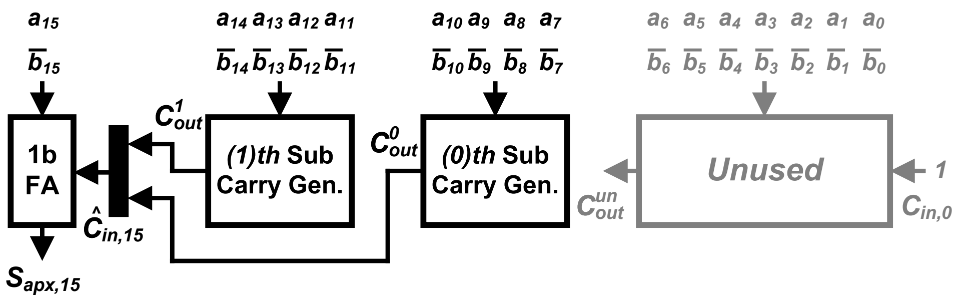

a_{15}
$\overline{b}_{15}$
$a_{14}\, a_{13}\, a_{12}\, a_{11}$
$\overline{b}_{14}\overline{b}_{13}\overline{b}_{12}\overline{b}_{11}$
$a_{10}\, a_9\, a_8\, a_7$
$\overline{b}_{10}\overline{b}_9\, \overline{b}_8\, \overline{b}_7$
$a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0$
$\overline{b}_6\, \overline{b}_5\, \overline{b}_4\, \overline{b}_3\, \overline{b}_2\, \overline{b}_1\, \overline{b}_0$
1b FA
C_{out}^1
(1)th Sub Carry Gen.
C_{out}^0
(0)th Sub Carry Gen.
Unused
1
C_{out}^{un}
$C_{in,0}$
$\hat{C}_{in,15}$
$S_{apx,15}$

Figure 4.5: Example of the comparator configuration (n=16, k=4, v=2).

propagated through the unused block. It is important to note that we should consider $C_{in,0}$ propagation since the proposed comparator adopts 1's complement, instead of 2's complement, for the subtraction. We assume that the comparator inputs A and B are bitwise independent. Then, the probability of the carry prediction error for the full adder is given by

$$\begin{aligned}
&\mathbf{P}(E_{cin,n-1}) \\
&\quad= \mathbf{P}(P^{v-1}_{k-1:0}P^{v-2}_{k-1:0}\cdots P^{0}_{k-1:0}\,(G_{n-kv-2:0}+P_{n-kv-2:0})) \\
&\quad= \mathbf{P}(P^{v-1}_{k-1:0})\mathbf{P}(P^{v-2}_{k-1:0})\cdots\mathbf{P}(P^{0}_{k-1:0})\times \\
&\qquad\qquad\qquad \underbrace{(\mathbf{P}(G_{n-kv-2:0})+\mathbf{P}(P_{n-kv-2:0}))}_{\mathbf{P}(C^{un}_{out}=1)}
\end{aligned} \tag{4.15}$$

where $G_{n-kv-2:0}$ and $P_{n-kv-2:0}$ are mutually exclusive. And, they are given by

$$\begin{aligned}
&\mathbf{P}(G_{n-kv-2:0}) \\
&\quad= \mathbf{P}(g_{n-kv-2}+g_{n-kv-3}p_{n-kv-2}+\cdots+g_0\prod_{i=1}^{n-kv-2}p_i) \\
&\quad= \mathbf{P}(g_{n-kv-2})+\cdots+\mathbf{P}(g_0\prod_{i=1}^{n-kv-2}p_i) \\
&\quad= \frac{1}{4}+\frac{1}{4}\cdot\frac{1}{2}+\cdots+\frac{1}{4}\cdot\frac{1}{2^{n-kv-2}}=\frac{1}{2}\left(1-\frac{1}{2^{n-kv-1}}\right)
\end{aligned} \tag{4.16}$$

$$\mathbf{P}(P_{n-kv-2:0})=\mathbf{P}(\prod_{i=0}^{n-kv-2}p_i)=\prod_{i=0}^{n-kv-2}\mathbf{P}(p_i)=\frac{1}{2^{n-kv-1}}$$

where $g_{n-kv-2},\ g_{n-kv-3}p_{n-kv-2},\ \cdots,\ g_0\prod_{i=1}^{n-kv-2}p_i$ are mutually exclusive. Thus, by putting (4.7), (4.15) and (4.16) together, the overall comparator error rate by the carry-skip scheme under random inputs is

$$\begin{aligned}
&\mathbf{P}_{\mathbf{err,cmp}}(n,k,v) \\
&\quad =\mathbf{P}(\overline{A_{n-1}\oplus B_{n-1}})\mathbf{P}(E_{cin,n-1}) \\
&\quad =\frac{1}{2}\cdot\frac{1}{2^{kv}}\cdot\left(\frac{1}{2}\left(1-\frac{1}{2^{n-kv-1}}\right)+\frac{1}{2^{n-kv-1}}\right) \\
&\quad =\frac{1}{2^{kv+2}}\left(1+\frac{1}{2^{n-kv-1}}\right)
\end{aligned} \tag{4.17}$$

4.3 Simulation Results

The proposed approximate arithmetic units were designed in Verilog HDL and synthesized with a commercial 90 *nm* CMOS technology and standard cell library. Also, the gate-level netlists were translated into transistor-level to perform HSPICE simulations. Each sub-adder in the proposed adder was implemented using an RCA structure.

4.3.1 Error Rate of the Proposed Approximate Adder

First, we examine the error rates of the proposed adder with various values of n, k and v. Figure 4.6 exhibits the error rates of the proposed adder under random inputs. The error rate worsens as the input bit-width n increases for given k and v, whereas it improves as k increases for fixed n and v. The error rate is significantly reduced when v increases under the same n and k. Specifically, by leveraging one more sub-carry generator for carry speculation, the error rate of the 128-bit adder with k=4 and v=2 is decreased from 5.19% to 0.32%, representing a 16.23× improvement (with v=3). Under the case of n=16, k=4 and v=2, compared to the previously presented approximate adders [74, 14, 32], the proposed adder is able to considerably reduce the error rate from 5.86% to 0.18% for random input patterns. Hence, the proposed carry prediction technique is very appealing to both wide and narrow bit-width

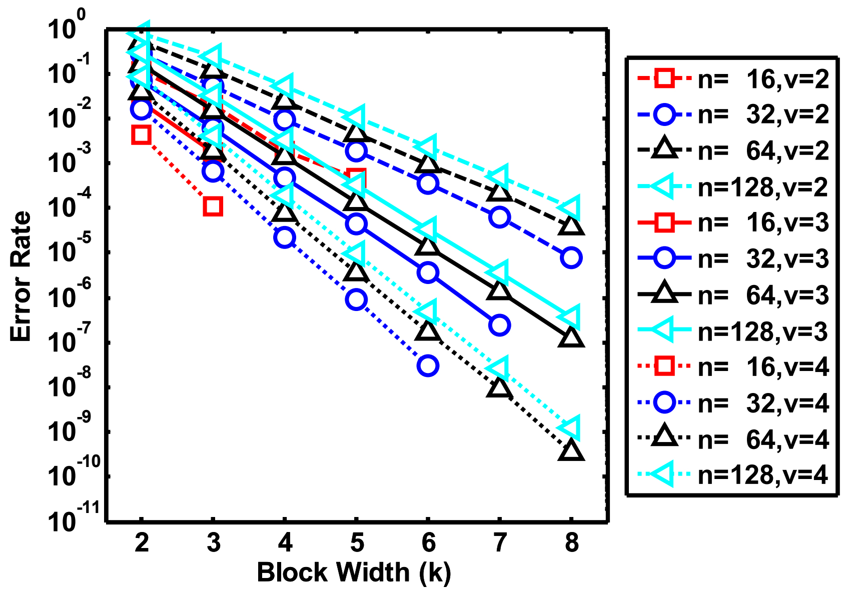


Figure 4.6: Error rates of the proposed adder under different n, k and v.

approximate additions requiring very low error rates.

4.3.2 Performance of the Proposed Approximate Adder

Table 4.1 reports the implementation results of area, delay, power, and error rate under the nominal supply of 1.2 V. Note that the error magnitude reduction circuit is included in all the proposed design implementations. The delay increases as k increases with a fixed n and v while the error rate and the power decrease. With a lower k for a given n and v, more carry prediction blocks are needed and the carry prediction with a smaller number of LSBs causes more errors. Meanwhile, under a given k and v, while the delay remains almost the same as n increases, the error rate deteriorates slowly. The proposed 64-bit adder with k=4 and v=2 has an error rate even over 2× less than the error rate of 5.86% of other 16-bit approximate adders with the same k [74, 14, 32]. Furthermore, when the adder exploits two more sub-carry generators for the carry prediction (*i.e.* v=4), it achieves 295× reduction of

Table 4.1: Proposed adder with different n, k and v.

Parameters (n, k, v)	*Area* (μm^2)	*Delay* (ps)	*Power* (mW)	*Energy* (pJ)	*Error Rate* (%)
(16, 2, 2)	525	202	0.902	0.182	11.55
(16, 3, 2)	533	297	0.708	0.210	2.05
(16, 4, 2)	466	359	0.600	0.215	0.18
(16, 5, 2)	509	431	0.591	0.255	0.05
(16, 2, 3)	550	221	0.917	0.203	2.34
(16, 3, 3)	543	324	0.744	0.241	0.17
(16, 2, 4)	557	232	0.897	0.208	0.44
(16, 3, 4)	547	332	0.863	0.287	0.01
(32, 4, 2)	1147	345	1.682	0.581	0.91
(64, 4, 2)	2389	344	3.039	1.046	2.36
(128, 4, 2)	4873	339	5.827	1.976	5.19
(32, 4, 4)	1185	363	1.884	0.684	0.002
(64, 4, 4)	2846	381	3.105	1.183	0.008
(128, 4, 4)	5250	370	6.009	2.222	0.019

the error rate at the expense of merely 6.6%, 10.7% and 2.2% extra area, delay and power, respectively.

4.3.3 Comparison with Seven Other Approximate Adders

We also implemented 16-bit (*i.e.* n=16) two traditional accurate adders (RCA and CLA) and seven previously presented approximate adders, which are Lu's Adder (LUA) [39], LOA [40], ETAI [76], ETAII [74], VLCSA-1 [14], ACA [32] and Dither Approximate Adder (DAA) [47] in the same commercial 90 *nm* CMOS technology, so as to compare with the proposed adder in various aspects. The VLCSA-1 and ACA have their own error detection and correction (EDC) mechanisms. The invoking of these modules, however, requires additional clock cycles, leading to timing overhead and potential architectural design complications needed for facilitating a

Table 4.2: Comparison with other 16-bit adders.

