On the Design of Adaptive Digital Neuromorphic System

Ogbodo Mark Ikechukwu

A DISSERTATION
SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN COMPUTER SCIENCE AND ENGINEERING

Graduate Department of Computer and Information Systems
The University of Aizu
2022

Declaration

I confirm that the research presented in this dissertation, except where I have specified otherwise, is my original reasearch. I also confirm that this dissertation has not been presented for any degree at any other university, and that I have not reproduced in part or whole; data, figures, graphs or information from other persons, researchers, or internet, unless otherwise acknowledged and cited as sourced from other persons, researchers, or internet.

Signed

グド オボ

Date

15/02/2022

The thesis titled

On the Design of Adaptive Digital Neuromorphic System

by

Ogbodo Mark Ikechukwu

is reviewed and approved by:

Chief referee *Professor* Abderazek Ben Abdallah	A. Ben Abdallah	[illegible]
Professor Tsuneo Tsukahara	T. Tsukahara	束原
Professor Junji Kitamichi	Junji Kitamichi	北道
Senior Associate Professor Saito Hiroshi	Hiroshi Saito	斎藤
Associate Professor Okuyama Yuichi	[illegible]	奥山

The University of Aizu
2022

Contents

List of Figures

List of Tables

List of Abbreviations

2D-NoC	Two dimensional Network-on-Chip
3D-IC	Three dimensional Integrated Circuit
3D-NoC	Three dimensional Network-on-Chip
3D-SIC	Three dimensional Stacked Integrated Circuit
ACT	Average Classification Time
ASIC	Application-Specific Integrated Circuit
ANN	Artificial Neural Network
BLoD	Bypass-Link-on-Demand
BC	Broadcast
CPU	Central Processing Unit
CT	Crossbar Traversal stage
DRAM	Dynamic Random Access Memory
ECC	Error Correction Codes
FIFO	First-In-First-Out
FT-KMCR	Fault-Tolerant K-means based MultiCast Routing algorithm
FTSP-KMCR	Fault-Tolerant Shortest-Path KMCR
HDL	Hardware Description Language
HF	Hopfield neural network
IC	Integrated Circuit
KMCR	K-means based MultiCast Routing algorithm
MC	Multicast
NoC	Network-on-Chip
PE	Processing Element
PNU	Parallel Neuron Update
PWU	Parallel Weight Update
RAB	Random-Access-Buffer
RC	Routing Computation stage
RNDC	Randomly Connected Neural Network
RTL	Register-Transfer Level
SA	Switch Allocation stage
SAW	Spike Arrival Window
SRAM	Static Random Access Memory
SNN	Spiking Neural Network
SNPC	Spiking Neural Processing Core

SoC	System-on-Chip
SP-KMCR	Shortest- Path K-means based MultiCast Routing algorithm
TSV	Through Silicon Via
UC	Unicast

To my father El Shaddai.

Acknowledgments

I would like to express my deepest appreciation to my supervisor Prof. Abderazek Ben Abdallah for his patience, support, encouragement, and guidance. I have benefited immensely from his wealth of knowledge, invaluable advice, and meticulous instructions.

I would also like to extend my sincere gratitude to Prof. Tsuneo Tsukahara, Prof. Junji Kitamichi, Prof. Saito Hiroshi, and Prof. Okuyama Yuichi for generously taking the time out of their busy schedules to revise my thesis. Their thoughtful and detailed feedback have been very important in completing this thesis.

Special thanks to Dr. Akram Ben Ahmed of AIST, and Dr. Khanh Nam Dang for taking out time from their schedules to have discussions with me and answer many of my questions. To my colleagues Fukuchi and Wang, and the other members of Adaptive Systems Laboratory, thank you for your great assistance throughout the period of my PhD program. Not forgeting the entire staff of the University of Aizu who contributed in making my study experience and settling into Aizuwakamatsu seamless, thank you.

The completion of my studies would not have been possible without the scholarshp support from Rotary Yoneyama Memorial Foundation, and Kitakata Chuo Rotary club. Their financial support helped me to maximize my study experience by focusing on my research, and their social activities provided me the opportunity to have a balanced academic and social life. I must also thank my counselor Mr. Kouichi Hoshi for his care and guidance.

I cannot begin to express my gratitude to my parents; Mr and Mrs Leonard Ogbodo, who raised me and have strongly supported me all my life. Their guidance and prayers have inspired me to reach for greater heights. Many thanks to my siblings, mentors, and friends who supported me, and generously spoke kind encouraging words that helped me through challenging times, and motivated me to work harder to achieve set goals.

Thesis advisor: Professor Abderazek Ben Abdallah Ogbodo Mark Ikechukwu

On the Design of Adaptive Digital Neuromorphic Systems

Abstract

With the increasing demand for computing machines that are analogous to the biological brain, the field of neuro-inspired computing has advanced to the exploration of neuromorphic architectures that best the limitations of the traditional computer systems. Traditional computer systems are based on the von Neumann architecture whose limitation have diminished further development in such systems. The biological brain, however, shows a disparity in structure, computational power, and power consumption, when compared to the traditional computer system. Therefore, a biologically inspired approach to computing, can potentially address some of the limitations faced in traditional computing systems.

Spiking neural network (SNN) has gradually gained awareness by reason of its ability to process and communicate sparse binary signals (spikes) in a highly parallel and event driven manner analogous to the biological brain. However, simulating large scale SNN in software is slow, and does not fully harness the energy efficiency of SNN. As an alternative, scalable multicore spike-based neuromorphic architectures that can support massive number of neurons and synapses, and leverage the spike sparsity available in SNN to deliver rapid parallel processing with low power are being proposed. Nevertheless, realizing such a neuromorphic architecture requires building small-sized spiking neuro-cores with low-power consumption, efficient neural coding scheme, and lightweight on-chip learning. In this dissertation, we propose an adaptive digital neuromorphic system (NASH) which leverages several design approaches to realize low power and adaptivity in a scalable architecture.

First, to ensure efficient and low power spike processing, the proposed NASH utilizes light-weight spiking neuro processing cores that employ a parallel neuron update (PNU) mechanism proposed in this dissertation to enable rapid parallel spike processing. Secondly, for efficient learning, a neuromorphic learning framework that enables NASH to explore various SNN learning approaches is employed. This learning framework also enables efficient implementation of an on-chip learning rule which utilizes a parallel weight update (PWU) mechanism proposed in this dissertation.

One additional issue that need to be addressed in realizing a spike-based neuromorphic system capable of supporting large SNN, is the need for a scalable interneuron communication architecture that can support the enormous amount of traffic generated by massive number of interconnected neurons. Also, highly connected neuromorphic architectures

encounter the reliability issue where a single point of failure can affect operation. Since neuromorphic systems rely heavily on spike communication, and an interruption in the timing of spike communication can adversely affect its performance, adaptation is necessary.

In addressing the issue of scalability and adaptation, NASH integrates spiking neuron processing cores in a fault-tolerant three-dimensional network on chip (3D-NoC) communication architecture which provide higher level of scalability, parallelism, low communication cost, and reliability when compared to communication architectures such as shared bus and two-dimensional network on chip (2D-NoC).

Finally, experiments are performed to evaluate the efficiency of NASH, and the result of these experiments are compared to some baseline neuromorphic systems.

適応型デジタルニューロモルフィックシステムの設計について

概要

生物学的脳に類似した計算機の需要が高まるにつれ、神経創発コンピューティングの分野は、従来のコンピュータシステムの限界を最もよく満たす神経形態学的アーキテクチャの探求に進んでいます。従来のコンピュータシステムはフォンノイマンアーキテクチャに基づいており、その制限により、このようなシステムのさらなる開発が減少しています。しかし、生物学的脳は、従来のコンピューターシステムと比較した場合、構造、計算能力、および消費電力に格差を示しています。したがって、生物学的触発コンピューティングへのアプローチは、従来のコンピューティングシステムが直面する制限のいくつかに対処できる可能性があります。

スパイキングニューラルネットワーク（SNN）は、生物学的脳に類似した高度に並列でイベント駆動型の方法でスパーススバイナリ信号（スパイク）を処理および通信する機能により、徐々に認識を高めています。ただし、ソフトウェアで大規模な SNN をシミュレートするのは遅く、SNN のエネルギー効率を十分に活用していません。別の方法として、膨大な数のニューロンとシナプスをサポートし、SNN で利用可能なスパイクのスパース性を活用して、低電力で高速な並列処理を実現できる、スケーラブルなマルチコアスパイクベースのニューロモルフィックアーキテクチャが提案されています。ただし、このようなニューロモルフィックアーキテクチャを実現するには、低消費電力、効率的なニューラルコーディングスキーム、および軽量のオンチップ学習を備えた小型のスパイクニューロコアを構築する必要があります。この論文では、スケーラブルなアーキテクチャで低電力と適応性を実現するために、いくつかの設計アプローチを活用する適応型デジタルニューロモルフィックシステム（NASH）を提案します。

まず、効率的で低電力のスパイク処理を保証するために、提案された NASH は、この論文で提案された並列ニューロン更新（PNU）メカニズムを採用する軽量スパイク神経処理コアを利用して、迅速な並列スパイク処理を可能にします。第二に、効率的な学習のために、NASH がさまざまな SNN 学習アプローチを探索できるようにするニューロモルフィック学習フレームワークが採用されています。この学習フレームワークは、この論文で提案されている並列重み更新（PWU）メカニズムを利用するオンチップ学習則の効率的な実装も可能にします。

大規模な SNN をサポートできるスパイクベースのニューロモルフィックシステムを実現する際に対処する必要があるもう 1 つの問題は、相互接続された多数のニューロンによって生成される膨大な量のトラフィックをサポートできるスケーラブルな介在ニューロン通信アーキテクチャの必要性です。また、高度に接続されたニューロモルフィックアーキテクチャでは、単一障害点が動作に影響を与える可能性があるという信頼性の問題が発生します。ニューロモルフィックシステムはスパイク通信に大きく依存しており、スパイク通信のタ

イミングの中断はそのパフォーマンスに悪影響を与える可能性があるため、適応が必要です。
NASH は、スケーラビリティと適応の問題に対処する際に、スパイクニューロプロセッシングコアをフォールトトレラントな 3 次元ネットワークオンチップ（3D-NoC）通信アーキテクチャに統合し、共有バスや 2 次元ネットワークオンチップ（2D-NoC）などの通信アーキテクチャと比較した場合、より高いレベルのスケーラビリティ、並列処理、低通信コスト、および信頼性を提供します。
最後に、NASH の効率を評価するために実験が実行され、この実験の結果がいくつかのベースラインニューロモルフィックシステムと比較されました。

1 Introduction

Traditional computer systems have significantly advanced over the years, demonstrating tremendous performance in high precision numerical computations. They are based on the Von Neumann architecture, which is good at solving such numerical computing problems. However, they are faced with a number of limitations which potentially stall further improvements that are required to meet recent growing computational needs. In overcoming these limitations, a neuro-inspired computing paradigm that models the computational principles of the brain, could potentially serve as a bridge to the next computational advancement. In this chapter, we first discuss the architecture and limitations of traditional computers, then the brain's computational principles that are modeled to overcome these limitations. Next, we discuss brain-inspired systems and some of the hurdles of realizing them. These hurdles translate to the motivation for this research, and to surmount them, the dissertation objectives and contributions highlight the proposed system. Finally, the chapter is concluded with the dissertation outline.

1.1 Neuro-inspired Computing: Beyond Traditional Computers

1.1.1 Traditional computing

The Von Neumann architecture as described in Figure 1.1, has the processing and memory unit separate. Data and instructions are stored in memory, and the processing unit has to communicate via busses with the memory to fetch and execute instructions on the data. The throughput of the memory however, is lower than the rate at which the processing unit can work, and consequently has become a bottleneck in traditional computer systems [12]. While the traditional computer system have been able to demonstrate good performance at high precision numerical computations, they are not able to carryout tasks that require cognition.

Decades ago, it was observed that exponential growth in the performance of traditional computer systems can be achieved by increasing the number of transistors on a single chip. This observation was made in 1965 by Gordon Moore, the cofounder of intel, and based on it, he postulated that the integration density of transistors will double every 18 months, increasing performance, and reducing cost. This postulate known as moore's law has been the feed-stock of the exuberant advancement in the performance of traditional computer systems. However in recent years, Moore's law no longer holds to be true as it did decades ago. This is because increasing the number of transistors on a chip no longer increases performance at the expected rate. As desribed in Figure 1.2, the challenge of quantum tunneling which arise from shrinking transistors make such performance unattainable. Moreover, further increasing the number of transistors on a single chip increases the power consumption and heat, as can be seen in Figure 1.3, making it very difficult to cool.

With the end of Moore's law in sight, a new computing approach is required to drive the next wave of computer systems. One promising approach is Neuro-inspired computing which draws inspiration from the biological brain. Studies from neuroscience show that the brain assumes a different architecture and computational principle from traditional computer systems. The brain is composed of billions of neurons which are highly interconnected via synapses. These neurons combine memory and processing, and are able to communicate among themselves using short

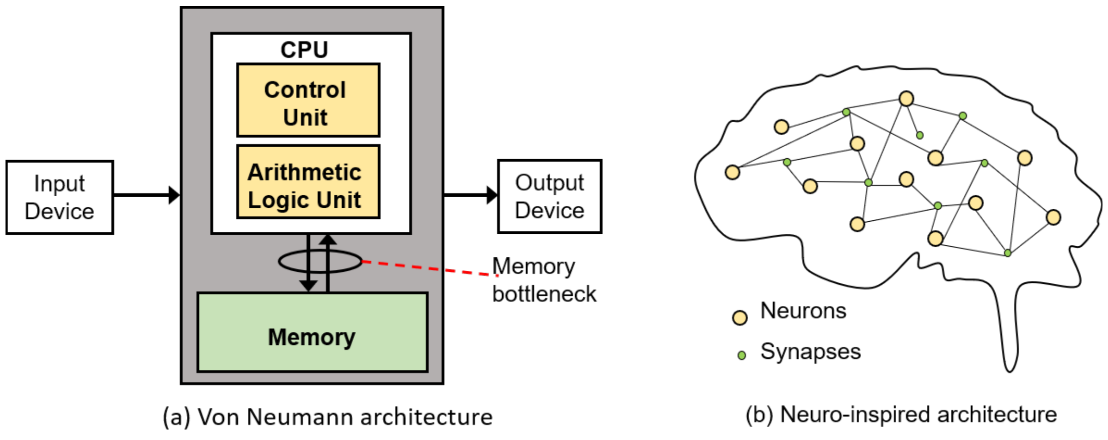


Figure 1.1: Computation paradigm is shifted from (a) von Neumann (centric computation) to (b) brain-inspired computing (distributed computation).

electrical signals known as spikes. This architecture and computational principle enable the brain to learn, and perform complex operations in a rapid, parallel, fault-tolerant, and energy efficient manner.

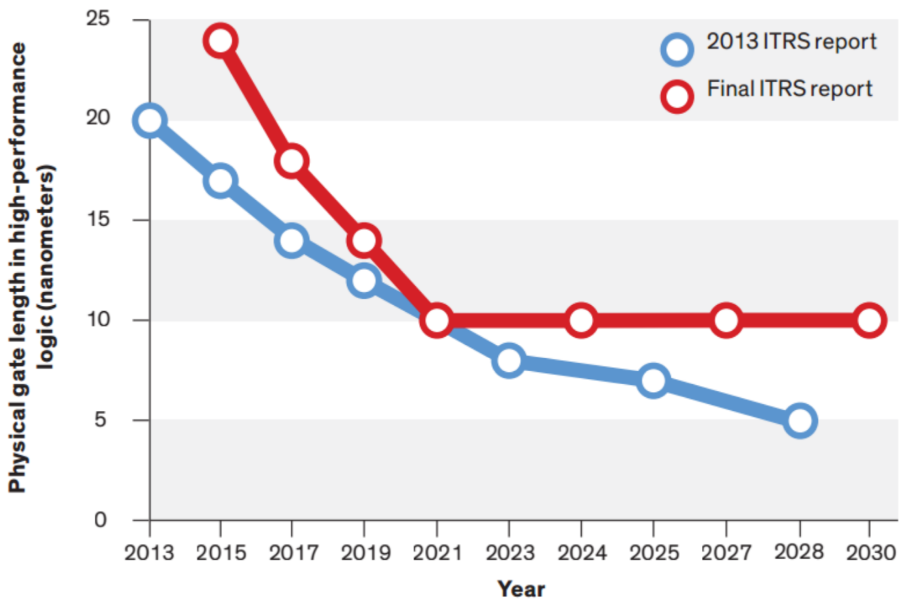


Figure 1.2: End of the road; a shringking challenge of of physical gate length at 10nm [1].