Design	*Area* (μm^2)	*Delay* (*ps*)	*Power* (*mW*)	*Energy* (*pJ*)	*Error Rate* (%)	*Avg. Error Magnitude*	*EDP* ($pJ \cdot ps$)	*EDAP* ($pJ{\cdot}ps{\cdot}\mu m^2$)	*EDEP* ($pJ \cdot ps \cdot \%$)
RCA	334 (0.72×)[1]	856 (2.38×)	0.343 (0.57×)	0.294 (1.37×)	N/A	N/A	251 (3.26×)	83936 (2.33×)	N/A
CLA	514 (1.10×)	407 (1.13×)	0.922 (1.54×)	0.375 (1.74×)	N/A	N/A	155 (2.01×)	81613 (2.27×)	N/A
LUA	609 (1.31×)	234 (0.65×)	0.908 (1.51×)	0.212 (0.99×)	16.68 (92.67×)	1363.7 (181.83×)	50 (0.65×)	30210 (0.84×)	827 (59.07×)
LOA (8-8)	200 (0.43×)	450 (1.25×)	0.420 (0.70×)	0.189 (0.88×)	43.75 (243.06×)	111.3 (14.84×)	85 (1.10×)	16976 (0.47×)	3719 (265.64×)
ETAI (8-8)	234 (0.50×)	435 (1.21×)	0.470 (0.78×)	0.204 (0.95×)	90.00 (500.00×)	178.3 (23.77×)	89 (1.16×)	20711 (0.58×)	7980 (570.00×)
ETAII	374 (0.80×)	254 (0.71×)	0.564 (0.94×)	0.143 (0.67×)	5.86 (32.56×)	127.5 (17.00×)	36 (0.47×)	13583 (0.38×)	213 (15.21×)
VLCSA-1[2]	673 (1.44×)	277 (0.77×)	1.337 (2.23×)	0.370 (1.72×)	5.86 (32.56×)	127.5 (17.00×)	103 (1.34×)	69159 (1.92×)	602 (43.00×)
ACA[2]	472 (1.01×)	374 (1.04×)	0.666 (1.11×)	0.249 (1.16×)	5.86 (32.56×)	127.5 (17.00×)	93 (1.21×)	44010 (1.22×)	546 (39.00×)
DAA (8-8)	370 (0.79×)	435 (1.21×)	0.566 (0.94×)	0.246 (1.14×)	25.00 (138.89×)	74.7 (9.96×)	107 (1.39×)	39509 (1.10×)	2671 (190.79×)
Proposed[3]	466	359	0.600	0.215	0.18	7.5/14.1[4]	77	35959	14

[1] *(*) normalization against the proposed adder* [2] *without the error detection and correction* [3] *with the error magnitude reduction*
[4] *without the error magnitude reduction*

given processing application. To examine the approximate natures of the different adder designs, to be fair, we exclude the EDC modules and their timing, power and area overheads from this comparison. The same RCA structure is used for the sub-adders in ETAII, VLCSA-1 and ACA and the parameters of k=4 are adopted in these adders as well as LUA. The proposed adder employs the same k with v=2 and the RCA structure for the sub-adders. Moreover, we split LOA, ETAI and DAA to have both 8-bit sizes for the accurate and inaccurate parts and the RCA structure is used for the accurate parts. For the dithering in DAA, we utilize the MSB of one input of the inaccurate part (*i.e.* A_7) as the dithering bit in order to alleviate an overhead due to the external dither control as presented in [47]. Also, we denote these three adders by LOA (N-M), ETAI (N-M) and DAA (N-M), where N and M indicate the bit widths of the accurate and inaccurate parts, respectively.

The energy-delay product (EDP) is widely adopted to effectively show energy efficiency of circuits and systems. In addition to EDP, we take into account the energy-delay-area product (EDAP) [37, 38] to compare both the energy and cost of silicon-area of the adders. Importantly, the energy-delay-error product (EDEP) is particularly an interesting metric in approximate arithmetic designs because it considers not only energy and delay but also error rate [49]. Thus, the adders are compared in terms of EDP, EDAP and EDEP, as well as the fundamental metrics, such as area, delay, power, energy, error rate and average error magnitude. Table 4.2 summarizes the performance comparison under the regular supply of 1.2 V. The RCA exhibits the lowest power with the longest delay due to the bit-by-bit carry propagate chain and the CLA consumes the largest energy. While the RCA is more energy and area efficient than CLA, over 2$\times$ longer delay degrades its EDP and EDAP. The LUA is the fastest but occupies the second largest area due to the considerable number of carry generators. Additionally, the carry prediction approach in LUA makes the errors to be able to occur in higher significant bits, which leads to the highest average error magnitude among the approximate adders. The error rate of EATI (8-8) reaches 90%, which may limit its practical use, due to lack of carry prediction for the accurate part (*i.e.* the carry is fixed to zero) and worsens EDEP remarkably. On the other hand, thanks to the simple carry speculation scheme of LOA (8-8), which is achieved by ANDing two MSBs of each operand of the inaccurate part, the error rate is improved to 43.75%, which is still fairly large. The use of the simple OR operation for the inaccurate part allows it to be the most area efficient adder. The proposed dithering control for the inaccurate part in DAA (8-8) further improves the error rate as well as the average error magnitude with area and power overheads. The high error rates in ETAI (8-8), LOA (8-8) and DAA (8-8) exacerbate the EDEP metric. Fortunately, the adders have fairly low average error magnitudes despite of the high error rates since approximation errors are concentrated on lower significant bits (*i.e.* inaccurate parts). Thanks to the dithering, the DAA (8-8) demonstrates the best performance in terms of the error rate, average error magnitude and EDEP among these three adders. Among the adders having the same error rate of 5.86%, the ETAII is the most efficient in terms of all the metrics. As a result of the use of carry selection in VLCSA-1, it dissipates the highest power, which is up to 3.9× more than the others and incurs EDP, EDAP and EDEP degradations. The proposed adder is 2.4× faster and 3.3× EDP efficient than RCA. The carry-skip scheme allows it to have the lowest error rate of 0.18% and EDEP of 14 among the approximate adders. Furthermore, the proposed error magnitude reduction approach improves the average error magnitude by 1.88× from 14.1 to 7.5, which is also the lowest value among the approximate adders. Our design is comparable to ACA with respect to area, delay, power and energy while having much lower error rate, average error magnitude and EDEP thanks to carry-skip.

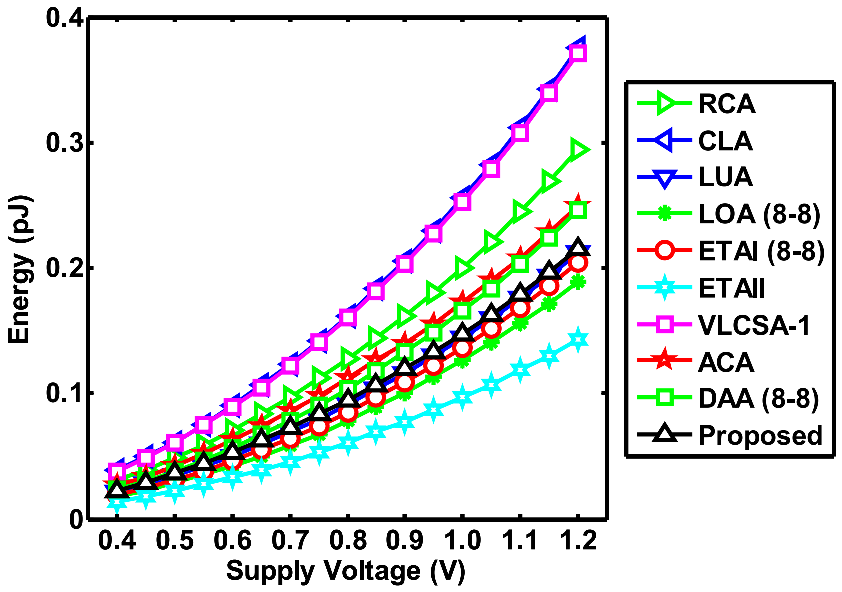


Figure 4.7: Energy comparison under supply scaling.

Figure 4.7 plots the energy comparison under scaled voltages. The energy efficiency of VLCSA-1 is hindered by a high power dissipation in spite of its relatively fast speed. It takes almost the same amount of energy as CLA that consumes the highest energy, whereas the ETAII does the lowest among the adders. The good tradeoff between delay and power obtained by the carry-skip approach allows our adder to be more energy efficient than the two accurate adders, VLCSA-1, ACA and DAA (8-8). Particularly, our design attains an energy saving of 27% and 43% compared to RCA and CLA, respectively. Besides, the proposed adder, LUA, LOA (8-8) and ETAI (8-8) show similar energy consumptions under the scaled voltages while our design enjoys the lowest error rate.

4.3.4 Comparison on Error-Free Operations

The main objective of this work is to develop an energy efficient approximate adder with a low error rate for neuromorphic applications. For completeness, we also compare the error-free operations of various designs. We consider the EDC schemes for VLCSA-1, ACA and the proposed adder. In each of these designs, the error detection is achieved by checking the propagate and generate signals in the approximate addition phase. Upon detecting an error, the error correction circuit reconstructs accurate results by leveraging propagate and generate signals [68, 14] or by adding "1" to sub-adder output [32], either of which requires an additional clock cycle. The VLCSA-1 and ACA exploit the prefix adder and incrementor, respectively, for error correction. For the proposed adder, the prefix adder based error correction circuit is implemented to produce error-free results [14]. Note that the error magnitude reduction of our adder is not necessary here and thus removed in this implementation. Obviously, the error correction circuit is activated whenever errors are detected in the addition phase [32] and the effective energy E_{eff} can be expressed by

$$E_{eff} = P_{apx} t_{apx} + \mathbf{P_{err}} \cdot P_{ec} t_{ec} \tag{4.18}$$

where P_{apx}, t_{apx}, P_{ec} and t_{ec} are the power and delay of the approximate adder and those of the error correction circuit, respectively, and $\mathbf{P_{err}}$ is the error rate of the approximate adder. The implementations are summarized in Table 4.3. The error correction circuits have shorter delays than the respective approximate adders. The critical path delays of VLCSA-1 and ACA are slightly longer than the delays in Table 4.2 due to the additional error detection logic. Conversely, the proposed adder's critical path delay becomes shorter in spite of the error detection circuit since the error magnitude reduction block is eliminated. Our design occupies the lowest area and is the most efficient adder in terms of power and effective energy.

Table 4.3: Approximate adders with error detection and correction.

Design	*Area* (μm^2)	*Delay*[1] (ps)	*Power* (mW)	*Energy*[2] (pJ)
VLCSA-1	899 (1.72×)[3]	296 (0.93×)	2.094 (2.38×)	0.473 (2.43×)
ACA	580 (1.11×)	389 (1.22×)	1.269 (1.44×)	0.287 (1.47×)
Proposed	524	318	0.881	0.195

[1] *critical path delay* [2] *effective energy*
[3] *(*) normalization against the proposed adder*

4.3.5 Error Rate of the Proposed Approximate Comparator

Figure 4.8 shows the error rates of the proposed approximate comparator with various n, k and v under random inputs. Unlike the proposed adder, the comparator error rate does not deteriorate and remains almost the same although n increases

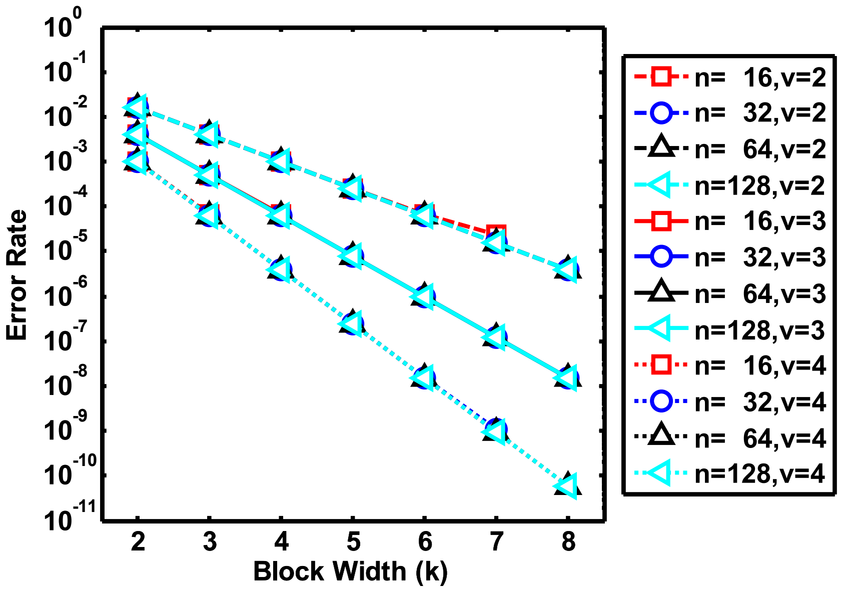


Figure 4.8: Error rates of the proposed comparator with various n, k and v.

under fixed k and v. This is because under the given k and v, $\left(1 + \frac{1}{2^{n-kv-1}}\right) \approx 1$ when n is fairly large and the overall error rate is, therefore, dominated by $\frac{1}{2^{kv+2}}$ according to (4.17). Also, the error rates are even better than those of the proposed adders in the same n, k and v (*e.g.* 11.55% *vs* 1.56% for the 16-bit comparator and the adder with k=2 and v=2). The main reason is that the proposed comparator necessitates only one correct carry prediction for the 1-bit full adder for the MSB to attain the correct comparison result, whereas the proposed adder requires the carry-in signals for all the sub-adders to be correct to achieve the correct summation. Consequently, it is equally attractive in very low error rate approximate comparisons for both wide and narrow bit-widths inputs.

4.3.6 Performance of the Proposed Approximate Comparator

Table 4.4 lists the comparator implementations with various n, k and v under the supply voltage of 1.2 V. The proposed comparators exhibits better performances than the proposed adders under the same n, k and v in terms of all the aspects since they consider only kv+1 bits of n-bit inputs for comparisons and require less

Table 4.4: Proposed comparator with different n, k and v.

Parameters (n, k, v)	*Area* (μm^2)	*Delay* (ps)	*Power* (mW)	*Energy* (pJ)	*Error Rate* (%)
(16, 2, 2)	61	152	0.089	0.014	1.563
(16, 4, 2)	121	215	0.111	0.024	0.098
(16, 2, 3)	101	194	0.132	0.026	0.391
(16, 4, 3)	193	233	0.155	0.036	0.007
(16, 2, 4)	140	216	0.163	0.035	0.098
(32/64/128, 4, 2)	121	215	0.111	0.024	0.098[1]
(32/64/128, 4, 4)	288	257	0.198	0.051	$3.81e^{-4}$[1]

[1] *error rate differences among 32-/64-/128-bit comparators are* $< 10^{-6}$ *%*

number of sub-carry generators than the adders, leading to the smaller area, delay and power. All the 16-bit comparators exhibit very low error rates ($< 1.6\%$) with good area and energy efficiencies and hence can be applied to low-cost comparisons in error-tolerant applications with a desired accuracy. Interestingly, the proposed comparators are the identical designs regardless of the input bit-width n as long as k and v are the same in that they consist of v k-bit sub-carry generators in spite of different values of n. Only difference here is the error rate. Interestingly, the error rate differences among the comparators in Table 4.4 are negligibly small as mentioned in Section 4.3.5 and thus ignorable. Hence, the same design can be reused without any change for various bit-widths comparisons with the same k and v while enjoying almost the same extremely low error rates.

From the authors' best knowledge, unfortunately, no approximate comparator is presented to date. So, we compare the proposed 16-bit comparator with k=4, v=2 to the two accurate comparators, which are ripple carry and carry lookahead based comparators (RCC and CLC, respectively). They are also implemented with the same 90 nm CMOS process and the results are summarized in Table 4.5. The accurate comparators have less area, delay, power and energy than their corresponding adders in Table 4.2 because they requires only the carry for the MSB and no logics to produce summation outputs. Also, they show much better EDP and EDAP performances than the adders. The proposed comparator demonstrates the best performance in all the aspects except that it consumes more power than RCC. It is up to $18\times$ more efficient than the other designs in terms of EDP and EDAP with an extremely low error rate ($< 0.1\%$), which is well suitable for error-tolerant applications.

Table 4.5: Comparison with other 16-bit comparators.

Design	*Area* (μm^2)	*Delay* (ps)	*Power* (mW)	*Energy* (pJ)	*Error Rate* (%)	*EDP* ($pJ \cdot ps$)	*EDAP* ($pJ{\cdot}ps{\cdot}\mu m^2$)	*EDEP* ($pJ \cdot ps \cdot \%$)
RCC	146 (1.21×)[1]	834 (3.88×)	0.090 (0.81×)	0.074 (3.08×)	N/A	62 (12.40×)	9086 (14.65×)	N/A
CLC	423 (3.41×)	323 (1.69×)	0.258 (2.90×)	0.083 (4.88×)	N/A	27 (5.40×)	11342 (18.29×)	N/A
Proposed	121	215	0.111	0.024	0.098	5	620	0.50

[1] *(*) normalization against the proposed comparator*

4.4 Summary

In this section, novel approximate adder and comparator designs to considerably reduce energy consumption with a very moderate error rate has been presented for energy efficient neuromorphic VLSI systems. The proposed carry prediction with carry-skip scheme significantly enhances the overall error rate and the critical path delay. Additionally, the error magnitude reduction technique for the adder reduces the amount of error further with low cost. Implemented in a commercial 90 *nm* CMOS process, the proposed adder is 2.4× faster and 43% more energy efficient over traditional adders with an error rate of merely 0.18%. Furthermore, the proposed approximate comparator exhibits an extremely low error rate of 0.098% and achieves an energy reduction of up to 4.9× over the conventional ones.

5. APPLICATION OF APPROXIMATE ARITHMETIC TO NEUROMORPHIC COMPUTING

5.1 Evaluation Environment

To evaluate our approximate arithmetic units designed in Section 4, we consider the general digital neuromorphic VLSI system described in Figure 2.13. Additionally, we adopt the digital LIF model for the silicon neurons since the LIF model is suitable for digital implementation with a few arithmetic components among various neuron models. This model is widely used in digital neuromorphic chip [58, 36] and its dynamics can be indicated by

$$\begin{aligned} V_i^{t+1} &= V_i^t + K_{syn}\sum_{j=1}^{M} w_{ji}S_j^t + K_{ext}E_i^t - V_{leak} \\ S_i^{t+1} &= \begin{cases} 1 & if\;\; V_i^{t+1} > V_{th} \\ 0 & otherwise \end{cases} \end{aligned} \tag{5.1}$$

where V_i^t is the membrane potential of neuron i at time t, S_i^t is the spike bit that indicates whether neuron i fired at time t and is set to "1" when the membrane potential exceeds the given threshold voltage V_{th}, w_{ji} is the synaptic weight between neuron j and i, E_j^t is the spike bit for the external input for neuron i, M is the number of pre-synaptic neurons, V_{leak} is the leaky potential, and K_{syn} and K_{ext} are the weight parameters for synapses and external input spikes, respectively. In (5.1), additions and comparisons are certainly the key operations to calculate the membrane potentials and determine the firing activities, respectively.

A digital implementation of the LIF neuron is shown in Figure 5.1(a). It contains a multiplier, a comparator and an adder. It accumulates one of the pre-synaptic

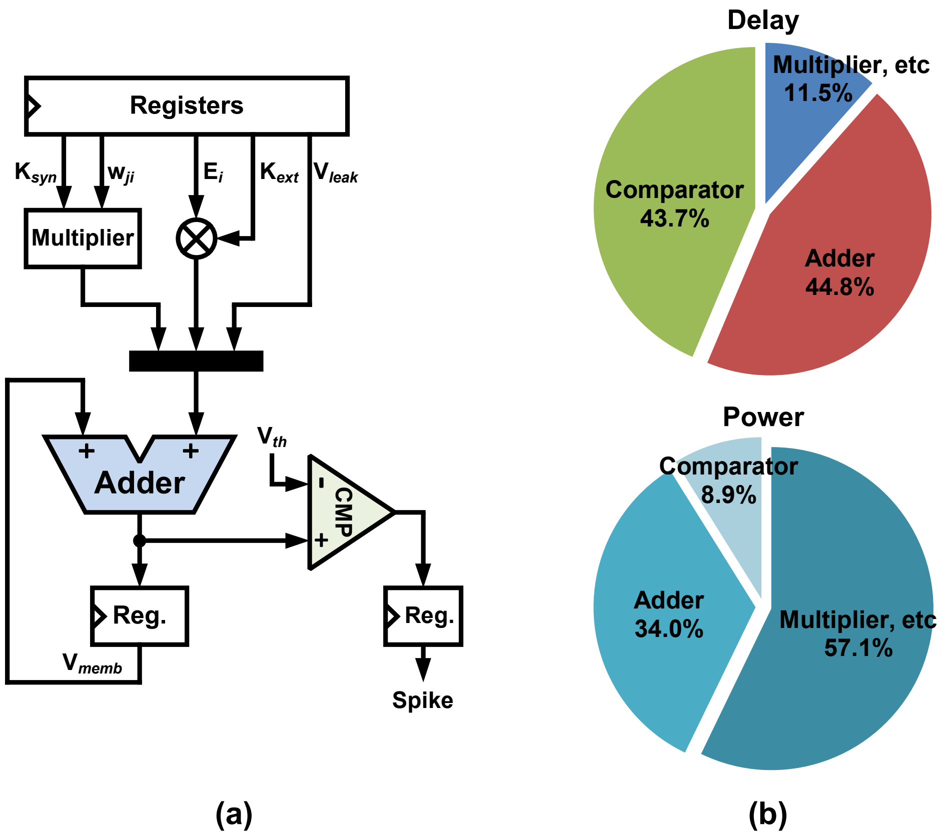


Figure 5.1: Digital LIF neuron: (a) block diagram and (b) delay and power breakdowns.