1.1.2 Neuro-inspired Computing

Neuro-inspired computing aims to mimic the achitecture and computational principle of the biological brain to realize computers that learn and operate in a similar manner. Over the years,

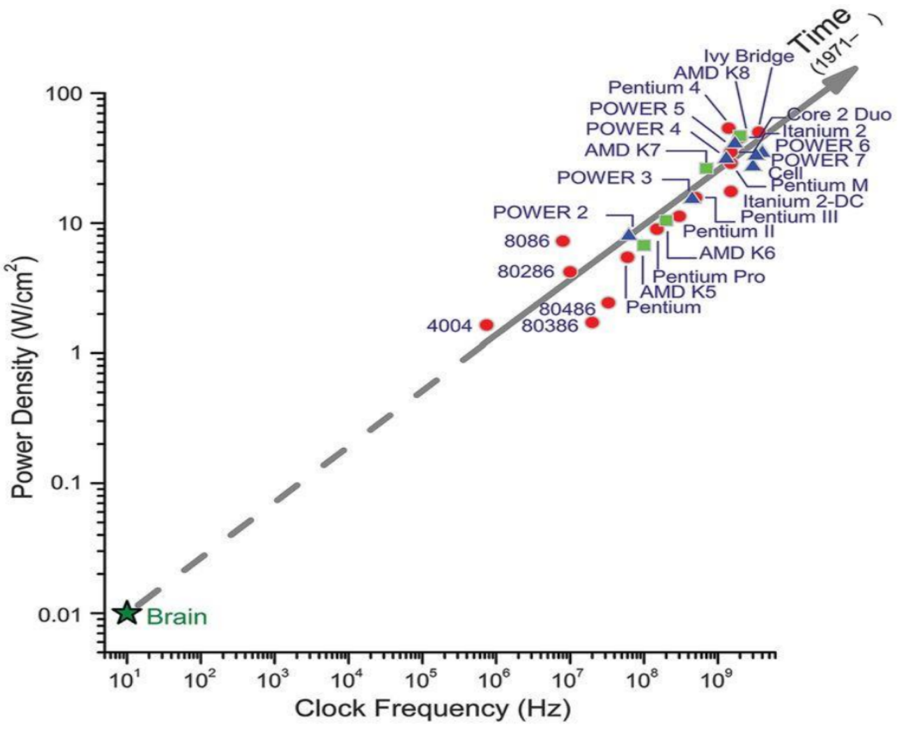


Figure 1.3: Power density and clock frequency limitations in traditional computer systems [2].

two notable approaches have been employed in mimicing this principle. The first approach which is a non spiking approach, is termed artificial neural network (ANN). Two known categories of this approach include recurrent neural network (RNN) and convolutional neural network (CNN). Several models of ANN have shown tremendous performance in tasks like classification, clustering, pattern recognition and prediction. These models which are trained using graphical processing unit to enable them perform tasks, have significantly grown in size over the years, inculcating more layers to form a deep neural network (DNN). This increase in size enable them carry out more complex operations. Models like the Residual Network (ResNet) proposed a 152-layer CNN which enabled it to achieve high accuracy on the ImageNet dataset in 2015. While these performance are admirable, the computational cost of these large models are quite high. For example, the implementation of Google's autoencoder [13] required a cluster of 16K processing cores, and consumes 100 kW of power to successfully recognize faces of cats from ten million images captured from YouTube videos.

The second approach which have been employed in mimicking the brain's computational principles is spiking neural network (SNN). This approach mimics more closely, the behavior of biological neural networks, and communicates via spikes in an event driven manner. In encoding information as spikes, SNN employs several coding schemes that include rate coding, population coding, and temporal coding [14]. Several spiking neuron models exist, and the prevalent ones which are often found in typical SNNs are the integrate and fire (IF) model [15] and its variants which include the leaky integrate and fire (LIF). The neuronal dynamics of this model can be thought of as an integration process, together with a spiking mechanism. Typical spikes irrespective of their amplitude and shape are handled as similar events, and from the outset to finish, lasts about two milliseconds [16] traveling down axonal lengths. Another spiking neuron model which is noted for its detailed modeling of biological neuron is the Hodgkin and Huxley model [17]. This model is nonlinear and stochastic. However, it is complex, making it less ideal for large-scale simulation.

Software simulation of SNN [18] has shown to be a flexible way of exploring the behavior of neuronal systems. However, simulating deep SNN in software is slow and consumes much power, making it less suitable for both implementing real-time systems and precise large-scale simulations of neural systems. Hardware implementation (neuromorphic system) on the other hand, provides an alternative approach that holds the potential for independent spikes to be generated accurately, and output spikes to be simultaneously fired in real-time. Also, hardware implementation has the edge of computational acceleration over software simulations, and peculiar multi-neuro-core neuromorphic architectures can leverage the parallelism and spike sparsity available in SNN to deliver rapid processing with low power.

1.1.3 Sparsity in Spiking Neural Network

Studies have shown that neural activity and connectivity in the neocortex of the biological brain are immensely sparse [19]. It is reported that about 0.5% to 2% of neural cells are active at any given moment, and about 1% to 5% of connections exist between connected layers of neurons. 30% of these connections are reported to change every few days. These dynamics of the brain enable it to demonstrate extemely low power consumption, and dynamic structural plasticity in the neocortex [19]. Given that SNN operates in a manner more analogous to these neurons, deep SNNs with such level of sparsity in connections and neuronal activity can demonstrate low power

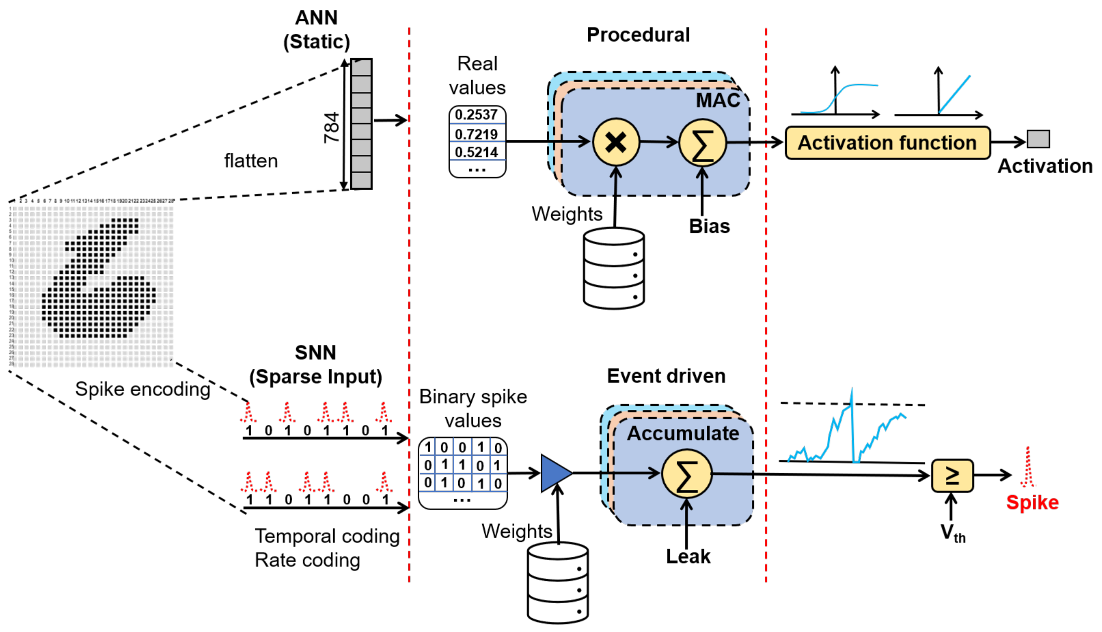


Figure 1.4: A Comparison of ANN and SNN implementations for a handwritten digit recognition application.

consumption.

Given a sparse neural network as illustrated in Figure 1.5(b), inactive neurons at any point can be power gated to reduce power consumption. However with all neurons active in the dense network as described in Figure 1.5(a), the power consumption becomes higher. Significant energy and acceleration can be gained while keeping performance the same when sparsity is employed [20].

1.2 Motivation

In adopting the structure and computational principle of the brain coupled with the benefits that hardware implementation provides, neuromorphic computing has the potential to provide solution beyond the limitations faced by traditional computers. This structure and computational principle also enable SNN to perform real-time cognition tasks which the traditional computer is not good at. However, the design of an efficient neuromorphic system is met with a number of hurdles that need to be surmounted. In this research three of those are considered, and they form the motivation for this research.

The first hurdle considered in this research is power, and the motivation for this stems from the understanding that the biological brain is postulated to operate at one exaflop with just 20

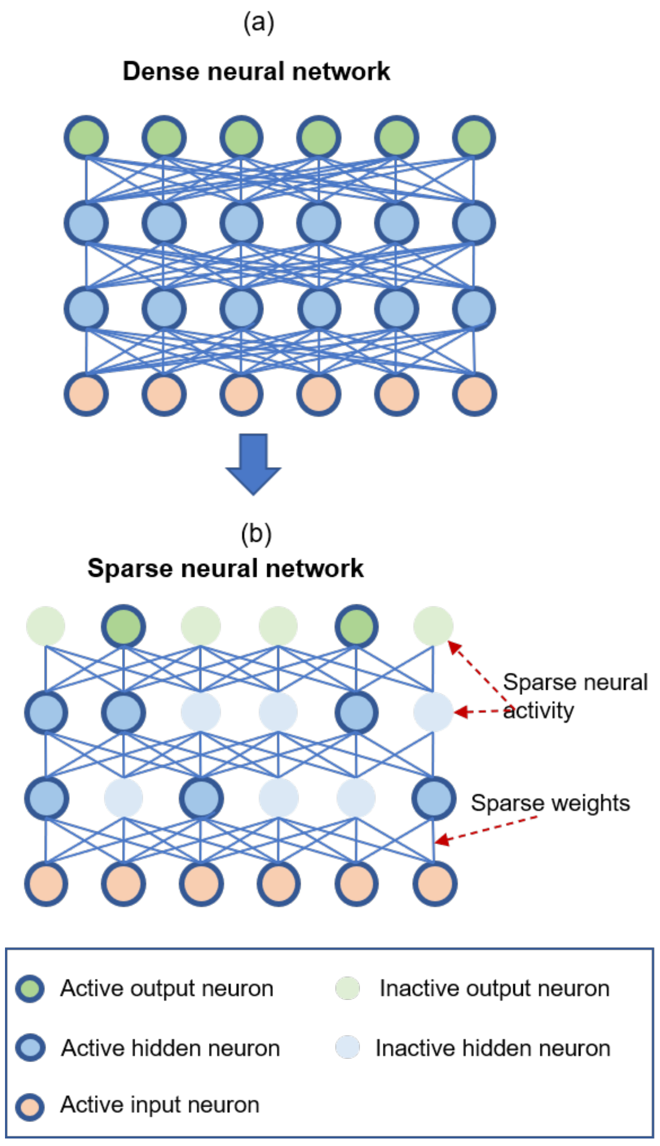


Figure 1.5: Illustration of sparsity in spiking neural network.

watts. Therefore, the need to aim towards low power and energy efficiency in neuromorphic system design. In contrast to ANN where all neurons fire at every propagation cycle, spiking neurons fire only when their voltage potential are stimulated beyond a threshold value [21]. As described in Figure 1.4, ANN operation for classifying an image requires a single feed-forward pass through the network, while SNN has to be appraised over a number of time-steps. However, peculiar hardware that computes only when needed, can allow for event-driven neural operation, and leverage it to demonstrate low power. It has been observed that spiking activity becomes more sparse with increase in network depth [22]. Therefore, an increment in energy gains is expected in such event-driven hardware with increase in network depth. For ANN, the amount of synaptic operations (SoP) in a single network layer can be determined from the structure of the layer, while that of SNN is the dot-product of the aggregate number of spikes in a layer and the corresponding number of SoPs. A single SoP in ANN requires a multiplyaccumulate (MAC) operation, while SNN perform only accumulate operation when an input spike is received. With MAC operations being more complex, more energy is required for it, compared to accumulate operation [23]. Thus, the energy consumption of SNN is expected to be much less when compared to ANN. It needs to be noted that while SNN operation requires a number of time-steps as opposed to ANN, the actual time needed to perform a single time-step in a neuromorphic system might be much less compared to a feedforward operation for ANN, and despite the delay overhead, the power gains of event-driven neuromorphic systems can significantly boost the energy efficiency of deep SNNs in comparison to ANN [22].

The second hurdle is scalability, and the motivation for this comes from the brain's ability to densely house billions of highly connected neurons. To design in silicon, a neuromorphic system capable of housing even a fraction of the number of neurons in the brain, and sustaining the traffic of their communication, a scalable inter-neuron communication architecture is required. Furthermore, since the timing of spikes is used to encode information in SNN, such inter-neuron communication architecture should not violate the timing of spikes, as this will afftect the performance of the SNN. There are various communication interconnects that could be employed when designing inter-neuron communication architectures, and they include shared bus and packet-switched network on chip (NoC) [24]. A shared bus however, is a poor choice when implementing a large-scale SNN since it suffers adversely with increased number of nodes. The nonlinear increase in neural connectivity will be too much for such an interconnect to handle. An interconnect that has been considered as a potential solution is the 2D packet-switched NoC (2D-NoC). However, with further scaling, 2D-NoC interconnects begins to experience communication challenges that affect power and performance, especially in large-scale SNN chips. 3D packet-switched NoC (3D-NoC), on the other hand, enables scaling and parallelism in the third dimension by combining NoC and 3D ICs (3D-ICs) [24]. With the help of its short through silicon vias (TSVs) that enable communication between layers, it can reduce communication costs. These merits of 3D-NoC make it suitable for large-scale SNN applications. Moreover, the brain is biologically organized in a 3D structure; therefore, by adopting the 3D interconnect, neuromorphic systems can inherit the shape, and the interconnects of a biological brain.

Finally, the third hurdle considered in this research is reliability. Despite being known for having some underlying fault-tolerance attribute resulting from their densely parallel framework, SNNs face some fault challenges, especially those assumed from implementing them in hardware [25]. When faults occur in inter-neuron connection the firing rate of post synaptic neurons are affected, and they become silent or near silent [26]. Such faults can have adverse effect especially in critical applications such as autonomous driving and biomedical devices. To handle such faults, an efficient fault-tolerant approach needs to be considered.

1.3 Dissertation Objectives and Contributions

In this dissertation, we present an adaptive digital neuromorphic system that leverages efficient spiking neuron processing cores to exploit the inherent 3D structure of the brain on a fault-tolerant 3D-NoC based architecture to achieve high scalability, small foot print, and rapid parallel spike processing with low power. The main contributions of this dissertation are summarized as follows:

1. **A light-weight Spiking Neuron Processing Core that enables rapid parallel spike processing, suitable for 3D-NoC-based spiking neuromorphic architecture**. The goal is to provide small-sized spiking neuro-cores with rapid processing, low-power consumption, efficient neurocoding scheme; which is the backbone of the proposed digital neuromorphic system. The rapid parallel spike processing is achieved using a parallel neuron update (PNU) algorithm presented in this dissertation.