weights, external input and leaky potential, which correspond to the terms $K_{syn}w_{ji}S_j^t$, $K_{ext}E_i^t$ and $-V_{leak}$ in (5.1), respectively, through the multiplexer at a time. If the adder output exceeds the given threshold voltage V_{th}, the digital comparator produces a spike. The adder and comparator dominate the computation time because the multiplier is relatively small due to narrower bit-widths in multiplications. To demonstrate the portion of the adder and comparator for the digital LIF silicon neuron, we show the delay and power breakdowns of the LIF neuron when implemented with the ripple carry based adder and comparator in Figure 5.1(b). Each synaptic weight, model parameter and membrane potential are represented using 3, 3 and 16 bits, respectively. The adder and comparator contribute to 88.5% of the processing time and 42.9% of the power in the entire LIF computation. Therefore, it is extremely crucial to reduce the delay and power of the adder to improve overall energy efficiency of neuromorphic computing.

Evaluating the performance of the proposed arithmetic units by simulating the long training process of the neuromorphic system at the transistor level is computationally intractable. Instead, we develop a hardware-aware spiking neural network simulator for the neuromorphic VLSI system to evaluate the performance of our new arithmetic designs. The key network features and hardware design parameters including the digital LIF neuron dynamics, the STDP learning rule, bit-widths used to represent various neuron model parameters are fully captured in the simulator. The proposed approximate adder and comparator are carefully characterized and their circuit profiles are extracted from HSPICE simulations [35]. To evaluate the approximate nature of our designs, we disable the error correction logic of the adder and inject the characterized input-specific adder and comparator errors into each addition and comparison operations in the behavioral simulator, providing a precise evaluation of the impacts of the approximate errors for neuromorphic computation.

We use the neuromorphic application for character recognition system as illustrated in Section 3 to systematically examine the impacts of approximate adder and comparator errors of several designs. We specifically consider the case where the neuromorphic hardware is configured to be the same two-layer network for character recognition as illustrated in Figure 3.16. Here, we extend the network to have over a thousand silicon neurons such that the input layer contains 1024 excitatory neurons receiving binary inputs representing pixel values in a 32×32 pixel input pattern while the output layer has 36 excitatory neurons receiving inputs from all excitatory input neurons through plastic synapses. The behavior of each layer is modulated by the inhibitory neurons as described in Section 3. To train the network, 26 input patterns of alphabets "*A*" – "*Z*", which are 32×32 pixel patterns, are applied one by one to the input layer. In this network, we use 3, 3 and 16 bits respectively to represent each synaptic weight, model parameter and membrane potential for each neuron and employ a 16-bit adder and comparator for the LIF computation.

5.2 Impacts of Approximation Errors on Neuromorphic Applications

5.2.1 Approximate Adder Error Effects

First, we fix the supply level to 1.2 V and clock the chip at the nominal clock rate of 100 *MHz* so that the errors produced are only due to the approximate natures of the adders since there is no timing failure. Also, the accurate 16-bit comparator RCC is used for the digital LIF neurons to compare the threshold voltage with the membrane potentials to generate neurons' firing activities. Figure 5.2(a) shows the

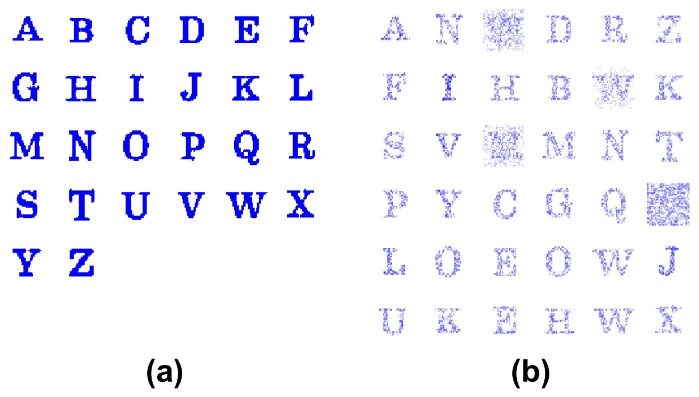


Figure 5.2: (a) Input character patterns and (b) receptive fields with 16-bit accurate adders.

input character patterns "*A*" to "*Z*" for the training and the receptive fields of all excitatory output neurons after the training with the accurate adders (*i.e.* RCA and CLA). The receptive fields as in Figure 5.2(b) are trained well to respond to the inputs from "*A*" to "*Z*". This means that every letter appears once at least in the receptive fields. The results in Figure 5.2(b) serves as a golden reference for the approximate adders.

We also test the proposed approximate adder and other approximate adders with the network. The receptive fields after the training with various approximate adders are shown in Figure 5.3. The corresponding error rates and average error magnitudes during the learning process are listed in Table 5.1 as well. The proposed 16-bit adder

Table 5.1: Error rates and average error magnitudes of various adders during training process.

Design	*Error Rate* (%)	*Avg. Error Magnitude*
LUA	14.24	200.47
LOA (8-8)	61.05	11.82
ETAI (8-8)	14.32	8.81
ETAII	14.24	544.11
VLCSA-1	14.24	544.11
ACA	14.24	544.11
DAA (8-8)	13.04	7.68
LOA (13-3)	60.95	1.59
ETAI (15-1)	17.50	0.17
DAA (11-5)	23.25	0.59
Proposed	0.18	0.03

with k=4 and v=2 has an error rate of merely 0.18% with an average error magnitude of 0.03 for the LIF computations during the training. Note that no EDC is considered but the error magnitude reduction is. Fortunately, thanks to the error resilience of

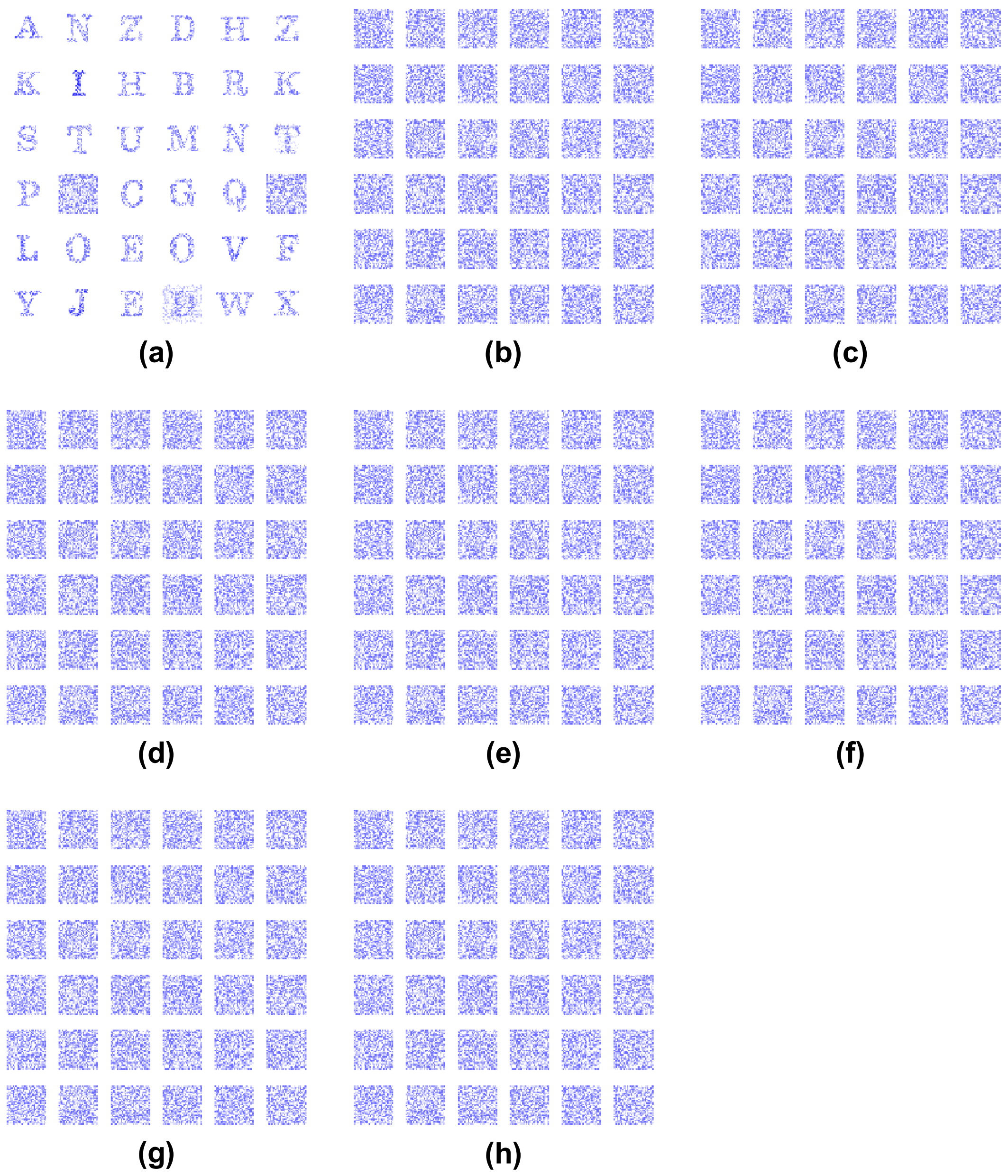


Figure 5.3: Receptive fields with 16-bit (a) proposed approximate adder, (b) LUA, (c) LOA (8-8), (d) ETAI (8-8), (e) ETAII, (f) VLCSA-1, (g) ACA and (h) DAA (8-8).

the neuromorphic system, Figure 5.3(a) shows that the receptive fields are trained successfully to recognize all the letters and the approximation errors have negligible effect on the training process of the character recognition system. For LOA, ETAI and DAA, the 8-bit accurate and inaccurate parts are used (*i.e.* LOA (8-8), ETAI (8-8) and DAA (8-8)). The MSB of one input of the inaccurate part (*i.e.* A_7) is leveraged for the dithering bit in DAA. The parameters of n=16 and k=4 are adopted in LUA, ETAII, VLCSA-1 and ACA and only the approximate adder part (*i.e.* without the EDC) is utilized for the later two adders. All these approximate adders have an error rate of more than 13% during the learning process. Especially, the error rate of LOA (8-8) reaches 61.05% throughout the training. As seen in Figure 5.3(b) – (h), the approximate adders produce a set of receptive fields with random synaptic weights. These high error rates give rise to failures in training the network since the approximation errors cause the neurons to either fire randomly or cease to fire. In particular, the 2's complement signed additions of small numbers frequently occur during leaky operations (*i.e.* $-V_{leak}$ in (5.1)) for the training. In this case, the LUA, ETAII, VLCSA-1 and ACA produce many wrong carry predictions, incurring an error rate of more than 14% and a high average error magnitude over 200 during the learning process and an unacceptable performance degradation. This result suggests the carry speculation with only 4-bit of less significant inputs in these 16-bit adders may be insufficient for this application.

To shed more light on this, we increase the accuracy of LOA, ETAI and DAA by expanding the accurate part of the adder at the cost of increased delay and energy dissipation. When the LOA, ETAI and DAA have 13, 15 and 11 bits accurate parts, respectively, the network starts to perform better. Figure 5.4 illustrates the receptive fields with these adders and the respective error rates and average error magnitudes are also listed in Table 5.1. Although the LOA (13-3) still has a relatively high error

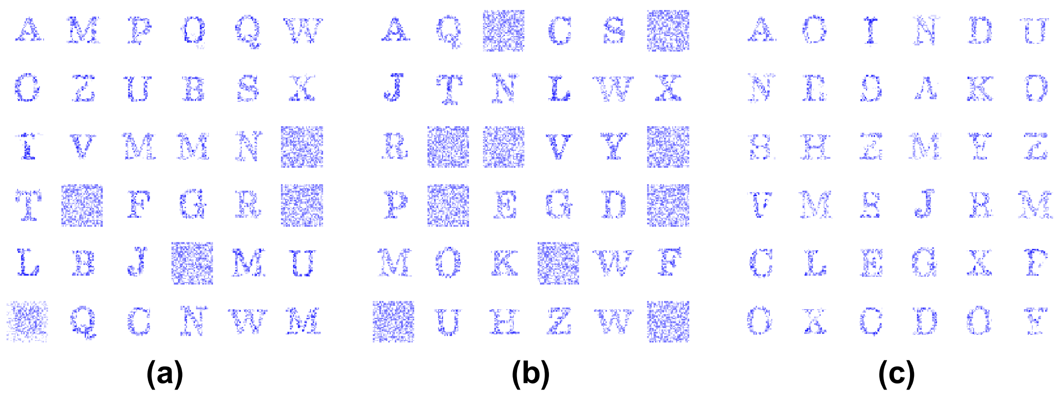


Figure 5.4: Receptive fields with 16-bit (a) LOA (13-3), (b) ETAI (15-1) and (c) DAA (11-5).

rate of 60.95%, the corresponding receptive fields as in Figure 5.4(a) are trained such that all alphabets except for "*D*", "*E*", "*H*" and "*K*" can be identified. In addition, a vague "I" shaped receptive field is trained, which may be confused with "Y". Due to the expansion of the accurate part of the adder, the errors now concentrate more on LSBs and the average error magnitude is reduced from 11.82 to 1.59. Similarly, the inclusion of a 15-bit accurate part in ETAI (15-1), which has an error rate of 17.50% with an average error magnitude of 0.17 during the training, allows the network to be trained for all letters except for "*B*", "*H*" and "*I*" as illustrated in Figure 5.4(b). The DAA requires only three more accurate bits to train the network better and achieves a relatively low average error magnitude of 0.59 thanks to the dithering scheme. Unfortunately, the high error rate of DAA (11-5) hinders the network from training all the letters. Several letters are not shown in the receptive fields and the other letters in the fields are less distinct than the receptive fields with the other adders as seen in Figure 5.4(c). Clearly, our design outperforms all other approximate adders.

5.2.2 Approximate Comparator Error Effects

To see the impacts of the errors of the proposed approximate comparator on the neuromorphic computing, we replace the accurate comparator by the proposed approximate one with k=4 and v=2 in the neuron circuits. The receptive fields with the proposed comparator are illustrated in Figure 5.5. Note that the errors are in-

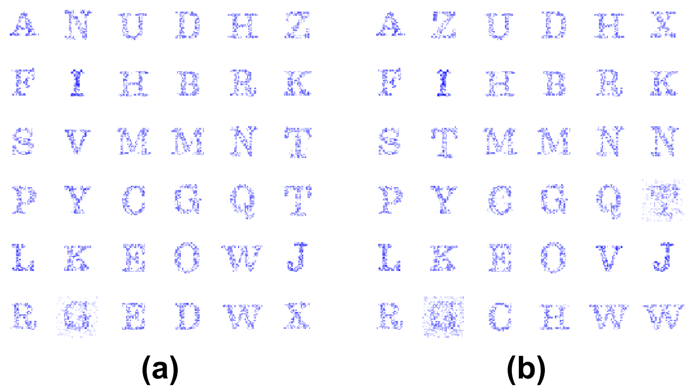


Figure 5.5: Receptive fields with 16-bit (a) accurate adder with proposed comparator and (b) proposed adder with proposed comparator.

duced from only the approximations without timing failures of the circuits. Figure 5.5(a) is the receptive fields with the accurate adder and the proposed comparator. The proposed comparator allows the network to train well to recognize all the letters by virtue of the extremely low error rate of 0.45% and the error resilience of neuromorphic computing. Additionally, the proposed approximate adder together with the proposed comparator also produce the good receptive fields that show all the letters as in Figure 5.5(b). The error rates of the adder and comparator are 0.19% and 0.48%, respectively, while training. These very low error rates affect the learning process negligibly and our two designs can be, therefore, equally adopted for the neuromorphic application without performance degradations.

5.3 Energy Efficiency of LIF Neurons with Approximate Adders and Comparators with Supply Voltage Scaling

To show the energy efficiency of the these adders in the neuromorphic hardware, we scale down the supply voltage and obtain the energy dissipation in one LIF operation involving a multiplication of a synaptic weight w_{ji} with the weight parameter K_{syn} and an addition of the membrane potential V_i^t with the multiplier output $K_{syn}w_{ji}$, a key processing step in (5.1). A comparison between the membrane potential V_i^t and the threshold voltage V_{th} is also included for the LIF operation and the RCC is employed in the neuron circuit. The clock frequency is fixed at the maximum value such that the neuron with RCA and RCC can operate without any error in the regular supply voltage of 1.2 V. For each adder design, we scale down the supply voltage with a 0.05 V step as long as there is no critical timing failure created in the neuron circuit. Figure 5.6 plots the energy comparison of one LIF operation with the neurons with the different adders under scaled power supply levels. The energies are normalized against the neuron with RCA and RCC. Neurons with LUA, ETAII or VLCSA-1 can operate at a supply voltage of 1.0 V. The ETAII is the most energy efficient adder design while having a much larger error rate than the proposed adder and leading to poor learning performance (see Figure 5.3(e)). Regretfully, the high power consumption from the carry selection in VLCSA-1 is an obstacle to attain energy efficiency. All the adders except for RCA is able to work at the scaled supply voltage of 1.05 V. The LUA, ETAI and the proposed adder show the similar energy efficiency and the proposed adder consumes about 8% and 6% less energy than ACA and DAA (8-8), respectively. Our design achieves the energy savings of

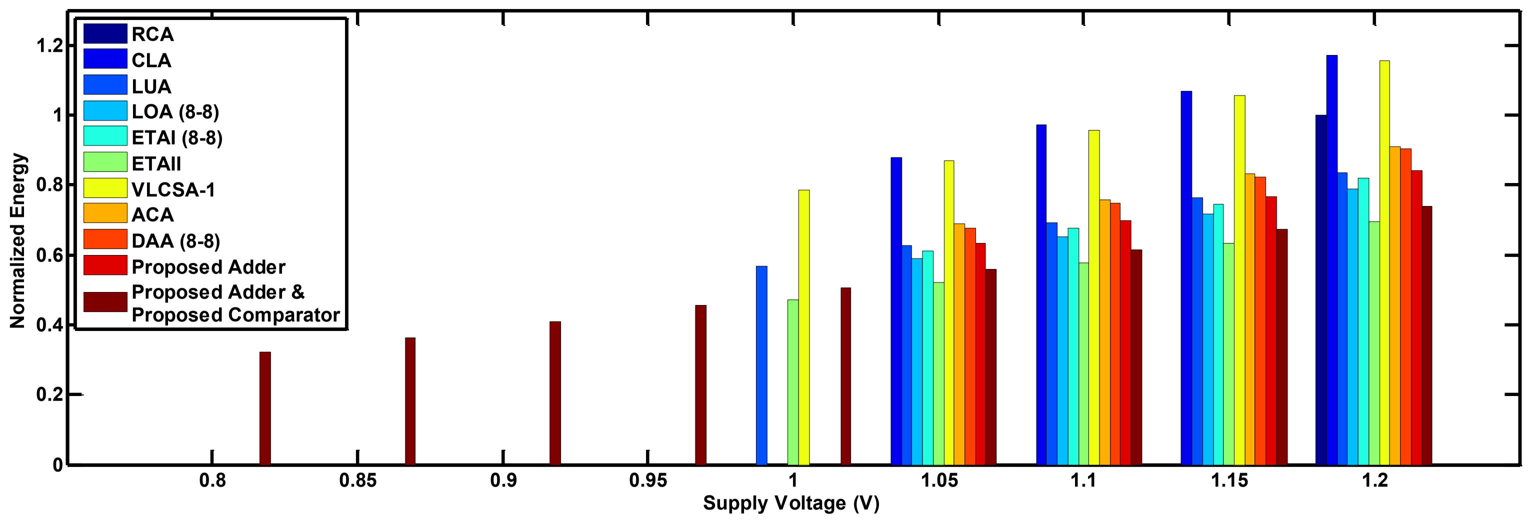


Figure 5.6: Normalized energies of one digital LIF neuron with various adders with supply voltage scaling.

up to 36.6% and 27.9% over RCA and CLA, respectively, in the scaled supply. It can be seen that our adder has the most competitive energy and error tradeoff among all these designs. We also compare a neuron leveraging the proposed adder and the approximate comparator to the others. This neuron allows the supply voltage to decrease to 0.8 V, which is 0.25 V lower than a neuron including the proposed adder only. The proposed adder and comparator enables the neuron to be 1.97×, 2.73× and 3.11× energy efficient over the neuron adopting the proposed adder, CLA and RCA with the accurate comparator RCC, respectively, in the scaled supply without performance degradation (see Fig 5.5(b)). Our comparator also provides a great energy saving with very low error rate for the neuromorphic computing Since a few hundreds of silicon neurons are integrated in the form of an array [58, 36], the total energy saving resulted from our designs are remarkable for the neuromorphic chip.