2. **A neuromorphic learning framework which explores several SNN learning approaches on the proposed adaptive digital neuromorphic system**. This learning framework enables several learning approaches, inclusing rapid on-chip learning with trace based spike time dependent plasticity (STDP) using a parallel weight update (PWU) mechanism presented in this dissertation.

3. **Integration of spiking neuron processing cores into an adaptive scalable neuron communication architecture to realize an adaptive digital neuromorphic system.**. Adopting the anatomical topology of brain which is an interconnection of small works requires the synapses of an SNN to be partitioned and mapped into interconnected neuro cores that represent the small world networks. The 3D-NoC based spiking neuromorphic architecture provides this topology, by integrating the spiking neuron processing cores into a scalable interconnect that provides efficient interneuron communication.

1.4 Dissertation Outline

The rest of this dissertation is organized as follows:

- In chapter 2, A review of neural network backgroud, its evolution, topologies, and learning approaches are presented.
- In chapter 3 A survey of important related works which describe different neuromorphic systems is presented. We focus mainly on their architecture targeting scalability. Additionally, we also present a survey of fault-tolerance in neural network systems.
- Chapter 4 Describes the parallel neuron update (PNU), and parallel weight update (PWU) mechanism employed the the Spiking neuro core design. The neuromorphic learning framework which explores SNN learning approaches is described.
- Chapter 5 presents and evaluate the architecture of the light-weight spiking neuro core which serves as the backbone of the proposed system.
- Chapter 6, describes the architecture and design of the proposed adaptive digital neuromorphic system. We describe the scalable 3D-NoC based communication architecture, learn-

ing, and the integration process of realizing the proposed system. Finally, the performance and complexity of the proposed system is evaluated.

- Finally in Chapter 7, we present the conclusion that has been drawn from this research, and discuss guidelines for future works.

2

Brain Inspired Computational Principle: Background

The neuro-biological system is made up of roughly 85 billion neurons which form the elementary processing unit behind most of the unique operations of the brain. These neurons are connected to each other in an intricate pattern. They percieve changes in the environment, convey these changes to other neurons, and directs body responses to these perceptions [16]. Because these neurons are able to carryout information processing in a rapid, parallel,fault tolerant and energy efficient manner, they have received so much attention, and become and inspiration in designing computational systems. In this chapter, We first discuss biological neurons and the dynamics that are abstracted from them to model artificial

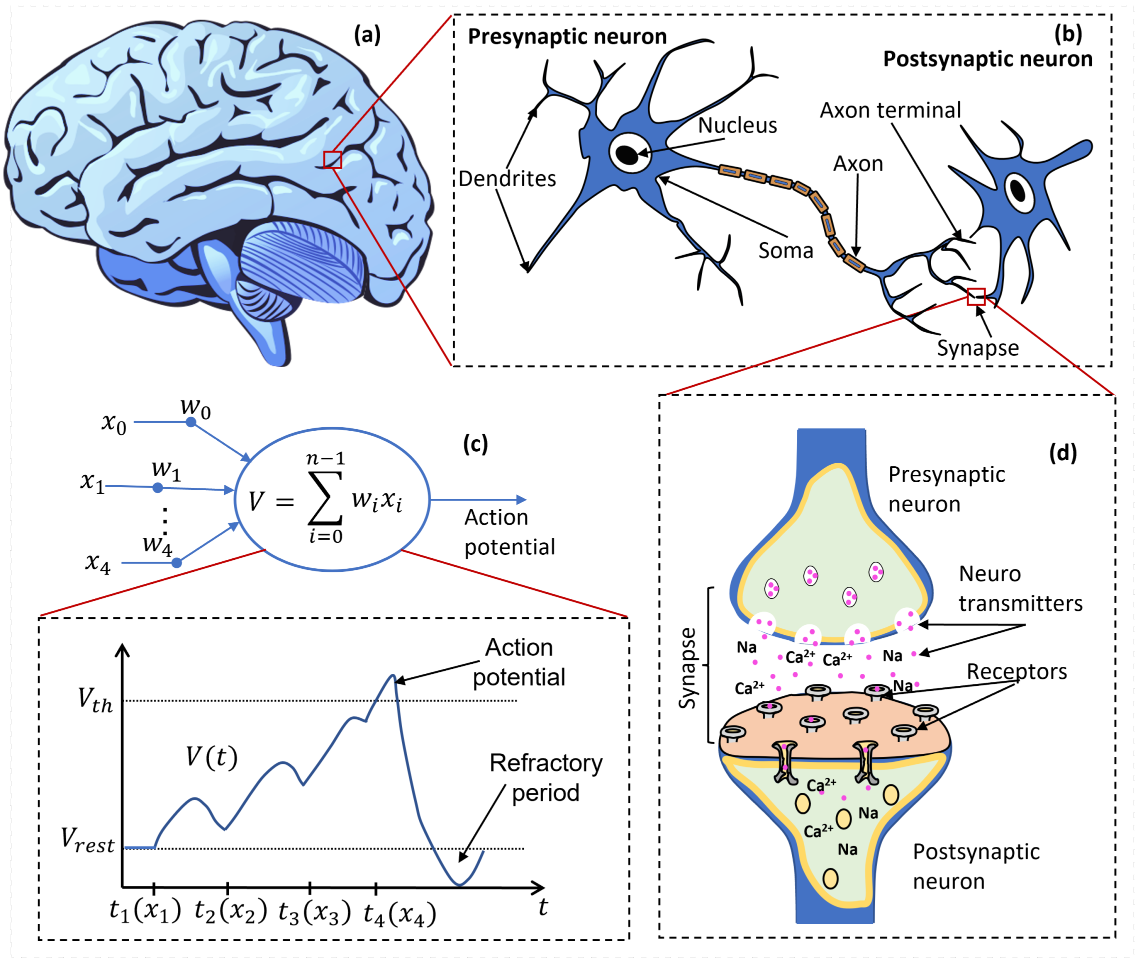


Figure 2.1: Brain inspired computational priciple: (a) The mamalian brain [3]. (b) Neurons. (c) Mathematical model of a neuron. (d) Synapse.

neurons. Next we discuss artificial neurons and how they have evolved in their representation of biological neuronal dynamics. Afterwards, we discuss the implementation of these neural networks in terms of neuron models, learning, and implementation approaches.

2.1 Neural Network

A crude description of a biological neuron is presented in Figure 2.1. From it, we can see that the neuron consists of several parts: The dendrite, axon, cell body which is called soma, and the synapse. The dendrite serves as input channel to the neuron, and receives information as electrochemical signals (action potential/ spike) from other neurons (presynaptic neurons) and transmit to the soma for processing [16]. The soma serves as the central processing unit, performing nonlinear processing operations by integrating the received action potentials into its membrane potential

V(t). until it crosses a threshold V_{th} and activates (fire) an action potential (spike). The axon takes on the role of an output device, and transmits the spike to other neurons (postsynaptic neurons) whose dendrites are connected to any of the axon terminals. This connection among neurons is enabled via the synapses. The neuro-biological system is formidably connected. A typical cortical neuron is connected to up to 10_4 presynaptic neurons, and some cerebellar neurons are connected to up to a quarter of a million presynaptic neurons [16].

A spike fired from the axon of a presynaptic neuron has an amplitude of about 100 mV and and last a duration of about 1-2 ms arriving the postsynaptic neuron [5]. It is considered the primary unit of information communication among neurons. A group of spikes pattern fired by a single neuron at regular or irregular intervals is called spike train, and retain their shape as they propagate along the axon [5]. The shape of all spikes from a neuron are similar, and remain the same as they propagate along the axon. As a result, spike shape do not carry any information. However, information is embedded in the the frequency of spikes, and in the precise timing of each spike. Neuroscience stusies have shown that when a cortical neuron repeatedly receives similar fluctuating input spikes, the timing of its output spikes are strikingly precise over repeated trials, and the output spike pattern seem to be unique. As a result, neurons have been shown to generate different spike patterns depending on their input spike history [27].

After a neuron fires a spike, it enters a refractory period as shown in Figure 2.1(c) where further spike cannot be fired immediately. As shown in Figure 2.1(b), the site where the axon of a presynaptic neuron and the dendrite of a postsynaptic neuron meet is called the synapse. This is where neurotransmitters (Na and Ca^{2+}) triggered by a spike are released by the presynaptic neuron. These neurotransmitters are detected by the receptors on the dendrite of the postsynaptic neuron, and this causes them to open. The open neurotransmitters then allow the influx of ions that increases the membrane potential of the postsynaptic neuron [16].

Figure 2.1(c) describes a computational neuron model. Input signals (e.g., x_0) received from the axon of other neurons are multiplied with the weight of the synapse that connects them (e.g., w_0). weighted inputs (e.g., w_0x_0) are then transported by the dendrite to the soma of the receiving neuron. At the soma, the weighted inputs are summed up as the neuron membrane potential, and passed through an activation function that maps it to the output of the neuron.

Over the years, several artificial neural network modeling approach have been proposed, differ-

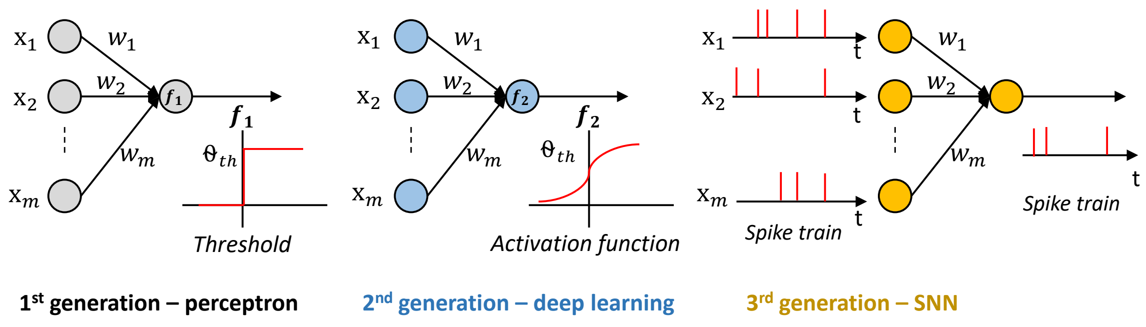


Figure 2.2: Generations of artificial neural network [4].

ing in topology and features so as to capture the dynamics of neural computation. As described in Figure 2.2, these modeling aproach have evolved through three generations, while emulating the computational principles of the biological brain. In the **first generation:**, the neurons were referred to as perceptrons. These perceptrons process only digital signals using a single layer. The sum of weighted inputs of this neuron is mapped to the neuron output using binary threshold. Some exmaples of perceptrons include Hopfield networks, and Boltzmann machines. The **second generation** neurons are interconnected to have conventional artificial neural network. This generation of neurons map the sum of their weighted inputs to their output using activation functions such as sigmoid, exponential, and polinomial, which have continuous set of possible outputs. Also, this second generation network employ learning algorithms which are based on gradient descent. Examples of this generation of neural netwok include feedforward, radial basis function units and recurrent sigmodial neural network. The **third generation** which is referred to as spiking neural network is modeled in a manner more analogous to the dynamics of biological neurons compared to previous generations. It is event-driven, and operates by accumulating input spikes at its membrane potential. An output spike is fired by the neuron only when its membrane potential exceeds a certain threshold.

2.1.1 Neural Network Topologies

To perform tasks, artificial neurons like their biological counterparts need to be interconnected. The manner in which neurons are interconnected determines their topology. A summary of some neural network topologies illustrated in Figure 2.3 are described as:

- **Feed forward neural networks:** This network topology described in Figure 2.3(a) is orga-

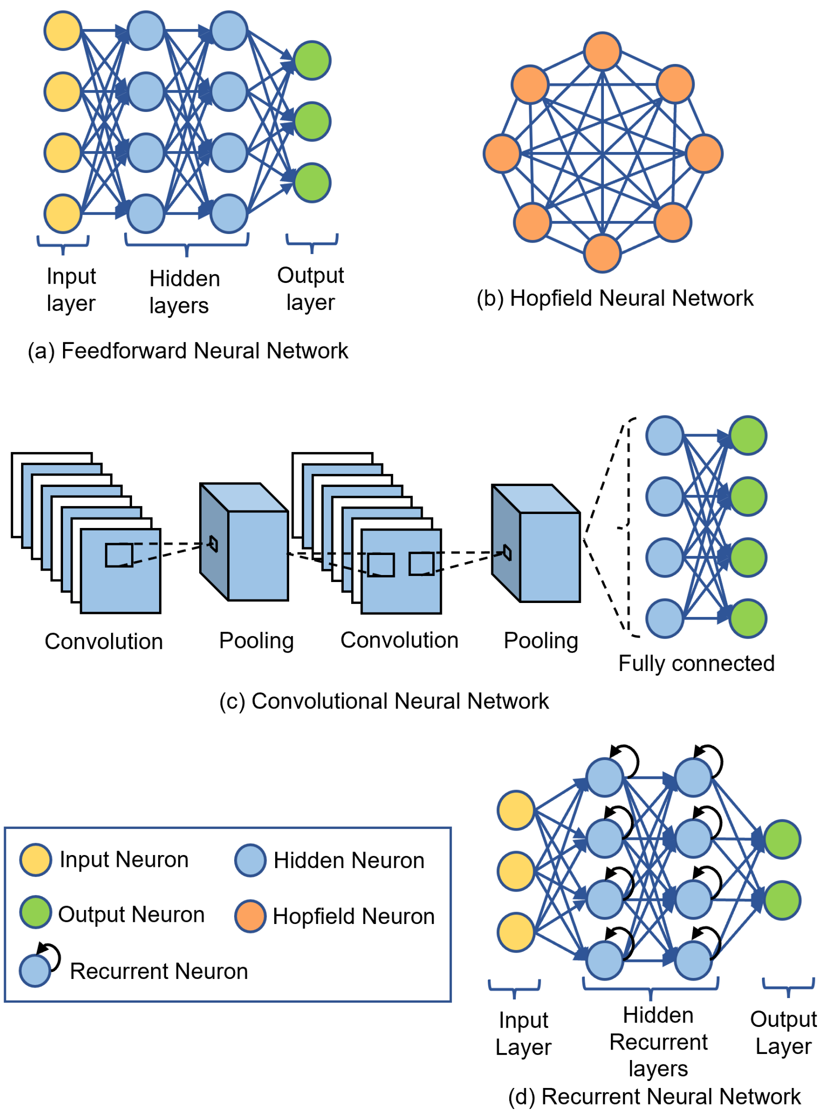


Figure 2.3: Some prevalent neural network topologies.

nized into three categories of layers: The input, hidden, and output layer. The connections between neurons in this network are made across layers and not within a layer. Information flows in a forward direction from the input layer, through the hidden layer(s), and finally to the output layer. The back propagation learning method is usually employed in training this network. The multilayer perceptron is another network with similar topology as the feed foward neural network. An example of a feed forward neural network usually employed in pattern recognition and classification tasks, is the radial basis function [28].

- **Hopfield neural network (HFs):** This network topology [29] shown in Figure 2.3(b), possess cyclic and recursive characteristics. They are made up of binary threshold neurons with recurrent connections between them, and can behave in several ways which include: settling in a stable state, oscilllating, or following less predictable disorganized trajectories. The global energy of Hopfield NN is determined by summing up several contributions. Each contribution can also be determined from one symmetric connections between neurons, and the binary states of the two neurons.

- **Convolutional neural networks (CNNs):** The convolutional neural network [30] is a popular neural network topology mainly employed in analyzing and classifying image data. As shown in Figure 2.3(c), it has an architecture that consists of a stack of distinct layers which are; convolution, pooling, and fully connected. The convolution layer is a mathematical function of convolution which is a unique kind of linear operation that multiplies two functions to produce a thrid. Using several filters, it employs a process called feature extraction to identify and separate different features of an image for analysis. The output of the convolution layer is called feature map, and it holds information such as corners and edges of the extracted image. These features are then propagated to the pooling layer which often follows the convolution layer.