5.4 Energy Efficiency during the Training Process with Supply Voltage Scaling

Finally, we examine the overall energy consumption by all the digital LIF neurons in the network for the training process. The LIF neuron dynamics in (5.1) is divided into three different types of additions to obtain the membrane potential and one comparison to determine the firing activity:

1) an addition of the membrane potential V_i^t with the multiplier output of each synaptic weight w_{ji} and the weight parameter K_{syn};

2) an addition of the membrane potential V_i^t with the external spike weight parameter K_{ext};

3) an addition of the membrane potential V_i^t with the leaky potential $-V_{leak}$;

4) a comparison of the membrane potential V_i^t with the threshold voltage V_{th}.

We extract the energy profiles for these three additions and one comparison from HSPICE simulations and inject the characterized energies into the behavioral simulator. In the synaptic weight integrations, which correspond to term $K_{syn} \sum_{j=1}^{M} w_{ji} S_j^t$ in (5.1), the adders are activated only after the pre-synaptic neurons fired (*i.e* $S_j^t = 1$) for the type 1) addition. Similarly, they work only when the external spiking events are applied ($K_{ext} E_i^t$ in (5.1)) for the type 2) addition. Therefore, we take into account the neurons' firing and the external spike activities in the training to obtain the realistic energy consumptions of the neuron circuits. The simulator accumulates the energy dissipations of all the neuron circuits in the network by not only discriminating these addition types and comparison but also considering the neurons' firing and the external spike activities during the learning process, achieving an accurate analysis of energy dissipations of all the neurons. We consider the error-free operation and the proposed addition scheme since the other approximation approaches show unacceptable learning performances. For the error-free operation, we enable the EDC of ACA, VLCSA-1 and the proposed adder to achieve the same receptive fields as the accurate adder after the training (*i.e.* Figure 5.2(b)). Also, the RCC is utilized for accurate comparison and the EDC is only activated only when errors are detected. The clock frequency of the chip is set to the same as that in Section 5.3 since the digital neuron circuits have the overall critical timing path of the neuromorphic chip [36]. The supply voltage is scaled down with a 0.05 V step without any critical timing failure in the neurons as well. Figure 5.7 depicts the overall energy consumed by the digital LIF neurons during the learning process. The energies are also normalized as the same in Figure 5.6. While the chip with all the designs except for RCA can operate at a supply voltage of 1.05 V, the proposed adder with EDC is 1.75× and 1.30× more energy efficient than VLCSA-1 and ACA, respectively. The proposed adder with EDC requires the smallest amount of energy for the

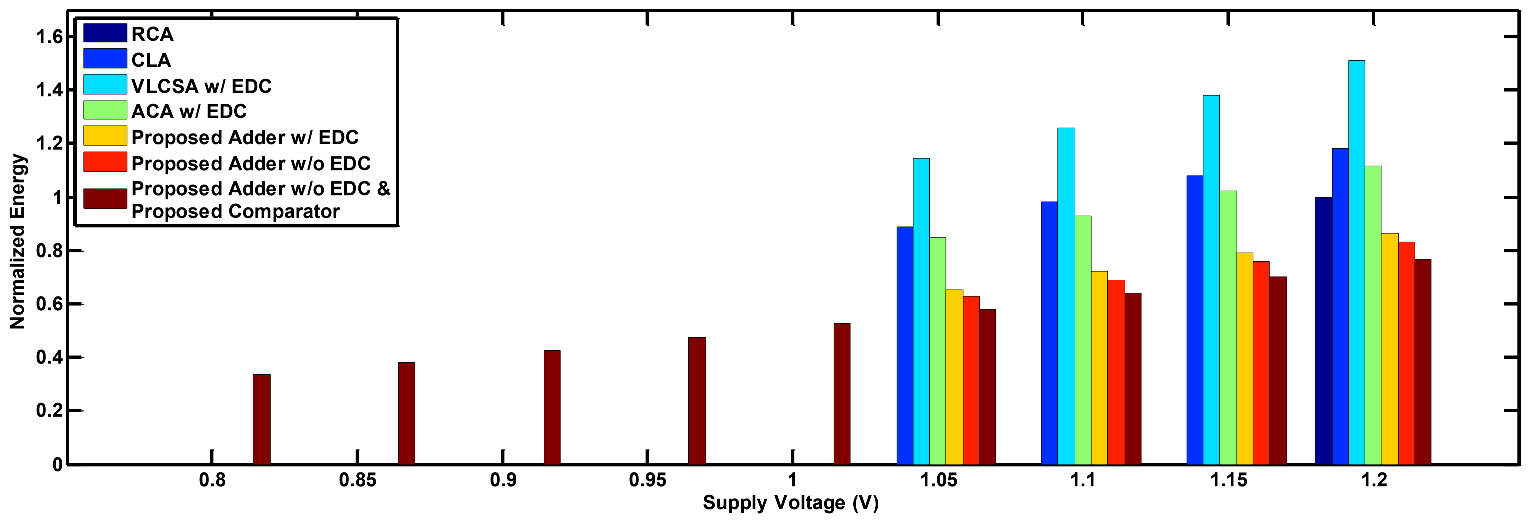


Figure 5.7: Normalized energy consumptions by all the digital LIF neurons of the network while training with various adders and comparators under supply voltage scaling.

neuron circuits among the error-free adders, which encompass RCA, CLA and EDC enabled VLCSA-1/ACA/proposed adder, thanks to the low-overhead EDC circuit. The proposed approximate adder without EDC makes the neurons dissipate 29.3% and 37.4% less energy than CLA and RCA under the scaled voltage. Moreover, when enabled with the proposed comparator and adder, digital LIF neurons are able to work at the scaled supply of 0.8 V while achieving an energy saving of 48.8% to 77.8% over all other error-free adders (*e.g.* 66.5% over RCA) with negligible performance degradation. The proposed arithmetic units demonstrate significant energy savings for the neuromorphic hardware. Additionally, the proposed adder with EDC also exhibits improved energy efficiency than the other error-free adders and can be equally employed for energy efficient accuracy significant applications.

5.5 Summary

This section has demonstrated the performance of the proposed approximate adder and comparator as part of an unsupervised learning based VLSI neuromorphic character recognition chip by developing a hardware-aware simulation approach. The results have proven that the approximation errors of the proposed adder affect the training performance negligibly while the other approximate adders severely degrade the learning performance. The digital LIF neuron adopting the proposed arithmetic units enables it to be over 3× energy efficient compared with the traditional accurate arithmetic ones. Moreover, the proposed adder and comparator allow for the energy saving of up to 66.5% over traditional counterparts for the digital LIF neurons during the learning process with scaled supply voltage levels.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This dissertation has developed techniques for designing a neuromorphic processor and approximate arithmetic for low-cost, reconfigurable and energy efficient neuromorphic computing in VLSI. By addressing the several key issues on implementing brain-inspired hardware architecture, we have significantly improve both flexibility and performance of neuromorphic VLSI systems. Furthermore, the proposed approximate arithmetic and inherit error resiliency in neurocomputing allow for excellent energy efficiencies with negligible performance degradations in neural computation. We conclude this research by summarizing the major contributions.

For digital neuromorphic processor, we have proposed a scalable digital architecture that incorporates synapse, neuron and learning arrays for large scale spiking neural networks. The memristor nanodevice is leveraged to build a high-density synapse crossbar array that consists of novel multilevel memory cells to store both a multibit synapse value and a network configuration information. Through the systematic analysis of the memristor, we have considerably enhanced the synaptic weight update performance by reducing the programming time for the memristive array that can be accessed both column- and row-fashion with the low-cost digital PWM scheme. Additionally, the proposed column based ADC scheme allows the digital neuron to efficiently perform the LIF neuron dynamics and reduce the area and power overheads required for the LIF operations. When implemented in a commercial 90 nm CMOS technology, our design with 256 digital spiking neurons with learning circuits and 65,536 synapses is evaluated to occupy an area of 1.86 mm^2 and dissipate a power of 6.45 mW under a supply voltage of 1.2 V. Furthermore, the validation result of the chip functionality by the behavioral digital simulation has shown that the proposed architecture realizes character recognition with unsupervised learning successfully.

In approximate arithmetic, a novel approximate arithmetic scheme to significantly reduce energy consumption with an extremely low error rate has been proposed for error resilient neuromorphic computing. The proposed carry speculation with a parallel carry-skip has been applied to both adder and comparator designs to considerably improve the overall error rate in computation and the critical path delay. Moreover, the error magnitude reduction technique for the adder further reduces the amount of error created by the approximate nature with low cost. The complete error rate analysis has proven that the proposed arithmetic units have extremely low error rates under random input patterns. The proposed approximate units have been implemented with the same 90 *nm* CMOS process. While the proposed adder exhibits 2.4× faster with an error rate of 0.18% and 43% more energy efficient over traditional ones, the proposed comparator has an error rate of less than 0.1% and achieves an energy saving of up to 4.9× over the conventional counterparts. To demonstrate the impacts of the approximate errors on the neuromorphic computing, we have conducted hardware-aware simulation of an unsupervised learning based VLSI neuromorphic character recognition chip that includes over a thousand of silicon neurons. The result has shown that the proposed approximate arithmetic units affect the training performance negligibly and outperform the other approximate adders. Furthermore, they allow for an energy reduction of up to 66.5% over traditional ones for the digital LIF computations during the learning process with scaled supply voltage levels.

Accordingly, the proposed architectural and circuit level design approaches are applicable to a wide range of energy efficient and error resilient neuromorphic computing systems, such as image and speech recognition, in VLSI.

6.2 Future Work

So far, we have demonstrated a neuromorphic processor configured as a 256 spiking neuron network in a single-die and able to successfully perform character recognition. Clearly, more complex networks are needed for other more sophisticated applications. Ultimately, one may think about integrating huge numbers of neurons and synapses to create an artificial brain that mimics the functions of the human brain such as reasoning, knowledge, planning, learning and memory. When an artificial human brain is implemented in silicon, tasks requiring complex reasoning and information processing as conducted by the humans may be readily solved with extremely short processing times. Also, artificial brains may allow people to better understand how the human brain works so as to advance cognitive science.