  The pooling layer is tasked with reducing the size of output feature map from the convolution layer to reduce computational costs. It does this by emplolying one of several types of pooling operations to reduce the connections between layers, and individually operate on each feature map. Types of pooling operations which can be employed for this task include Max Pooling; which takes the largest element in the feature map, Average Pooling; which takes the average of elements within a section of the feature, and Sum Pooling; which takes a sum of the elements within a section of the feature. The Pooling Layer also serves as a link between a convolutional layer and a fully connected Layer.

The fully connected layer is usually the last layer of a convolutional neural network, and consists layers of neurons that are completely connected, along with their weights and biases. Features from the pooling layer are usually flattened before they are fed to the fully connected layer for classification.

- **Recurrent neural networks (RNNs):** The recurrent neural network is derived from the feed forward neural network. However, as described in Figure 2.3(d), its hidden layers are replaced with recurrent layers. The layers of an RNN don't only receive inputs from previous layers, but also the output of its own layer [31]. The ability of RNNs to process sequences of inputs with their internal state, make them suitable for speech recognition and connected handwriting recognition.

2.2 Conventional Artificial Neural Network

This section describes conventional artificial neural network (ANN) which is the second generation of neural networks, the learning methods employed in training it, and some rof its hardware implementation approaches.

2.2.1 Synapse learning rules

Learning in neural networks is simply the process of finding the best set of synaptic weights for maximixing a neural network's accuracy. Implementing learning has been one of the major challnges in the design of neuromorphic systems. Learning in neuromorphic system is implemented either on-chip or off-chip. The choice of implementation approach is made on a number of factors which inlude the neural network model, and hardware resources. In this subsection we review sonme of the learning approach and rules employed in training conventional artificial neural network.

2.2.1.1 Supervised Learning

Supervised learning approach entails training a neural network based on input-output pairs, where the network learns by example. One of prevalent learning methods used in training conventional ANN is the backpropagation (BP) learning rule. This learning rule trains an ANN by modifying its synaptic weights based on the error rate obtained in its previous training stage. It can be employed in training neural network topologies such as feed-forward, recurrent, and convolution. The BP learning rule is usualy implemented off-chip [32] on a traditional host machine. There, it is used to trains the ANN weights, and after the training, the trained weights are mapped to the hardware platform. This learning rule has demonstrated high precision in its training, taking advantage of the software platform. However, it is not suitable for neuromorphic systems that require frequent re-training of their synaptic weights. There have been several on-chip implementations of BP on neuromorphic systems [33], and variants of it, have also been either optimized or simplified for on-chip implementation [34]. Other supervised learning algorithms include support vector machines, and linear regression.

2.2.1.2 Unsupervised Learning

In contrast to the supervised learning approach, the unsupervised learining approach has no inpu-output pair, and has no example to learn from. This learning approach is quite unpopular, and there have been some usupervised learning rules based on self organizing maps [35] that have been implemented on neuromorphic chips.

2.2.2 Fundamental Implementation

In conventional ANN input signals and weights are represented with real values, and the implementation methods can be categorised either as analog or digital. Analog implementations offer rapid processing, power efficiency, and low area cost. However, they are susceptible to noise which makes representation of real numbers difficult, and accuracy limited. Digital implementation on the other hand, provides programmability, high precision, and reliability. Compared to analog implementation, they suffer from high area cost and latency. Hardware implementation platforms for ANN include CMOS technologies, and FPGAs.

2.3 Spiking Neural Network

Conventional ANNs have shown impressive results in various applications ranging from visual to signal processing [36]. However, when compared to biological neurons, they are highly abstracted and lack the ablility to assert the complex temporal dynamics observed in biological neurons. As a result, Spiking Neural Networks (SNNs); the third generation of ANN focuses on more biological plausibility by modeling biological neurons more closely. SNNs are able to amply express the dynamics of biological neurons, and also, represent and integrate several information in time, frequency, and phase. Due to their ability to closely model the complex information processing observed in the brain, SNNs offer a promising computing paradigm which is potentially adept at handling considerable amount of data, and representing them with spike trains. Furthermore, they are well suited for low power hardware implementation (neuromorphic). In this section, we first present the fundamentals of spiking neural network regarding spiking neuron models, encoding methods, and learning rules. We then briefly introduce interneuron communication architectures and platforms for implementing neuromorphic systems.

2.3.1 Spiking Neuron Models

As described in previous sections, SNN unlike its predecessors, is more analogous to the behaviour of the brain, and over the years, several spiking neuron models have been proposed. These spiking neuron models [37] differ in the level of details they abstract from biological neurons. In this subsection, we describe some of the prevalent models.

2.3.1.1 Leaky Integrate and fire

The Leaky Integrate and Fire (LIF) neuron model which is described in Figure 2.4 is one of the prevalent spiking neuron models. Its operation can be described as an accumulation process, together with leakage and firing mechanism. V_j^l of a LIF neuron j in layer l at a time-step t is described as:

$$V(t) = V(t-1) + \sum_i w_{ij} {}^* x_i^{l-1}(t-1) - \lambda \tag{2.1}$$

where w_{ij} is the synaptic weight from neuron i to j, λ is the leak and x_i^{l-1} is pre-synaptic spike from previous layer $l-1$.

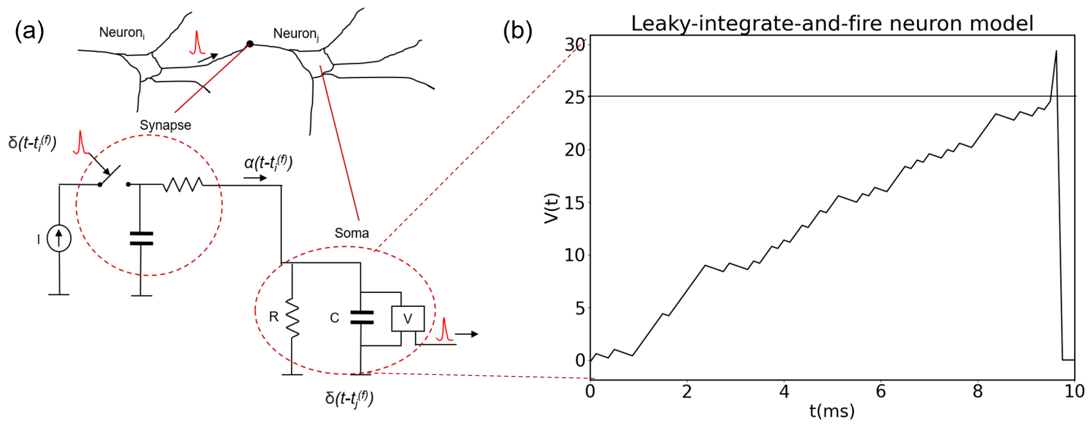


Figure 2.4: Leaky-integrate-and-fire (LIF) neuron model: (a) Circuit diagram of the LIF neuron model [5]. (b) Dynamics of the LIF neuron membrane potential V, simulated for 10ms.

As shown in Figure 2.4(b), when the value of accunulated weighted spikes is completed, the value of $V(t)$ is compared with the threshold value θ. If it exceeds, a spike is fired and the neuron resets. This is mathematically expressed as:

$$\begin{cases} 1, \text{if } V_j^l > \theta \\ 0, \text{otherwise} \end{cases} \tag{2.2}$$

2.3.1.2 Hodgkin-Huxley

The Hodgkin-Huxley neuron model described in Figure 2.5(a) was proposed In the early 1950s [17], is a mathematical representation of neuron dynamics. It describes the electric current through the membrane potential V of a neuron, providing the details of spike generation, as given in (2.3)

$$\frac{dv}{dt} = (\frac{1}{C})I - g_k n^4(v - E_k) - g_{Na} m^3 h(v - E_{Na}) - g_L(v - E_L) \tag{2.3}$$

where C is the capacitance of the circuit, I is the external current, conductances are potassium g_k, sodium g_{Na}, and leakage g_L. Gating parameters n, m, and h are determined by (2.4), (2.5), and (2.6), respectively

$$\frac{dn}{dt} = (n_\infty(v) - n)/\tau_n(v) \tag{2.4}$$

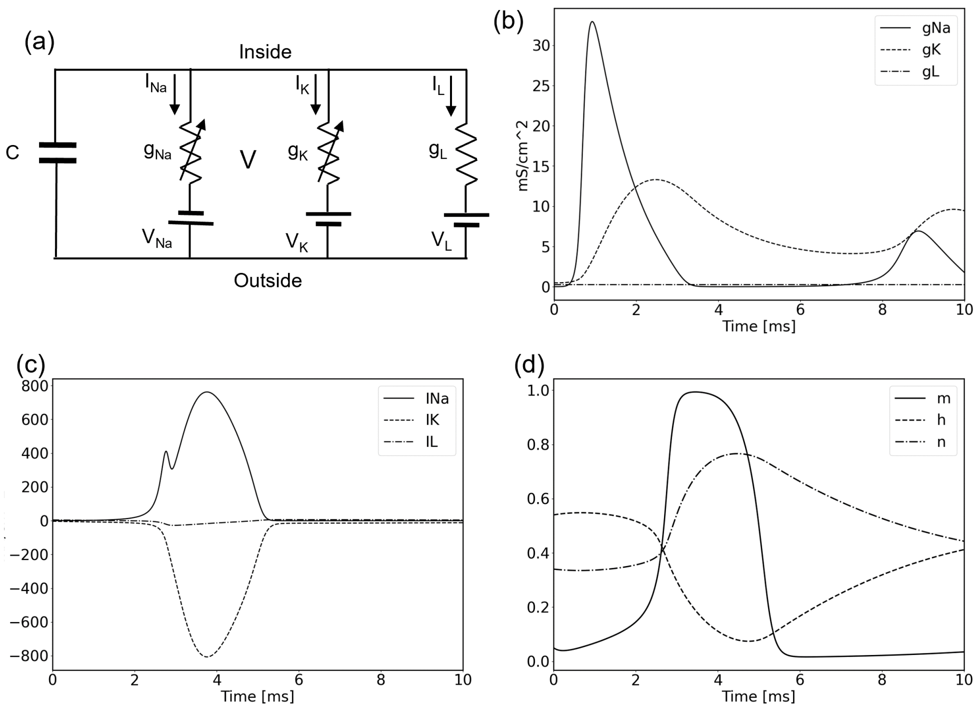


Figure 2.5: A simulation of the hodgkin and huxley model with input current of 10 and simulation time of 10ms: (a) Dynamics of the neuron voltage. (b) Dynamics of the gating variables (m, h, n). (c) The conductances of Na, K, and L. (d) The current of Na, K and L [6].

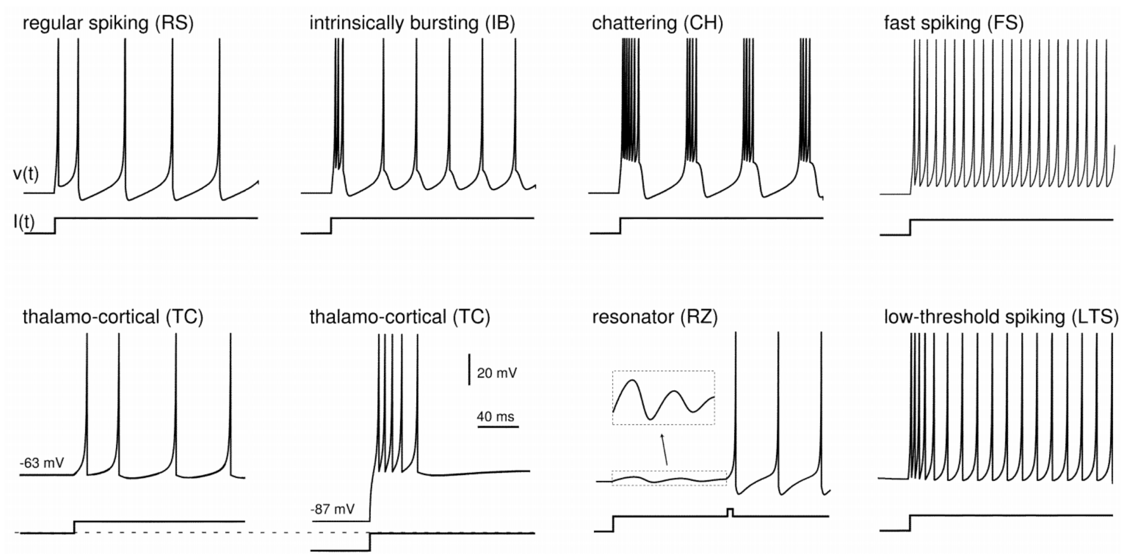


Figure 2.6: Spiking and bursting behaviors of known types of cortical neurons reproduced by the Izhikevicz neuron model [7].

$$\frac{dm}{dt} = (m_\infty(v) - m)/\tau_m(v) \tag{2.5}$$

$$\frac{dh}{dt} = (h_\infty(v) - h)/\tau_h(v) \tag{2.6}$$

The Hodgkin-Huxley model is the most biological plausible spiking neuron model. However, it requires many parameters, which increases its complexity, making it consume huge amount of hardware resources. It, therefore, is extremely expensive for large scale implementations.

2.3.1.3 Izhikevich

Compared to Hodgkin-Huxley, a less complex model was proposed by Izhikevich [7]. The model is described by the following equations:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{2.7}$$

$$\frac{du}{dt} = a(bv - u) \tag{2.8}$$

$$\begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad if\, v \geq 30mV \tag{2.9}$$

where v is the membrane potential of the neuron, u is a membrane recovery variable, I is the neuron current, a, b, c, d are parameters of the models, in which the various values of these parameters result in different types of neuron spike patternd described in Figure 2.6. When membrane potential v exceeds the threshold (30mV), the membrane potential v and recovery variable v are reset as 2.9.

In summary, among existing spiking models, Hodgkin-Huxley, Izhikevich, and Leaky Integrate-and-Fire (LIF) are often used. The Hodgkin-Huxley type is the best when measurable physiological parameters are highly considered. However, it is composed of many coefficients. This leads to challenges when implementing large SNNs because of high cost. In contrast, we can simulate hundreds of thousands of neurons when using LIF neural model; but, it is incapable of producing rich spiking patterns. Finally, the Izhikevich exhibits a good compromise in terms of biophysical similarity and computational cost. It is close to the Hodgkin-Huxley model in biological plausibility while analogous to the LIF in computational complexity.

2.3.2 Neural Coding Schemes

It is well known that information transfered among neurons in the brain are encoded in spike patterns, and a neural code describes how this informatiion is represented with electrical activity in a pattern, both in the neuron level, and in networks of neurons [16]. Although, extensive studies have been carried out on neural coding schemes of spike patterns with the aim of understanding the cognitive system and the principle of information transmission and processing in the brain, information encoding still remains a wrangle in the computational neuroscience community. It was previously postulated that the brain encodes information through spike rates due to its experimental discovery in sensory systems such as visual and motor cortex. However, rate coding scheme is defined by long processing time and slow information transmission, and neuroscience research have uncovered rapid processing in the brain that cannot be achieved by rate coding scheme alone. The human visual processing have been discovered to carry out a recognition task in less than 100ms using neurons in various layers from the retina to the temporal lobe [38]. Each neuron takes about 10ms to process spikes, making the time-window too small to achieve rate coding.

Tasks which require rapid processing can be accomplished using precise timing of spikes. One

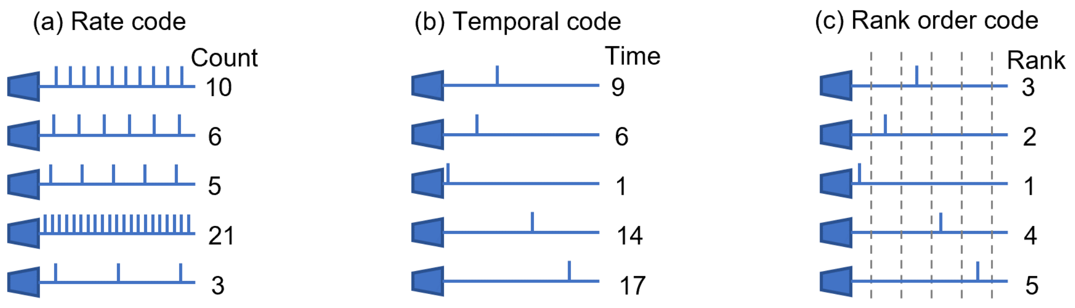


Figure 2.7: SNN Neural Coding Methods.

issue that has been associated with the rate coding scheme is that the firing of so many spikes for a stimulus requires ample amount of energy and resources. However, it is posited that the precise timing of spikes encodes more information in a small network of spiking neurons. Therefore, it can be said that information is likely to be encoded not just in the number of spikes or firing rate, but also, in the precise timing of individual spikes. In this subsection, we describe these two coding schemes and how they are employed in encoding information in SNN.

2.3.2.1 Rate Coding

The rate coding scheme which is sometimes called frequency coding, conveys stimulus information based on the firing rate of a neuron, which is proportional to the level of stimulus. As described in Figure 2.7(a), it can be performed for a single neuron in terms of the average of spike count in an interval of duration over a single trial, and density rate of spike count over several trials. A population activity rate can also be performed for a population of neurons which have similar characteristics, and interract together [39].

2.3.2.2 Temporal Coding

From studies, it has been shown that the temporal resolution of neural code is on a scale of milliseconds. Therefore, Figure 2.7(b), describes how information is represented in the precise timing of spikes. Variants of temporal coding include:

- **Time to First Spike:** In this coding scheme, only the first spike after stimulus is considered, and then the timing of this spike is used to encode information. A strong stimulation in a neuron could cause it to fire, shortly after a stimulation, while the following spike becomes weaker [40].

- **Phase:** In phase coding, temporal information is encoded into spikes using a reference global oscillator, and the relative time difference between spikes. The global oscillators function as an internal reference to signals. An evidence of this coding scheme was discovered in the hippocampus of the rat [41].

2.3.2.3 Rank Order Coding

Rank order coding [42] employs a somewhat similar priciple as temporal coding by taking the temporal information of spikes into consideration. However, it adopts a simpler approach which uses the relative timing of spikes across a population of neurons, rather than their precise timing. As shown in Figure 2.7(c), rank order coding depends only on the order in which spikes arrive, making it considerably easier to compute. With factorial N possible orderings, a rank order code can relay up to log2(N!) bits of information where each presynaptic neuron can fire only a single spike.

2.3.3 Spiking Neural Network Learning Rules

The computational substrate employed by the biological brain to carry out its cognitive fuctions is facilitated by the plasticity of synapses described in Figure 2.8(a), which connect biological neurons to form networks. Several studies have taken diverse approaches in describing the functions attributed to synaptic plasticity, and this has led to many forms and mechanisms of SNN learning algorithms. The enhancement or depression of synaptic transmission in the brain is significantly influenced by activities. These activities span temporal domains in the range of milliseconds, hours, days, and presumably even longer as basically all excitatory synapses in the biological brain assert various forms of synaptic plasticity [16]. As an inspiration for an energy efficient system, prominent forms of plasticity observed at excitatory synapses in the brain are modeled in spiking neural network using various learning algorithms which are generally classified as unsupervised, or supervised. In this subsection, we prsent a broad overview of the mechanisms, possible functions of the long-term potentiation (LTP) and long-term depression (LTD) phenomenas observed in synaptic plasticity, and describe some of the prevalent unsupervised SNN learning algorithms that model them. We also present some supervised learning algorithm which are employed in SNN learning.

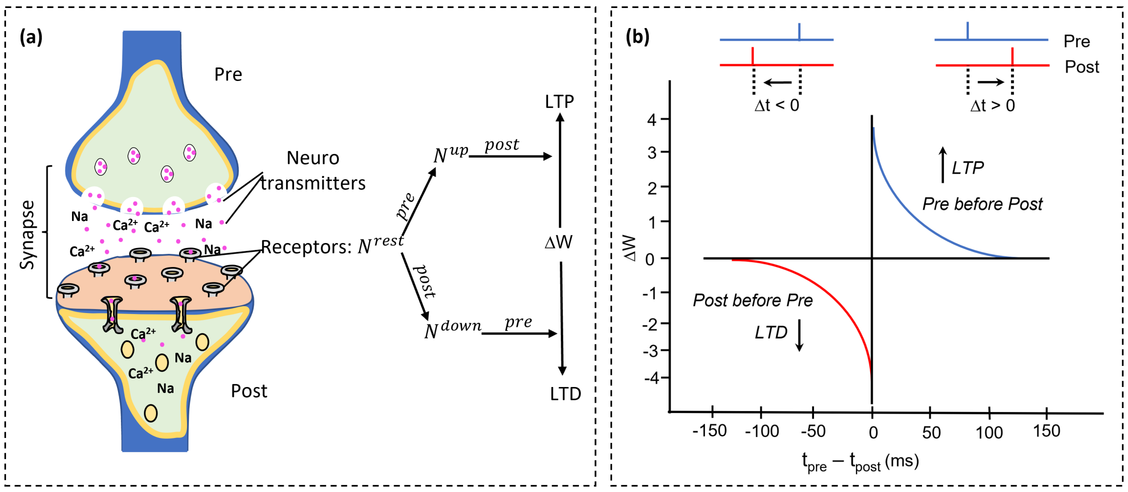


Figure 2.8: Synaptic Plasticity: (a)The physiological mechanism of synaptic plasticity in the brain. (b) Spike timing dependent platicity (STDP) curve.

2.3.3.1 Unsupervised Learning

One of the most prevalent SNN learning rules implemented in neuromorphic systems goes by the term "Spike timing dependent plasticity" (STDP) [43] which refers to the observation that the precise timing of spikes significantly enhances or depresses synaptic transmission. As described in Figure 2.8(a), a presynaptic spike *pre* preceding a postsynaptic spike *post* within a small time window enables receptors to open N^{up}, and if this persists for an extended period of time, leads to long-term potentiation (LTP). If the order is reversed, and the post synaptic spike precedes the presynaptic spike, the receptors become closed N_{down}, and results to long-term depression (LTD) if it persists for an extended period of time. Inducing such presynaptic and postsynaptic spike pairs repeatedly with a fixed time interval gives the timing-dependence of plasticity. As illustrated in Figure 2.8(b), such an STDP curve is thought to facilitate the prediction plasticity which results from varying the time intervals. Consequently, the STDP curve is considered as a learning rule which delineates how a specific type of synapse engages in information storage and eventually cognition. The STDP learning rule is further defined by the following equation [44].

$$W(\delta t) = \begin{cases} A_+ e^{-\delta t/\tau_+}, \text{ if } \delta t \geq 0 \\ A_- e^{-\delta t/\tau_-}, \text{ if } \delta t < 0 \end{cases} \tag{2.10}$$

where A+ and A− are the rate of synaptic potentiation and depression respectively, and τ_+ and τ_- are time constants that characterize the span of positive and negative learning window.

2.3.3.2 Supervised Learning

Apart from the unsupervised learning, supervised learning for SNNs is also a significant research area. Over the years, several SNN supervised learning rules have been proposed, and in this subsection, we present some of them.

The authors in [45] successfully implemented the spike-driven synaptic plasticity (SDSP) learning rule which update synaptic weights in the event of a presynaptic apike. Modified versions of STDP has been proposed as a supervised learning algorithm by different researchers. An inflected STDP (weight-dependent STDP) was also proposed in [46] as a method of estimating gradients for online supervised learning in a multi-layer SNN.

Several Back propagation based SNN learning algorithms have been designed and proposed by several researchers for training Neuromorphic SNNs. One of the popular ones SpikeProp [47], acts on SNNs that use explicit spike time temporal coding. Each neuron fires only one spike, and the timing of the spike is modified using error back propagation. Spike Prop is limited, in that it learns the timing of output spikes but cannot adapt the number of spikes.

One of the known SNN supervised learning algorithm Tempotron [48], is an online gradient based learning rule that implements a binary classification of multi-neuronal spike patterns. It classifies spike patterns either as a target pattern (P+) if the neuron fires a minimum of one spike in response to it, or a null pattern(p-) if the neuron does not fire at all. Several modifications on Tempotron have been proposed by researchers to handle different tasks.

In [49], the authors proposed a supervised learning algorithm that implements error back propagation and uses memristor as synapse device. This algorithm unlike others, does not modify weights but stores and recalls patterns by changing the delay of every connection, making for uncomplicated neuromorphic implementation without multipliers or digital-analog converters (DAC).

2.4 Fundamental Implementation

Research interests in the development of hardware based neural network have increased over the years as a result of the breakthrough achieved by deep learning. In this regard, several architectures,

Table 2.1: Platform comparison for neuromorphic implementation [11].

	Parallel computer	FPNN in FPGA	DSP	FPGA	Analog ASIC	Digital ASIC
Speed	+	+	-	+	+++	++
Area	- -	+	-	+	+++	++
Cost	- -	++	++	++	- -	- -
Design time	+	+++	++	++	- -	- -
Reliability	++	++	++	++	- -	+
- - very unfavorable, - unfavorable + favorable, ++ very favorable, +++ highly favorable						

and several technologies relating to implementation type and platform, synapses, and inter-neuron communication are being explored for both conventional ANN and SNN acceleration [50]. The computational model of Conventional ANNs is quite different from that of SNN. ANN operations are characterized by matrix-vector multiplications which are often easily performed on central processing units (CPUs), graphics processing units (GPUs), and recently, tensor processing unit (TPU). Conversely, SNN as described in prior sections are event driven, and alongside the conventional ANN accelerators, research on the acceleration of SNNs in hardware have significantly increased in recent years. SNN is often simulated in discretized timesteps, and in those timesteps, the state of every neuron is evaluated. Shorter simulation timesteps, usually gives better simulation results, however; this leads to prolonged simulation time. Consequently, a specialized hardware that leverages the dynamics of SNN for rapid low power simulation on several hardware platforms will not only enable biologically plausible neurons and synapses to be simulated, but will also provide better understanding of the brain's operation [50].

2.4.1 Hardware implementation methods and platforms

update There have been increased interest in the potential benefits of neural network optimization techniques in hardware. These benefits which include high performance and energy efficiency, are the driving force for this increased interest. To this aim, several hardware platforms are being exploited, and among them, field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs) are mostly employed. In this subsection, we present some of these hardware platforms and describe their advantages and limitations towards neural network acceleration.

A summary of the comparison among parallel computers, FPGA, DSP, and both analog and digital ASIC is presented in Table 2.1.

Field Programmable Gate Array (FPGA:) An FPGA is a specialized hardware that is particularly designed to be reconfigurable based on the requirement of the user. It consists of several programmable logic blocks and configurable interconnections which enable the interconnection of its blocks to realize different configurations. Its configurability, short development time, low cost, and recent increased support for multiply and accumulate operations makes it a deriable platform for implementing ANNs. Because of these advantages FPGAs are increasingly being employed in several applications such as real-time hand gesture detection and tracking [51] and face tracking and identity verification in video sequances [52]. However, its resources are constrained, therefore it cannot used to simulate large neural networks.

ASIC: With the increasing need to perform deep learning in a rapid and energy efficient manner, many custom ASIC and system on chip (SoC) chips are being developed by several laboratories and companies around the world. because ASIC is application specific, it is much more power efficient, and can run at higher clock frequency than FPGAs. The implementation of ASIC chips are generally divided into Analog and Digital. Despite the similarity of analog ASIC to the brain, digital Full custom ASIC is the prevalent platform for spiking neuromorphic implementation. This is due to the ease of information storage, processing and decision making in digital systems. Two well-known examples of digital neuromorphic systems are TrueNorth [53] and SpiNNaker [54]. Compared to digital approaches, analog platforms are less common due to challenges in implementations. They are attractive for their energy-efficiency and throughput in AI applications. Example of analog ASIC is intel's electronically trainable analog neural network (ETANN) 80170NX chip [55] which consists of 64 neuron that are fully connected, together with their synapses. A mixed signal chip was proposed in [56] and utilized capacitors for weight storage. neural signals are conveyed as currents. This makes the system relatively robust and scalable thanks to neural signals being maintained over long distances.

2.4.2 Learning

Several opproaches are considered when implementing learning on neural network acceletors, and this is influencd by the target application. The two implementation approaches considered in this regard are on-chip and off-chip. If the target is to design a general accelerator for machine learning where the application requires realtime adaptation, especially in a dynamic environment, then on-chip learning is required chip. however, if the target application does not require real-time adaptation, off-chip learning is considered appropriate.

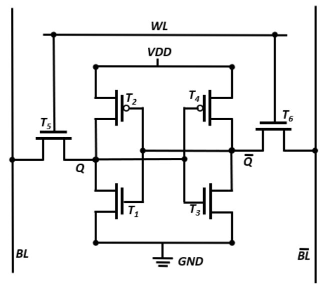
The off-chip learning approach is realized by performing a training of the neural network on a some high end server, and afterwards, the resulting network along with its configuration parameters and trained synaptic weights are programmed on the chip. With this learning approach the complexity and resource demand of learning is alleviated from the chip. The draw back however, is that the resulting neural network cannot learn a new task without loosing accuracy. Its synaptic weights and configuration parameters will have to be updated from time to time, for improvement, or to enable it perform other tasks depending on the requirement of the target application. authors in [57], performed some training with backpropagation where an ANN was first trained, and then converted to SNN by mapping real-value inputs/activations to average firing rates of Poisson spikes. This training approach can be used on neuromorphic systems as off-chip learning.

On-chip learning on the other hand requires that the training of synaptic weights to be performed on-chip, which necessitates the inclusion of learning circuits and memory on chip. For SNN, Unsupervised learning algorithms like STDP and its equvalents are often considered suitable for on-chip learning since no information is transmitted between neurocores in performing synapse update. The drawback however is that the learning circuit and memory further increases the complexity of the chip in terms of area and power. Nevertheless, there are ongoing research on energy efficient on-chip learning. Conversely. existing supervised learning algorithms on the other hand are not considered suitable for on-chip learning because complex neuron and synaptic models with fixed point value gradient communication among neurocores, which signifcantly increases complexity. Furthermore, algorithms like backpropagation is likely to have high operation latency, since forward propagation will have to be put on hold for learning to occur.

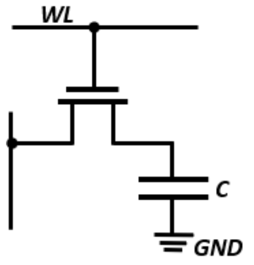
2.4.2.1 Synapse Memory Technologies

In traditional computer systems, memory speed is a bottleneck because processors have significantly improved over the years surpassing memory speed and throughput, and this leaves processors idle while waiting for memory. But SNN is different. It's architecture provides memory in company with processing, and they both operate in parallel. These spiking neurons process with events, and as opposed to traditional processors that operate and communicate in megahertz and gigahertz ranges, they communicate in around 10 hertz. This difference in speed enables neuromorphic processors to use time multiplexing to combine many events into a single communication channel. In SNN, Storing and reading of synaptic wights constitute the major operation, and designing a large SNN with enormous number of synapses will require large memory bandwidth. So while communication speed is not somewhat a challenge in neuromorphic systems, memory bandwidth definitely is one that need to be overcome. In addressing this challenge, researchers have taken several approach to realizes these synapses in CMOS by exploring various memory technologies. Some of these memory technologies include; static random-access memory (SRAM), a prominent memory technology in semiconductor design. in [58] a neuromorphic chip with 256 neurons that implements a transposable 8-transistor SRAM based crossbar which grants row and column access was employed.