The CMOS technology scaling enables increasing numbers of neurons and synapses to be integrated in a given silicon area. Figure 6.1 demonstrates the number of neu-

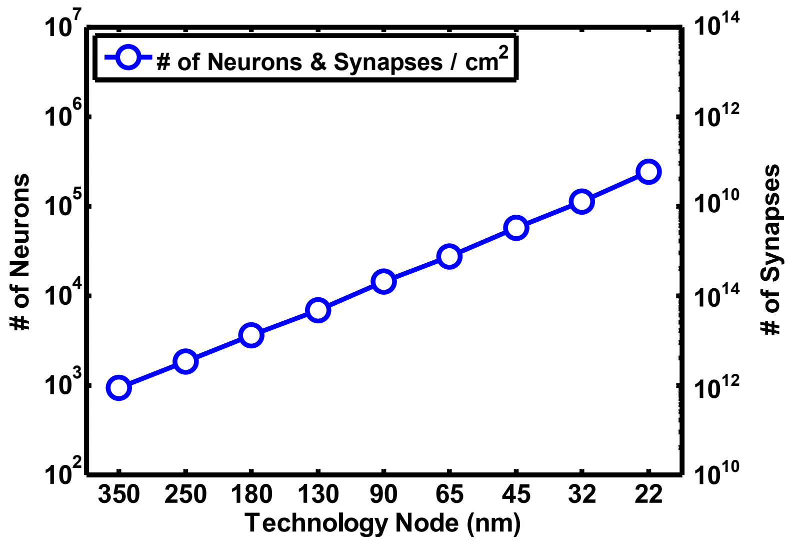


Figure 6.1: Neuron and synapse integration densities as a function of technology.

rons and synapses that can be integrated in a fixed area (*e.g.* 1 cm^2) at different technology nodes. They are evaluated based on our neuromorphic processor designed in Section 3. For the estimations, we take into account the area for only the CMOS switches in our CMOS/memristor hybrid cell since they dominate the overall area of the memristor synaptic crossbar array. Additionally, the entire chip area is assumed to be a linear function of the number of integrated silicon neurons N (*i.e.* chip area $\propto N$) since the memristive crossbar array, whose area is proportional to N^2, occupies very small portion (<10%) of the overall area. We also consider that the chip area is scaled with a factor of L^2 where L is the technology feature size. Then, we can not only estimate the area cost per a silicon neuron in the scaled CMOS technology but also obtain the numbers of integrated silicon neurons and synapses in a given silicon area. Note that the number of integrated synapses is N^2 in our crossbar structure. The number of integrated neurons approximately doubles for each new generation CMOS technology. At the 22 nm node, over 0.2 million (2×10^5) neurons and 50 billion (5×10^{10}) synapses, a complexity similar to the nervous system of ants, can be integrated in a 1 cm^2 silicon area. Obviously, technology scaling will continue in the coming decades. According to the International Technology Roadmap for Semiconductors (ITRS), the CMOS feature size (*i.e.* gate length) will reach 5.9 nm in 2026 and the supply voltage levels will decrease continuously as shown in Figure 6.2 [70]. We are able to predict the numbers of integrated neurons and synapses in a chip in the future from Figure 6.2. Figure 6.3 predicts the trend of the number of integrated neurons and synapses per 1 cm^2 of silicon area. Our neuromorphic processor and the same estimation method used in Figure 6.1 are also applied to the prediction. The predicted scaling trend exhibits that the number of silicon neurons in a cm^2 area will increase 25% $\sim$ 35% for every year from 2013 to 2026. In 2026, the 1 cm^2 silicon-die is estimated to include over 3 billion (3×10^6) neurons and 10 tril-

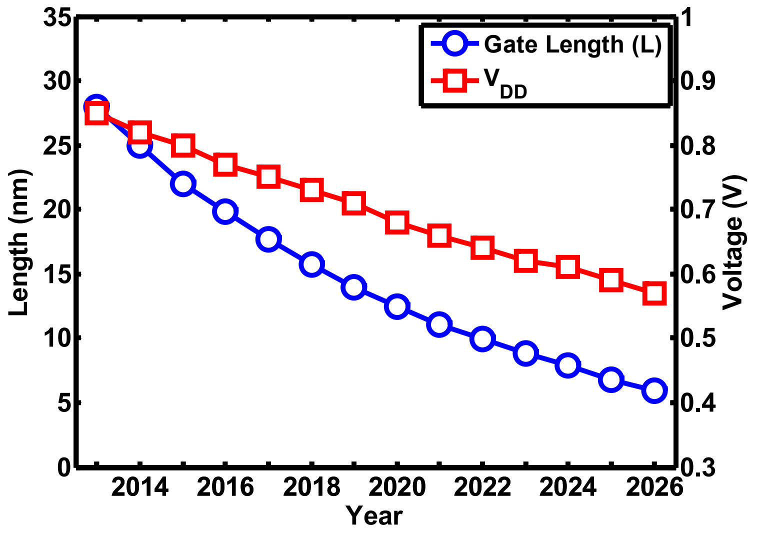


Figure 6.2: Trends of gate length and power supply [70].

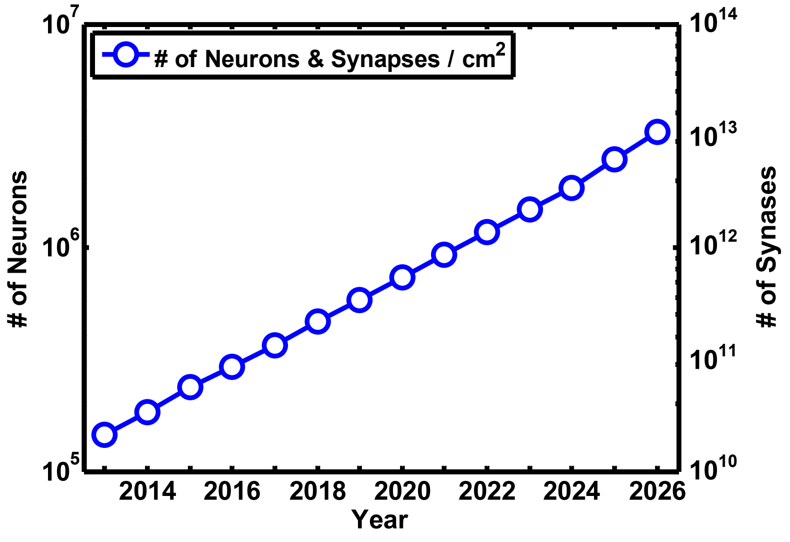


Figure 6.3: Scaling trend of neuron and synapse integration.

lion (10^{13}) synapses, which may be enough to mimic the cockroach's nervous system that contains a billion neurons. Approximately 5 and 17 cm^2 silicon-dies would be needed to emulate the brains of frogs (16 billion) and rats (56 billion), respectively.

To this end, it deserves to further optimize the existing neuromorphic hardware design to better deal with such design complexity. As an example, to compute the membrane potential of a neuron by accumulating a billion 3-bit pre-synaptic weights would require a 24-bit resolution for the proposed column ADC according to (3.6). It is very difficult to achieve a 24-bit resolution by the VCO-based ADC and other ADC architectures such as $\Delta\Sigma$ ADC may be considered. In a different direction, an alternative schemes to access the synaptic array may be investigated. In addition to hardware design, very importantly, appropriate learning algorithms and applications have to be developed to fully utilize the computing power of future neuromorphic chips integrating tremendous numbers of silicon neurons and synapses.

REFERENCES

[1] J. V. Arthur and K. Boahen. Silicon-Neuron Design: A Dynamical Systems Approach. *IEEE. Trans. Circuits Syst. I, Reg. Papers*, 58(5):1034–1043, 2011.

[2] J. V. Arthur, P. A. Merolla, F. Akopyan, R. Alvarez, A. Cassidy, S. Chandra, S. K. Esser, N. Imam, W. Risk, D. B. D. Rubin, R. Manohar, and D. S. Modha. Building Block of a Programmable Neuromorphic Substrate: A Digital Neurosynaptic Core. In *Proc. of Int. Joint Conf. Neural Netw. (IJCNN)*, pages 1–8, 2012.

[3] G.-Q. Bi and M.-M. Poo. Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type. *The Journal of Neuroscience*, 18(24):10464–10472, 1998.

[4] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan. A Learning-Enabled Neuron Array IC Based Upon Transistor Channel Models of Biological Phenomena. *IEEE Trans. Biomed. Circuits Syst.*, 7(1):71–81, 2013.

[5] R. A. Brualdi. *Introductory Combinatorics*. Prentice-Hall, Upper Saddle River, New Jersey, 2009.

[6] L. Camunas-Mesa, A. Acosta-Jimenez, C. Zamarrefio-Ramos, T. Serrano-Gotarredona, and B. Linares-Barranco. A 32×32 Pixel Convolution Processor Chip for Address Event Vision Sensors With 155 ns Event Latency and 20 Meps Throughput. *IEEE. Trans. Circuits Syst. I, Reg. Papers*, 58(4):777–790, 2011.

[7] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. Analysis and Characterization of Inherent Application Resilience for Approximate Comput-

ing. In *Proc. of IEEE/ACM Design Automation Conf. (DAC)*, pages 1–9, 2013.

[8] V. K. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. T. Chakradhar. Scalable Effort Hardware Design: Exploiting Algorithmic Resilience for Energy Efficiency. In *Proc. of IEEE/ACM Design Automation Conf. (DAC)*, pages 555–560, 2010.

[9] H. Cho, L. Leem, and S. Mitra. ERSA: Error Resilient System Architecture for Probabilistic Applications. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 31(4):546–558, 2012.

[10] L. O. Chua. Memristor-The Missing Circuit Element. *IEEE Trans. Circuit Theory*, 18(5):507–519, 1971.

[11] J. Cosp, J. Madrenas, and D. Fernandez. Design and Basic Blocks of a Neuromorphic VLSI Analogue Vision System. *Neurocomputing*, 69(16-18):1962–1970, 2006.

[12] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 31(7):994–1007, 2012.

[13] D. A. Drachman. Do We Have Brain to Sparse? *Neurology*, 64(12):2056–2062, 2005.

[14] K. Du, P. Varman, and K. Mohanram. High Performance Reliable Variable Latency Carry Select Addition. In *Proc. of Design, Automation, Test in Europe (DATE)*, pages 1257–1262, 2012.

[15] S. K. Esser, A. Ndirango, and D. S. Modha. Binding Sparse Spatiotemporal Patterns in Spiking Computation. In *Proc. of Int. Joint Conf. Neural Netw. (IJCNN)*, pages 1–9, 2010.

[16] D. E. Feldman. The Spike-Timing Dependence of Plasticity. *Neuron*, 75(4):556–571, 2012.

[17] R. FitzHugh. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophy. J.*, 1(6):445–466, 1961.

[18] S. Ghosh-Dastidar and H. Adeli. Third Generation Neural Networks: Spiking Neural Networks. In *Advances in Computational Intelligence*, pages 167–178, 2009.

[19] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 32(1):124–137, 2013.