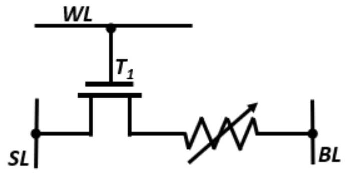
A typical SRAM contains six transistors (6-T), and although having high leakage current and low density, SRAM offers multiple read and write. Its major advantages are speed and reliability when compared to other memory technologies [59]. The eight transistor (8-T) SRAM has two transistors more than the typical SRAM. These two transistors adds access to the word and bit lines in transposed orientation to the typical 6-T design. To handle the general leakage power of the chip, ultra-high-Vt devices which reduced it by 3, at a cost of increased minimum operating volatage of the memory array is leveraged. Another memory technology is embedded dynamic random-access memory (eDRAM) which was used in [60] to design a high-density 3D memory for a programmable digital neuromorphic architecture.



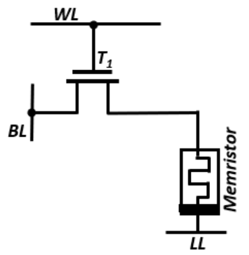
(a) 6-T SRAM cell



(b) eDRAM cell



(c) STT-RAM cell



(d) RRAM cell

Figure 2.9.

Table 2.2: Summary of Memory Technologies.

Feature	SRAM	eDRAM	STT-RAM	RRAM	PCM	DWM	Flash(NAND)	HDD
Cell size *(F^2)*	120-200	60-100	6-50	4-10	4-12	≥ 2	1-4	2/3
Speed (R/W)	Very fast	Fast	Fast/Slow	Fast/Slow	slow/Very slow	Fast/slow	Very slow	extremely slow
Write Endurance	10^{16}	10^{16}	4×10^{12}	10^{11}	10^{8}-10^{9}	10^{16}	10^{4}	10^{4}
Leakage power	High	Medium	Low	Low	Low	Low	Very Low	extremely low
Dynamic energy (R/W)	Low	Medium	Low/high	Low/high	Medium/high	Low/high	low	Very high
Retention period	Voltage applied	30-100 μ	years	years	years	years	years	years

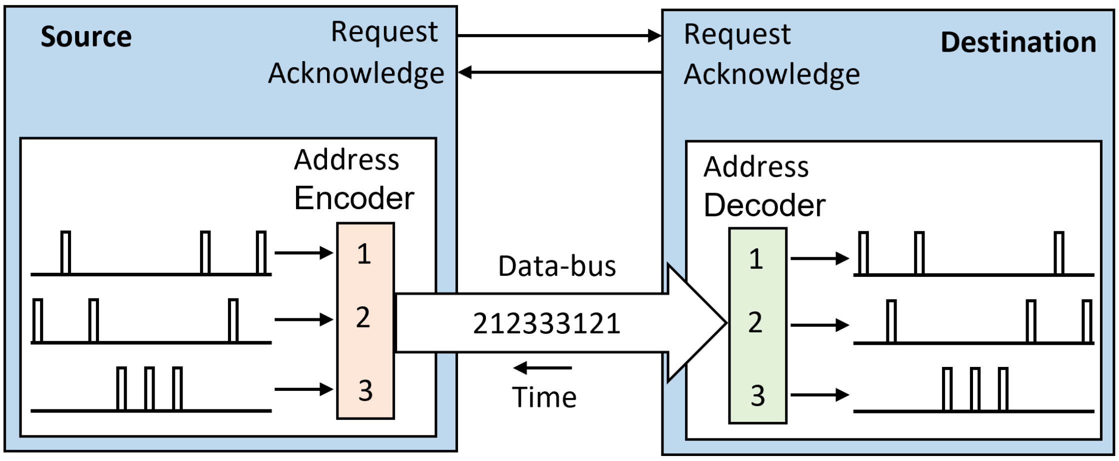


Figure 2.10: Illustration of address event representation (AER) inter neuron communication communication.

eDRAM is a capacitor-based memory integrated on the same multi-chip module. Due to its simple conventional one 1-transistor 1-capacitor design, it is inclined to having less area cost when compared to SRAM. However, with the gradual leakage of its storage charge during operation, its retention period is low. also, its design is not easily compatible with CMOS. in [61], a spin transfer torque ram (STT-RAM) was propsed as a stochastic memristive synapse for neuromorphic systems. An STT-RAM is a magnetic RAM which uses magnetic tunnelling junction (MTJ) in its cells, and its simple design has small area when compared to SRAM and eDRAM. By adapting a schema proposed in [62], the authors organized the STT-RAM MJT as a crossbar connecting input and output neurons. However, the magnetization of the STT-RAM makes writing process slow, and consumes more energy. Another memory technology that has been used in the design of neuromorphic chips, is Resistive random-access memory (RRAM or ReRAM). It is a memory technology that rely on the resistance change of its cells to store information, and it has similar architecture to eDRAM. in [63] the authors proposed an FPGA-based hardware emulator for neuromorphic chip with RRAM-based crossbar. RRAM offers low area, low power, and easy integration on CMOS. However, it suffers fron stuck at faults (SAF) [64] which may cause short circuiting and lead to increased power [63], and dynamic switching variation which leads to a large variation in the resistance of the memory [65]. Other memory technologies include the Phase Chanhe Memory (PCM), A summary of some memory technology is presented in Figure 2.2

2.4.3 Inter Neuron Communication

To connect PEs together, these systems use various communication architectures such as common bus [66], ring [67], and network-on-chip [68]. Efficient inter-neuron Communication is crucial for the operation of a neuromorphic system, and several existing networking solutions are edapted to achieve this. In adapting existing networking solutions, it is important to address the challenges that stem from the disparity between the requirements of computer networks and those of neuromorphic systems. While exiting communication architectures are known to connect thousands of nodes at different levels, neuromorphic systems need to connect millions of neurons at the chip- or core level. Hence, the efficiency communication architectures, employed for inter neuron communication has to be in several orders of magnitude.

A known communication protocol popularly employed for inter neuron communication in neuromorphic systems is the address-event representation (AER), which is used to transmit spikes from an array of presynaptic neurons on one core/chip to an array of postsynaptic neurons on another core/chip [69]. As illustrated in Figure 2.10, an address-encoder in a source core/chip generates a unique binary address for each presynaptic neuron whenever it spikes, and a data bus is used to transmit these address to the destination core/chip, where an address decoder updates the corresponding post synaptic neuron. To establish communication between connected cores/chps, a four phase hand shake is performed using the pair of connections shown in Figure 2.10. The source core/chip initiates the handshake by asserting a request signal, and the destination core acknowledges. The sent address event is assumed to be valid when the request signal arrives the destination, which requires their propagation delays to be matched.

AER is suitable for SNN implementations since it only needs to be active whenever neurons fire. To scale up the system, a hierarchical AER as a tree structure was implemented in [70].

Figure 2.11 illustrates the variety of neuron communication interconnect emoployed in neuromorphic system design. Figure 2.11(a), describes the shared bus interconnect where processing elements (neurocores) use a shared bus for communication. Given SNNs can have extremely high traffic, the shared bus approach is not suitable for neuromorphic implementation because with number of nodes, communication in the dsystem suffers adversely. Figure 2.11(b) illustrates a different type of communication interconnect where processing elements are diredtly connected to

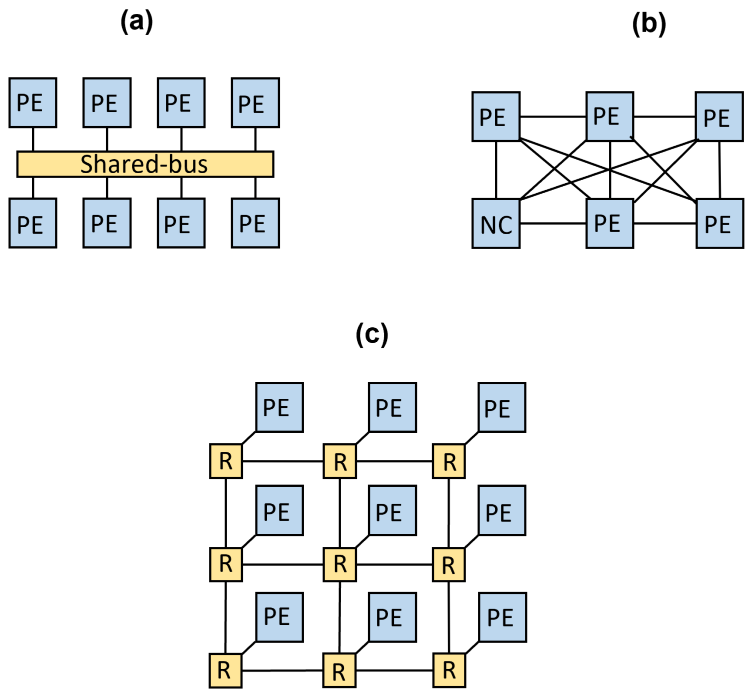


Figure 2.11: inter-neuron communcation architectures. NC- neural core, R- Router. (a) Shared bus (b) Point-to-point (c) Network on chip.

each other from point to point. This communication are=chitecture is also not good for SNN implenention because a lot of lengthy wires are required to connect many processing elements which is not practical. Network on chip (NoC) wchich is shown in Figure 2.11(c) on the other hand, is commonly implemented for on-chip inter meuron communication because it more scalable.

2.5 Conclusion

In this chapter, we presented an overview of artificial neural networks including the spiking neural network as the latest generation and how they are implemented. Compared to conventional approaches, SNNs offer not only the capability of simulating biological neural networks but also extreme energy efficiency thanks to event-based operations and fewer operation computations. In the next chapter, we focus on how prior works tried to solve the interconnect challenge in spiking neuromorphic and faults in neural networks.

3
Related Works

Quite a number of neuromorphic systems have been proposed over the years. These neuromorphic systems leverage the extensive trasnsistor resource available, and scalable architectures to provide scalable neuromorphic architectures capable of supporting SNN with massive number of synapses. In this chapter, we review some of these neuromorphic systems, together with the various interconnect architectures they employ. We also review some existing works on fault tolerance, and their approach.

3.1 Neuromorphic System Architectures

A number of large-scale neuromorphic systems have emerged in recent years, taking advantage of the enormous transistor resource now available on a single microchip and, in some case, a full silicon wafer. Recent increase in technological capabilities combine, alongside scalable architectures, enable neuromorphic systems to support scales of neural network which incorporate millons of neurons with billions of synapses.

3.1.1 Digital Implemenation

3.1.1.1 Hierarchical Bus

In [71] an extension of the AER protocol, that enables auditory processing elements to send events using a shared asynchronous bus that requires no external logic for arbitration was proposed. Several of these implemented auditory processing elements utilize a shared asynchronous bus for communication.

3.1.1.2 2D Packet Switched

With the aim of modelling large scale SNN with massive number of synapses, the SpiNNaker project [54] proposes a full custom digital neuromorphic system. Housing a total of one million mobile integer cores, each with 32Kbytes of instruction memory and 64Kbytes of data memory. The project aims to simulate up to a billion neurons in real-time, and also provide support for different neural models. Each SpiNNaker chip is designed on a 130nm CMOS, and houses 18 ARM968 cores. 16 of those cores are used to implement spiking neuron models, one is used for monitoring, while the remainder as spare which can be used in the event of fault. To interconnect the cores, a 2D triangular mesh communication fabric is employed. The spinnaker system have been used for applications such as Vision processing, robotic control, and realtime implementation of 2.5 million neuron Spaun model [72].

The Loihi neuromorphic chip [73] designed by intel is another 2D packet switched neuromorphic architecture that comprises a total of 131 072 leaky-integrate-and-fire neurons distributed among 128 digital cores interconnected to form a spatial asynchronous 2D mesh. Each Loihi chip houses 1024 neurons, 128 kB of synaptic state, and 20 kB of routing tables which can be variably allocated accross its neurons. Spike based communication among neurons is represented with a 32-bit packet which contain destination neuron address, source neuron address, and graded-value payload. for inter-chip communication, four off-chip interfaces are implemented, and the offchip packets are further encapsulated with a 32-bit header, used to specify chip address [73].

TrueNorth [53] system is a known hardware-based SNN chip developed by IBM with 2D packet switched interconnect. With a full custom digital ASIC design, each chip in 5.4 million transistor 28nm, houses 4,096 neural cores. Each core contains 256 integrate-and-fire neurons, and each neuron has a fan in of 256. With a 256$\times$256 synapse crossbar, each core is able to selectively connect presynaptic spikes with postsynaptic neurons. For communication among cores, configurable point-to-point routes are setup such that output spike events from one core is directly routed to another core as input. This direct connection extend to cores on other chips as well. A total of 16 chips directly connected on a circuit board have been developed.

In [45], a quad-core binary weight neuromorphic chip was proposed. Each core houses 512 LIF neurons, a total of 528k binary synapses grouped in three levels, and a stochastic spike-driven synaptic plasticity learning. Communication among the cores are enabled via the 2D-NoC star-based topology communication architecture.

3.1.1.3 3D Packet Switched

The work in [74] investigated the architecture and design of a 3D stacked neuromorphic accelerator. The 3D stacking architecture used face-to-face bonding of two 20cm wafers with micro-bumps. A Recent work was presented in [75] about a neuromorphic system for the simulation of large-scale conductance-based SNNs. The architecture was implemented in six Altera Stratix III FPGA boards to simulate one million neurons [75]. Another recent work was presented in [76], where the authors proposed an SNN architecture based on 3D memristive synapses. An *AER* multicast routing mechanism was used for inter-neuron communications. Although the NoC architecture meets the requirements of the system, it is hardly deployed in embedded neuromorphic systems [77]. The authors in [78] proposed a neuromorphic architecture for large-scale biophysically meaningful neural networks with multi-compartment neurons in a 3D-NoC architecture, using four Altera stratix III FPGAs. However, this system utilizes a butterfly fat-tree-based topology, which although mimics the brain's connectivity more closely, cannot be mapped on an ASIC design as its shape will lead to wastage of silicon area. Moreover, the point-to-point connection at the parent node limits path diversity, and due to the bandwidth imbalance in this topology, thermal hot-spots can be created. In [79], a 3D-NoC based neuromorphic system was proposed, which houses neuro cores in 2D mesh topology layers that are stacked to form a 3D architecture. Communication with the stacked layers was enabled using through silicon vias (TSV).

3.1.2 Analog Implementation

3.1.2.1 Hierarchical Bus

The work in [80] proposes a multichip analog neural processing communication architecture that utilizes a nonarbitered, asynchronous shared-bus for communication of events. Although this approache support multicast and broadcast routing, it suffers from the limitation of scalability when the network size increases.

3.1.2.2 2D Packet Switched

The BrainScales [81] system which is supported by the EU Human Brain project uses above-threshold analogue circuits to implement models of neuronal dynamics. These above threshold circuits enable BrainscaleS to run at up to 10K times biological speed. BrainScaleS utilizes wafer-scale integration, and a wafer has 48 reticles, each of which house 8 hgh-count analogue neural netrwork (HiCANN) die. Each of the 8 HICANN dies, embeds 512 adaptive exponential integrate and fire (AdExp) neurons with about 100K synapses [72]. For communication among dies, Hi-speed serial channel are employed to convey presynaptic spikes from 64 neurons in a HiCANN die to post synaptic neurons in another HiCANN die.

In contrast to BrainScaleS, the Neurogrid [82] system uses subthreshold analogue circuits to model neuron dynamics, together with digital spike communication. The neurons on Neurogrid are inplemented using a shared leaky integrator dendritic structture. Each

Neurogrid chip is accompanied by a router with which it communicates spike packets with other chips. The routers utilize a 2D tree based multicast routing approach where spike packetes are routed to upstream routers up the tree till it reaches a router above the intended destinations. The maximum number of neurons per Neurogrid layer is limited (up to 2,175 neurons), and this makes it unable to offer biological real time behavior [82]

The ROLLS neuromorphic processor [83]. ROLLS has 128K analog synapse and 256 neuron circuits that support biologically plausible dynamics and bi-stable spike-based plasticity mechanisms that endow it with on-line learning abilities.

3.2 Fault-tolerant Neural Network

Over the years, several works that focus on fault-tolerance in neural network hardware hae been proposed [84]. Some of these works and the various approaches they employ in realizing fault-tolerance are in Figure 3.1

3.2.1 Learning-based approaches

The learning based fault tolerance approach modifies conventional learning rules, and uses them to handle faults that occur in neural networks systems. The authors in [85] proposed a fault-tolerant method which temporarily injects faults in hidden neurons during training process. In this method, about three neurons are randomly injected for each input example. A modified training rule was also presented in [86]. This rule functions by adding a regularization term to the cost function. In [87], a back propagation based method was proposed for dealing with faults that arise in classification applications. This method constrains weights to a certain range. The modified learning methods have demonstrated good performance, and do not need external interactions after the training. However, their computation cost is significantly high, and they require a long time to complete their training process.

Another method that is widely employed is the retraining methods. this method was employed in [88], where periodic retraining in GPGPU is performed to improve fault tolerance. The Work in [89] proposed a retraining scheme that handles the effect of timing errors in neuromorphic systems.

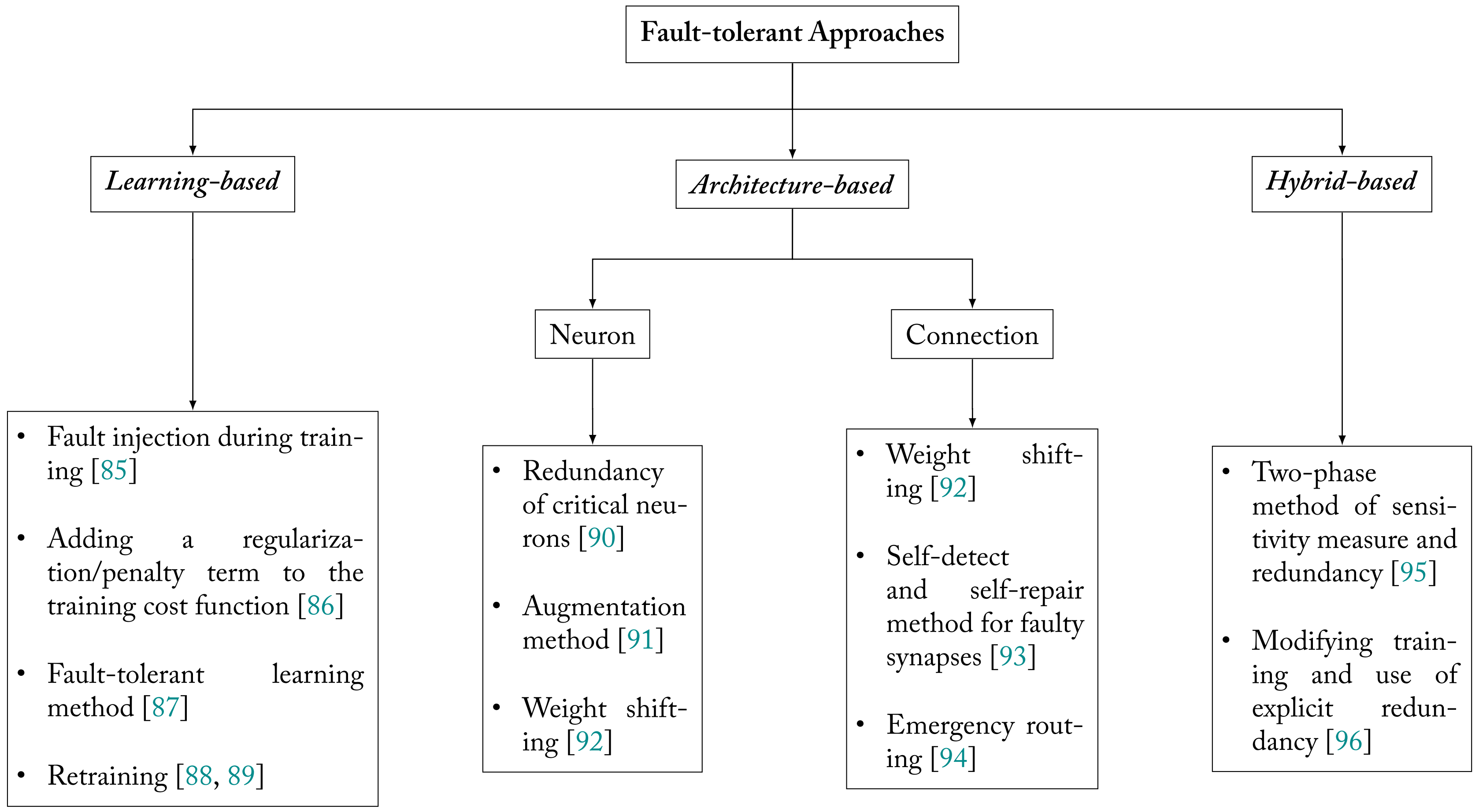

Fault-tolerant Approaches
Learning-based
Architecture-based
Hybrid-based
Neuron
Connection
• Fault injection during training [85]
• Adding a regularization/penalty term to the training cost function [86]
• Fault-tolerant learning method [87]
• Retraining [88, 89]
• Redundancy of critical neurons [90]
• Augmentation method [91]
• Weight shifting [92]
• Weight shifting [92]
• Self-detect and self-repair method for faulty synapses [93]
• Emergency routing [94]
• Two-phase method of sensitivity measure and redundancy [95]
• Modifying training and use of explicit redundancy [96]

Figure 3.1: Summary of fault-tolerant approaches in neural network architectures.

This retraining scheme is performed when the output results of a neural network have been affected by timing errors. The authors in [97] propsed a learning scheme which mimics the self-repair ability observed in the brain, by reestablishing the firing rate of neurons in the event of synaptic faults.

3.2.2 Architecture-based approaches

The architecture based approach to fault tolerance primarily employs redundancy in the hidden neurons and synapses of pretrained networks. In [90] the authors proposed a fault-tolerant architecture which employs redundant neurons. The authors in [91] utilized redundant critical hidden neurons together with a technique called augumentation to half the weights of synapses between augumented neurons and output neurons.

Apart from faults that occur in neurons, the connection between neurons can also be faulty, therefore they have also been of concern. In handling such faults, the authors in [92] propoosed a method called weight shipping where the weights of faulty connections are shifted to other connections of the neurons that are not faulty. The authors in [93] also proposed fault-tolerant self-repairing architecture which uses a self-detect and self-repair mechanism to detect and repair up to 40% synaptic faults, while maintaining good system performance. However, the evaluation of this fault-tolerant method was done with only two neurons, and the performance may deterriorate due to area overhead when the architecture is scaled. SpiNNaker [94] also employs, an emergency routing mechanism that uses redundant adjacent connections to automatically redirect blocked packets to their destination when there is a congested or faulty connection in its 2D-NoC torus topology. This mechanism enables the SpiNNaker system to mitigate timing violations of SNNs in the event of congestion or faulty connections.

3.2.3 Hybrid approaches

The hybrid approach to fault-tolerance combines both the learning-based and architecture-based approaches. The authors in [95] proposed a two-phase method which first removes unresponsive hidden neurons by measuring their response when sending inputs to them.

Afterwards, redundant neurons are added, and the network is retrained. The evaluation results of this approach show an improvement in the system's fault-tolerance when performing two multiclass classification problems. This work was further extended in [96], where the authors first used weight constrained backpropagation training to avoid fault-tolerant degradation that results from unconstrained weights. During the training, some faults are injected into some neurons and connections. The network is then pruned to remove irrelevant neurons. After prunning, redundant neurons are added to the network to share the role of critical neurons in the network. The evalution results of this method showed better robustness when compared to other methods.

3.3 Conclusion

To summarize, we reviewed in this chapter some existing neuromorphic systems together with their neuron communication architecture. We also presented some existing works on faults tolerance in neural networks and the various approaches they employed. From the review, we can see that 3D-NoCs has the potential to provide high parallelism, scalability, and small footprint when employed in neuromorphic system design.

4 Neuromorphic Design Approach and Learning framework

The brain has its processing elements (neurons), organized in such a way that they function and communicate in a highly parallel manner. neuromorphic systems aim to model this in hardware by distributing the memory (synapses) of an SNN architecture into small neurocores such that they are close to processing. Each neurocore houses some neurons, and some part of the synapses along with their weights. During simulation, each core receives and transmit packets using address event representation (AER) through a network on chip communication infrastructure. As illustrated in Figure 4.1 each neurocore in a neruomorphic system integrates

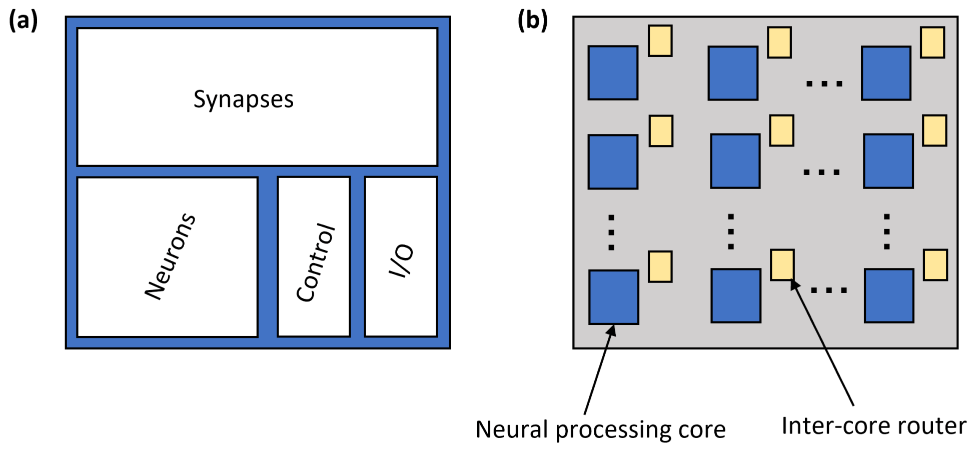


Figure 4.1: Illustration of neuromorphic system design approach.

neuron circuitry for integration of inputs to neuron membrane potential, synapse memory for storing synaptic weights, and possibly neuron states, a control circutry, and finally, input and output interfaces for spike communication. The number of neurons embedded in each neurocore depends on the complexity of the neurons. The organization of a neurocore in a neuromorphic system is influnced by a number of factors which include the SNN topology, and the target application. The choice of platform whether digital or analog, depends on the prefernce of the designer. Several approaches can be employed in the neurocores for updating neurons. One prevalent approach that have been observed in most neuromorphic systems, is the implementation of a single neuron, which utilizes time multiplexing to simulate several neurons, combining events into a single communication channel. This approach is advantageous since it reduced hardware complexity. However, this approach requires a lot of clock cycles to update the neurons in a single timestep especially in neuromorphic systems with large number of neuron. Furthermore, more complex neuron implementations will require more clock cycles which may not be suitable for some applications [98]. Mimicking the organization of the brain, some design take a different approach by having more neuron circuits on a neurocore, so as to enable rapid parallel operation. In this chapter, we present a parallel neuron update mechanism,

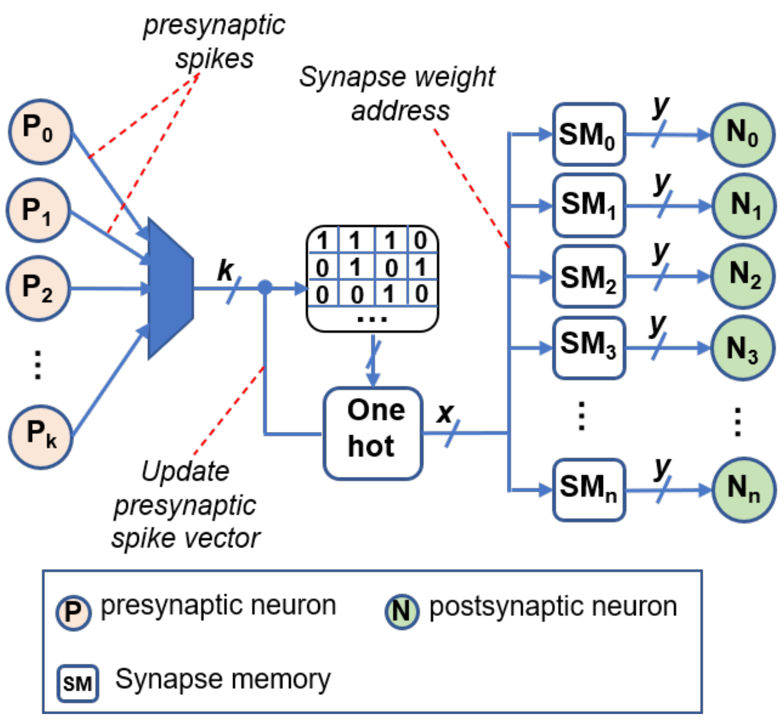


Figure 4.2: The parallel neuron update (PNU) mechanism.

and parallel weight update mechanism that leverages neuron circuits of a neruo core for rapid spike processing.

4.1 Parallel Neuron Update (PNU) Mechanism

As described in previous chapters, communication in SNN are done via spikes, and these spikes are represented with binary values; 1 indicating the presence of spike event, and 0 otherwise. The PNU illustration described in Figure 4.2 shows how spikes from presynaptic neurons P_0 to P_k in a previous layer, are used to udate the post synaptic neurons N_0 to N_n in the next layer. As described in Algorithm 4.1, all spike from the presynaptic neurons are multiplexed onto a single vector with length K, representing the number of presynaptic neurons. The post synaptic layer uses distrubuted synapse memory SM_0 to SM_n, one for each postsynaptic neuron. At the PNU, the presynaptic spike vector is taken through several one-hot operations to find which index of the spike vector has spike event, i.e the presynaptic neuron with spike. The spike vector index with event is used as synapse memory address, for the synapse memory of all post synaptic neuron connected to the

Algorithm 4.1: Algorithmic description of the parallel neuron update (PNU) mechanism.

```
  Input: input presynaptic spike (i_pre_spk)
  Output: weight, address (addr)
  /* store input presynaptic spike array                                    */
1 i_pre_spk_reg[0:k] = i_pre_spk[0:k];
  /* Check for spike events in the presynaptic spike array                  */
2 if |i_pre_spk_reg[0:k] == 1 then
3     neuron_update == True;
4 else
5     neuron_update == False;
  /* If events are present in the presynaptic spike array, begin processing the
     events                                                                 */
6 if neuron_update == True then
      /* Obtain memory address of the synapses associated to the neurons to be
         updated using onehot encoding                                      */
7     one_hot_res[0 : k] = one_hot(i_pre_spk_reg[0 : k], res[0 : k]);
8     for i ← 0 to k by 1 do
9         if one_hot_res[i] == 1 then
10            addr = i;
              /* Fetch the weights from the synapse memory address          */
11            weights[0 : n] = $SM_{[0:n]}$(addr, weight[]);
              /* Update postsynaptic neurons $N_{[0:n]}$ with the weights          */
12            $N_{[0:n]}$(weights[0 : n]);
      /* When all weights are updated, set update complete to true          */
13    update_complete == True;
```

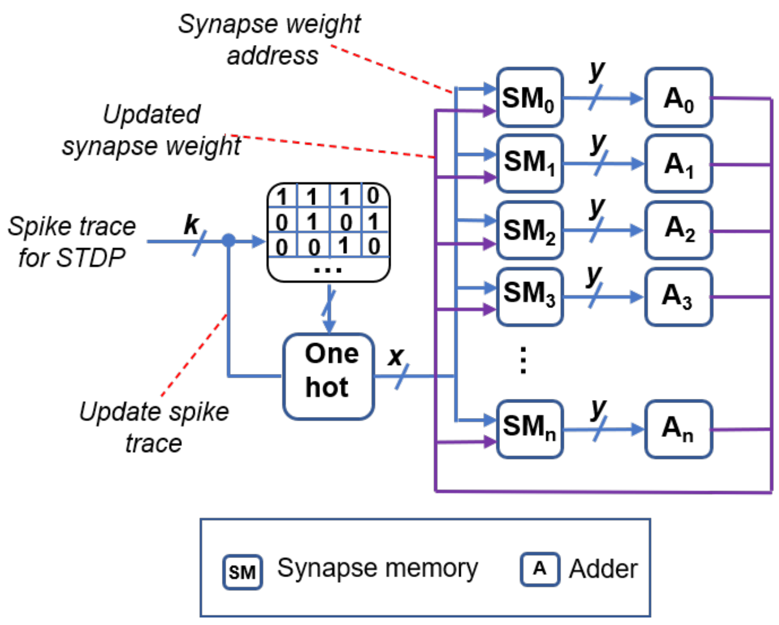


Figure 4.3: Illustration of the parallel weight update (PWU) mechanism

presynaptic neuron that vector index represents. The width of of this address depends on the number of presynaptic neurons K, and is given as $2^x \geq K$. When the address has been received at the synapse memory, the weights stored at those address are sent to all corresponding postsynaptic neurons at once.