[20] S. O. Haykin. *Neural Networks and Learning Machines.* Prentice-Hall, Upper Saddle River, New Jersey, 2008.

[21] R. Hegde and N. R. Shanbhag. Soft Digital Dignal Processing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 9(6):813–823, 2001.

[22] J. L. Hindmarsh and R. M. Rose. A Model of Neuronal Bursting using Three Coupled First Order Differential Equations. *Proc. R. Soc. Lond. B.*, 221(1222):87–102, 1984.

[23] Y. Ho, G. M. Huang, and P. Li. Dynamical Properties and Design Analysis for Nonvolatile Memristor Memories. *IEEE. Trans. Circuits Syst. I, Reg. Papers*, 58(4):724–736, 2011.

[24] A. L. Hodgkin and A. F. Huxley. A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve. *J. Physiol.*, 117(4):500–544, 1952.

[25] M. Hu, H. Li, Q. Wu, and G. S. Rose. Hardware Realization of BSB Recall Function using Memristor Crossbar Arrays. In *Proc. of IEEE/ACM Design Automation Conf. (DAC)*, pages 498–503, 2012.

[26] J. Huang, J. Lach, and G. Robins. A Methodology for Energy-Quality Tradeoff Using Imprecise Hardware. In *Proc. of IEEE/ACM Design Automation Conf. (DAC)*, pages 504–509, 2012.

[27] N. Imam, F. Akopyan, J. Arthur, P. Merolla, R. Manohar, and D. S. Modha. A Digital Neurosynaptic Core Using Event-Driven QDI Circuits. In *Proc. of IEEE Int. Symp. Async. Circuits and Syst. (ASYNC)*, pages 25–32, 2012.

[28] G. Indiveri, E. Chicca, and R. Douglas. A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses with Spike-Timing Dependent Plasticity. *IEEE Trans. Neural Netw.*, 17(1):211–221, 2006.

[29] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Haliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen. Neuromorphic Silicon Neuron Circuits. *Frontiers in Neuroscience*, 5(73):1–23, 2011.

[30] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artificial Neural Networks: a Tutorial. *Computer*, 29(3):31–44, 1996.

[31] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Letters*, 10(4):1297–1301, 2010.

[32] A. B. Kahng and S. Kang. Accuracy-Configurable Adder for Approximate Arithmetic Designs. In *Proc. of IEEE/ACM Design Automation Conf. (DAC)*, pages

820–825, 2012.

[33] R. Kempter, W. Gerstner, and J. L. Van Hemmen. Hebbian Learning and Spiking Neurons. *Physical Review E*, 59(4):4498–4514, 1999.

[34] J. Kim, T.-K. Jang, Y.-G. Yoon, and S. Cho. Analysis and Design of Voltage-Controlled Oscillator Based Analog-to-Digital Converter. *IEEE. Trans. Circuits Syst. I, Reg. Papers*, 57(1):18–30, 2010.

[35] S. H. Kim, S. Mukhopadhyay, and M. Wolf. Modeling and Analysis of Image Dependence and Its Implications for Energy Savings in Error Tolerant Image Processing. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 30(8):1163–1172, 2011.

[36] Y. Kim, Y. Zhang, and P. Li. A Digital Neuromorphic VLSI Architecture with Memristor Crossbar Synaptic Array for Machine Learning. In *Proc. of IEEE Int. System-on-Chip Conf. (SOCC)*, pages 328–333, 2012.

[37] S. Li, J.-H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures. In *Proc. of IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, pages 469–480, 2009.

[38] A. Lingamneni, C. Enz, K. Palem, and C. Piguet. Parsimonious Circuits for Error-Tolerant Applications through Probabilistic Logic Minimization. In *Lecture Notes in Computer Science*, pages 204–213, 2011.

[39] S.-L. Lu. Speeding Up Processing with Approximation Circuits. *Computer*, 37(3):67–73, 2004.

[40] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas. Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing

Applications. *IEEE Trans. Circuits Syst. I, Reg. Papers*, 57(4):850–862, 2010.

[41] H. Manem, J. Rajendran, and G. S. Rose. Design Considerations for Multilevel CMOS/Nano Memristive Memory. *ACM J. Emerg. Technol. Comput. Syst.*, 8(1):6:1–6:22, 2012.

[42] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs. *Science*, 275(5297):213–215, 1997.

[43] T. M. Massoud and T. K. Horiuchi. A Neuromorphic VLSI Head Direction Cell System. *IEEE. Trans. Circuits Syst. I, Reg. Papers*, 58(1):150–163, 2011.

[44] W. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[45] C. E. Merkel, N. Nagpal, S. Mandalapu, and D. Kudithipudi. Reconfigurable N-level Memristor Memory Design. In *Proc. of Int. Joint Conf. Neural Netw. (IJCNN)*, pages 3042–3048, 2011.

[46] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha. A Digital Neurosynaptic Core using Embedded Crossbar Memory with 45pJ per Spike in 45nm. In *Proc. of IEEE Custom Integrated Circuits Conf. (CICC)*, pages 1–4, 2011.

[47] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and Synthesis of Quality-Energy Optimal Approximate Adders. In *Proc. of IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, pages 728–735, 2012.

[48] S. Mitra, S. Fusi, and G. Indiveri. Real-Time Classification of Complex Patterns Using Spike-Based Learning in Neuromorphic VLSI. *IEEE Trans. Biomed. Circuits Syst.*, 3(1):32–42, 2009.

[49] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy. Design of Voltage-Scalable Meta-Functions for Approximate Computing. In *Proc. of Design, Automation, Test in Europe (DATE)*, pages 1–6, 2011.

[50] S. Moradi and G. Indiveri. A VLSI Network of Spiking Neurons with an Asynchronous Static Random Access Memory. In *Proc. of IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, pages 277–280, 2011.

[51] C. Morris and H. Lecar. Voltage Oscillations in the Barnacle Giant Muscle Fiber. *Biophy. J.*, 35(1):193–213, 1981.

[52] J. Nagumo, S. Arimoto, and S. Yoshizawa. An Active Pulse Transmission Line Simulating Nerve Axon. *Proc. IRE*, 50(10):2061–2070, 1962.

[53] Z. Pan and M. A. Breuer. Basing Acceptable Error-Tolerant Performance on Significance-Based Error-Rate (SBER). In *Proc. of IEEE VLSI Test Symp. (VTS)*, pages 59–66, 2008.

[54] H. Paugam-Moisy and S. Bohte. Computing with Spiking Neuron Networks. In *Handbook of Natural Computing*, pages 335–376, 2012.

[55] Y. V. Pershin and M. D. Ventra. Experimental Demonstration of Associative Memory with Memristive Neural Networks. *Neural Netw.*, 23(7):881–886, 2010.

[56] T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier. Is a 4-bit Synaptic Weight Resolution Enough? - Constraints on Enabling Spike-Timing Dependent Plasticity in Neuromorphic Hardware. *Frontiers in Neuroscience*, 6(90):1–19, 2012.

[57] J. B. Reece, L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, and R. B. Jackson. *Campbell Biology.* Benjamin Cummings, San Francisco, California, 2010.

[58] J.-S. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman. A 45nm CMOS Neuromorphic Chip with a Scalable Architecture for Learning in Networks of Spiking Neurons. In *Proc. of IEEE Custom Integrated Circuits Conf. (CICC)*, pages 1–4, 2011.

[59] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. C. Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, and B. Linares-Barranco. CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory-Processing-Learning-Actuating System for High-Speed Visual Object Recognition and Tracking. *IEEE Trans. Neural Netw.*, 20(9):1417–1438, 2009.

[60] B. Shim, S. R. Sridhara, and N. R. Shanbhag. Reliable Low-Power Digital Signal Processing via Reduced Precision Redundancy. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 12(5):497–510, 2004.

[61] G. S. Snider. Spike-Timing-Dependent Learning in Memristive Nanodevices. In *Proc. of IEEE/ACM Int. Symp. Nanoscale Arch. (NANOARCH)*, pages 85–92, 2008.

[62] S. Song, K. D. Miller, and L. F. Abbott. Competitive Hebbian Learning Through Spike-Timing-Dependent Synaptic Plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.

[63] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The Missing Memristor Found. *Nature*, 453:80–83, 2008.

[64] A. Syed, E. Ahmed, D. Maksimovic, and E. Alarcon. Digital Pulse Width Modulator Architectures. In *Proc. of IEEE Power Electron. Specialists Conf.*

(PSEC), pages 4689–4695, 2004.

[65] A. van Schaik. Building Blocks for Electronic Spiking Neural Networks. *Neural Netw.*, 14(6-7):617–628, 2001.

[66] J. Vanne, E. Aho, T. D. Hamalainen, and K. Kuusilinna. A High-Performance Sum of Absolute Difference Implementation for Motion Estimation. *IEEE Trans. Circuits Syst. Video Technol.*, 16(7):876–883, 2006.

[67] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan. MACACO: Modeling and Analysis of Circuits for Approximate Computing. In *Proc. of IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, pages 667–673, 2011.

[68] A. K. Verma, P. Brisk, and P. Ienne. Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design. In *Proc. of Design, Automation, Test in Europe (DATE)*, pages 1250–1255, 2008.

[69] J. H. B. Wijekoon and P. Dudek. Compact Silicon Neuron Circuit with Spiking and Bursting Behaviour. *Neural Netw.*, 21(2-3):524–534, 2008.

[70] L. Wilson. *International Technology Roadmap for Semiconductors.* SEMATECH, Albany, New York, 2011.

[71] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie. Design Implications of Memristor-based RRAM Cross-Point Structures. In *Design, Automation and Test in Europe (DATE)*, pages 1–6, 2011.

[72] J. J. Yang and R. S. Williams. Memristive Devices in Computing System: Promises and Challenges. *ACM J. Emerg. Technol. Comput. Syst.*, 9(2):11:1–11:20, 2013.

[73] Y.-G. Yoon, J. Kim, T.-K. Jang, and S. Cho. A Time-Based Bandpass ADC Using Time-Interleaved Voltage-Controlled Oscillators. *IEEE. Trans. Circuits Syst. I, Reg. Papers*, 55(11):3571–3581, 2008.

[74] N. Zhu, W. L. Goh, and K. S. Yeo. An Enhanced Low-Power High-Speed Adder for Error-Tolerant Application. In *Proc. of Int. Symp. Integrated Circuits (ISIC)*, pages 69–72, 2009.

[75] N. Zhu, W. L. Goh, and K. S. Yeo. Ultra Low-Power High-Speed Flexible Probabilistic Adder for Error-Tolerant Applications. In *Proc. of Int. SoC Design Conf. (ISOCC)*, pages 393–396, 2011.

[76] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong. Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 18(8):1225–1229, 2010.