4.2 Parallel Weight Update (PWU) Mechanism

The parallel weight update (PWU) mechanismdescribed in Figure 4.3, follows the same operation principle as the PNU. However this mechanism is employed during learining, and is used to update synaptic weights. As described in Algorithm 4.2, when the learning rule eg (trace based STDP) implemented on a neuromorphic system have been determined, the presynaptic spike trace is sent to the PWU, using the same onehot mechanism as the PNU, the synapse memory addresses of the weights to be updated are obtained from the presynaptic spike trace. These addresses are sent to all the synapse memory within SM_0 to SM_n whose neuron is and connected to the presynaptic neurons. The synapse weights stored at those addresses are then sent to the adders A_0 to A_n to be updated, and after the update, the new weight values are written back to the same address. With this mecha-

Algorithm 4.2: Algorithmic description of parallel weight update (PWU) mechanism.

```
   Input: input presynaptic spike trace (i_pre_spk_t)
   Output: weight, address (addr)
   /* store input presynaptic spike trace array                                 */
 1 i_pre_spk_t_reg[0:k] = i_pre_spk_t[0:k];
   /* Check for spike events in the presynaptic spike trace array               */
 2 if i_pre_spk_t_reg[0:k] == 1 then
 3     weight_update == True;
 4 else
 5     weight_update == False;
   /* If events are present in the presynaptic spike trace array, begin weight update
      for the events                                                            */
 6 if weight_update == True then
       /* Obtain memory address of the synapses to be updated using onehot encoding */
 7     one_hot_res[0 : k] = one_hot(i_pre_spk_reg[0 : k], res[0 : k]);
 8     for i ← 0 to k by 1 do
 9         if one_hot_res[i] == 1 then
10             addr = i;
               /* Read the weights from the synapse memory (SM) address         */
11             weights[0 : n] = SM[0 : n](addr, weight_out[], 0);
               /* send weights to adder (A) for update                          */
12             weights_new[0 : n] = A_[0:n](weights[0 : n]);
               /* Write the updated weights back to the SM                      */
13             SM_[0:n](addr, 0, weight_in[]) = weights_new[0 : n];
       /* When all neurons are updated, set update complete to true             */
14     update_complete == True;
```

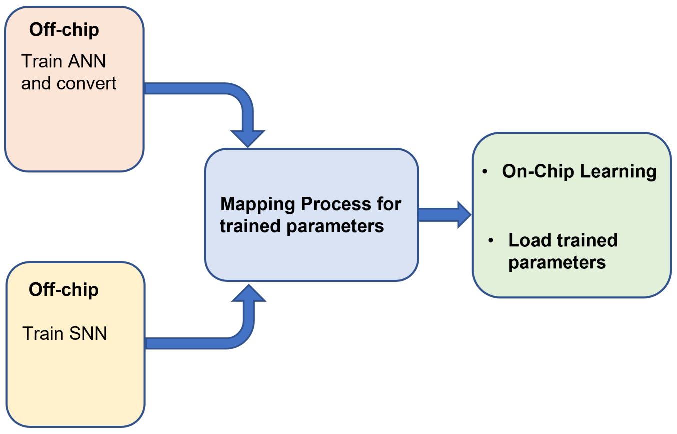


Figure 4.4: Learning framework for neuromorphic systems.

nism, a neuromorphic system that incorporates multiple neuron circuits can explore the parallellism available in SNN.

4.3 Neuromorphic learning framework

In comparison with ANN, SNN has fewer learning algorithms and techniques. Some of these techniques employed in neuromorphic system design are summarized in the learning framework described in Figure 4.4. The learning freamework consists of three approaches. The first approach involes converting conventionally trained ANN into SNN by adapting the synaptic weights, and other parameters of the spiking neurons, so as to realize a similar input/output mapping as the ANN. To achieve this, we employ the method in [57], by first training the ANN with backpropagation. The trained parameters are then transformed into an SNN model similar to the target neuromorphic system. The input/outputs are also encoded with a suitable spike coding scheme. The resulting SNN weights are normalized, quantized into a lower bit representation required by the target neuromorphic system, and afterwards, they are mapped to it. This method however is not always feasible, because not all ANNs can easily be converted into SNN.

The second approach directly trains an SNN in software using variants of backpropagation for supervised learning. The resulting trained weights are then adapted, normalized, and mapped to the neuromorphic system. This approach however, is not aimed at achieving biological plausibility. The thrid approach focuses on on-chip learning with an on-chip trace based spike-timing-dependent-plasticity (STDP) learning rule presented in chapter 5. This is achieved by either performing learning directly on-chip, or designing the environment of the neuromorphic system in a simulator (Bindsnet [99]), and performing learning with same STDP model inmplemented on chip. After the learning is completed in the simulation environment, it can also be done on the neuromorphic system for validation.

4.4 Conclusion

In summary, this chapter has presented the design approach employed in the design of the proposed digital neuromorphic system. The PNU mechanism enable parallel neuron update for quick spike processing, and the PWU mechanism enables parallel of update of weights for rapid learning. The learning framework presented in the summarizes the various approaches that are employed in performing learning on the proposed digital neuromorphic system. In the next chapter we present how these design and learn approach are integrated to build the light-weight neuro processing cores of the proposed system.

5
Light-weight Neuron-Processing Core Architecture

IN REALIZING A SCALABLE MULTICORE NEUROMORPHIC ARCHITECTURE, an efficient design of the of the individual cores that make up the architecture is required. Therefore we present in this chapter, the architecture and design of a light-weight spiking neuron processing core, suitable for a multi-core neuromorphic architecture.

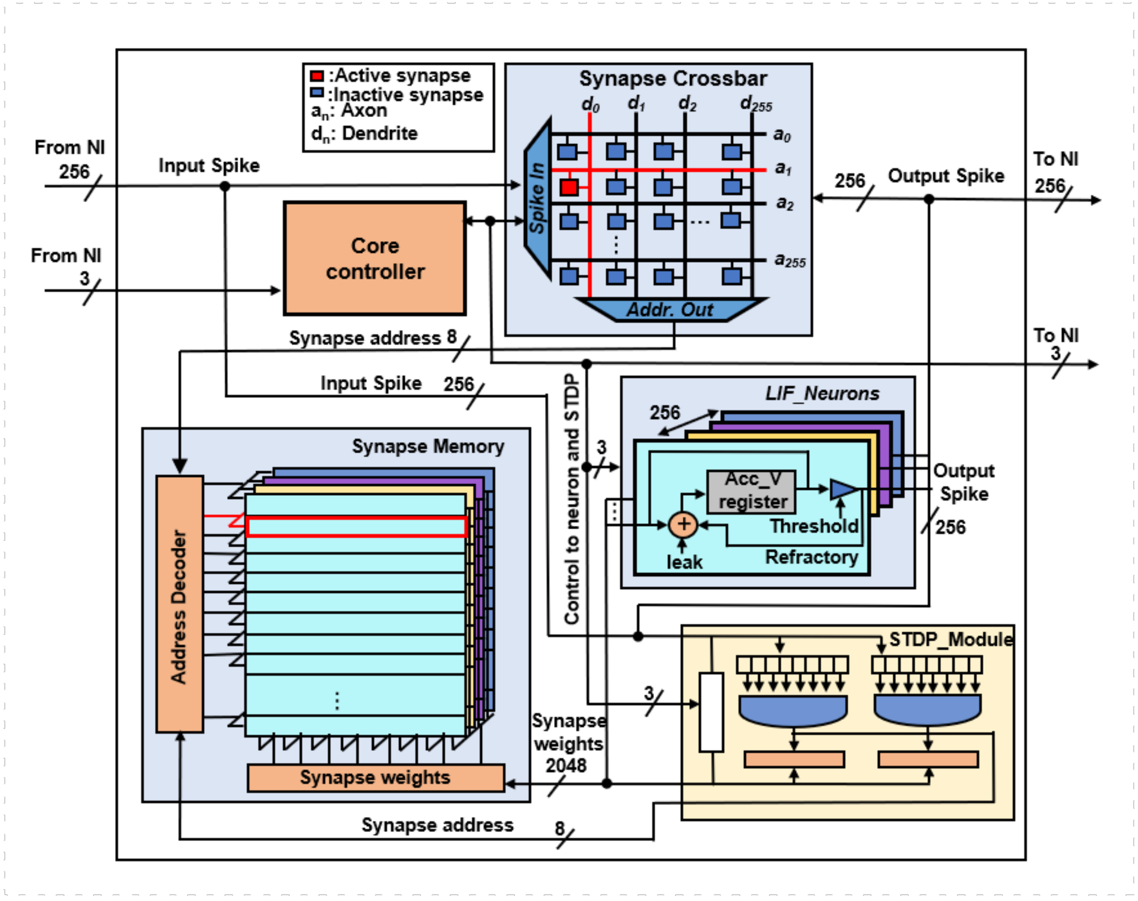


Figure 5.1: High level view of the spiking neuron processing core.

5.1 Spiking Neuron Processing Core Architecture

The SNPC is shown in Fig. 5.1c, and it contains 256 physical LIF neurons, a crossbar-based synapse, a control unit, a synapse memory, and an STDP learning module [100]. In this section, we describe these components and how they operate.

5.1.1 Synapse Crossbar

The neurons in the biological brain are arranged in a 3D manner, and this allows for more connection between them. To attempt having the same level of connection in the SNPC, a crossbar approach (which aims at merging memory and neuron update details) is used to implement the synapses. An illustration of the synapse crossbar can be seen in Figure 5.1. The crossbar which is a composite of an array of axons and dendtrites

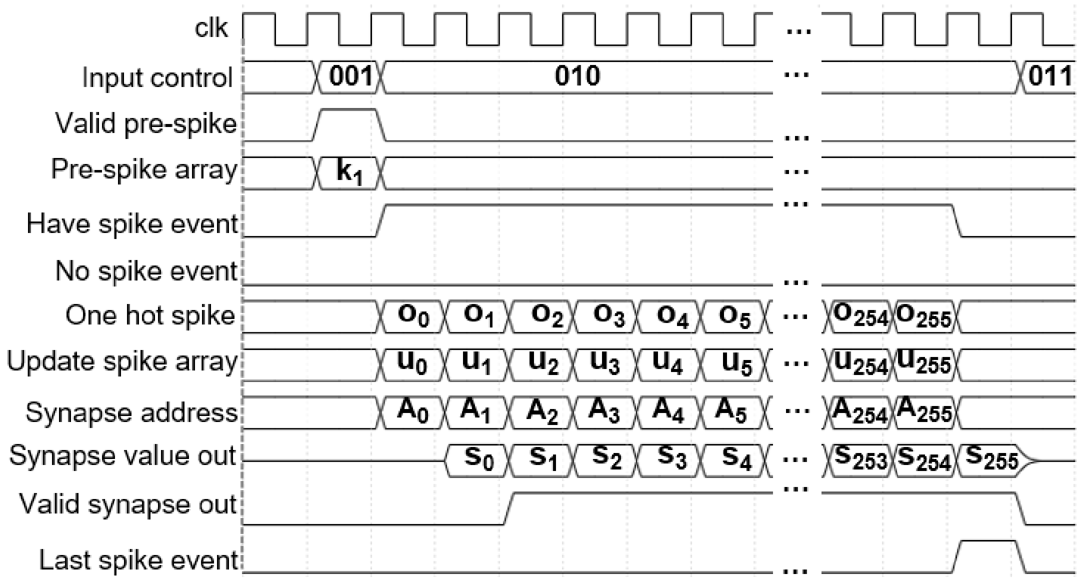


Figure 5.2: Signal diagram of crossbar operation.

crossing each other in an othorgonal manner with the horizontal representing the axon, and the vertical representing the dendrite of the LIF neurons. Synapses are represented by the intersections, and the value of those synapses are stored at the synapse memory. The synapse memory is implemented with an on-chip SRAM. A total of 65K synapses are represented in the synapse crossbar. Figure 5.2 illustrates the pipelined process of fetching synaptic weights from synapse memory after its address has been determined from the input spike [101].

An input spike train to the SNPC is received as a vector. During operation, the received presynaptic spike vector is sent to the crossbar. At the crossbar, an OR operation is performed on the vector to check for the presence of spike event(s). In the presence of spike event(s), the PNU (parallel neuron update) mechanism is applied to the spike vector, to get the memory address of the corresponding synapses whose values are stored on the in the synapse memory. When the memory address have been gotten, they are used to fetch the synapse values, and sent to the corresponding post-synaptic neurons for computation. In the absence of spike event, the neuron update is not performed.

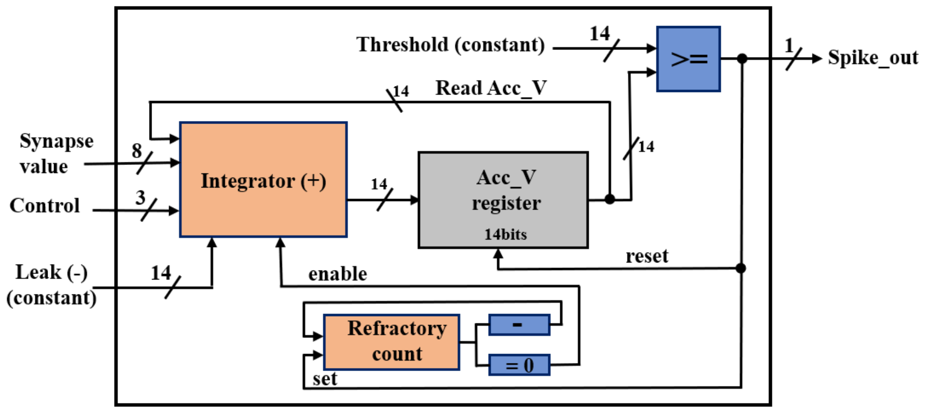


Figure 5.3: Block diagram of the leaky integrate-and-fire neuron.

5.1.2 Synapse Memory

The SNPC synapse memory stores the weights of each of the 65K synapses represented in the crossbar. It is implemented with SRAM, and each synapse is represented with 8 bits. With a fan in of 256, a single neuron utilizes 256×8 bits SRAM. The SNPC design embeds 256 physical LIF neurons, and each of the neurons has an SRAM. This design approach enables the synapse weights to be read from the synapse memory, to update the neurons in parallel.

5.1.3 Leaky Integrate and Fire Neuron

A block diagram of the implemented LIF neuron is described in Fig. 5.3. The neuron membrane potential is accumulated by adding up the input weighted spikes received from the synapse memory in the integrator [102]. The resulting value is then stored in a 14-bit register, which utilizes 13-bit to store the membrane voltage value and 1-bit for overflow. To mimic the leak current found in biological neural membrane, a set leak value that causes decay in the membrane voltage value is received by the neuron when the leak is activated. After the leak ooperation, the value of the membrane potential is compared with the neuron threshold value. If it exceeds the threshold, an output spike is fired. If not, no spike is fired. In the event of an output spike, the membrane potential value is reset to zero, and the neuron enters a refractory period that lasts a few time steps [102]. As illustrated in

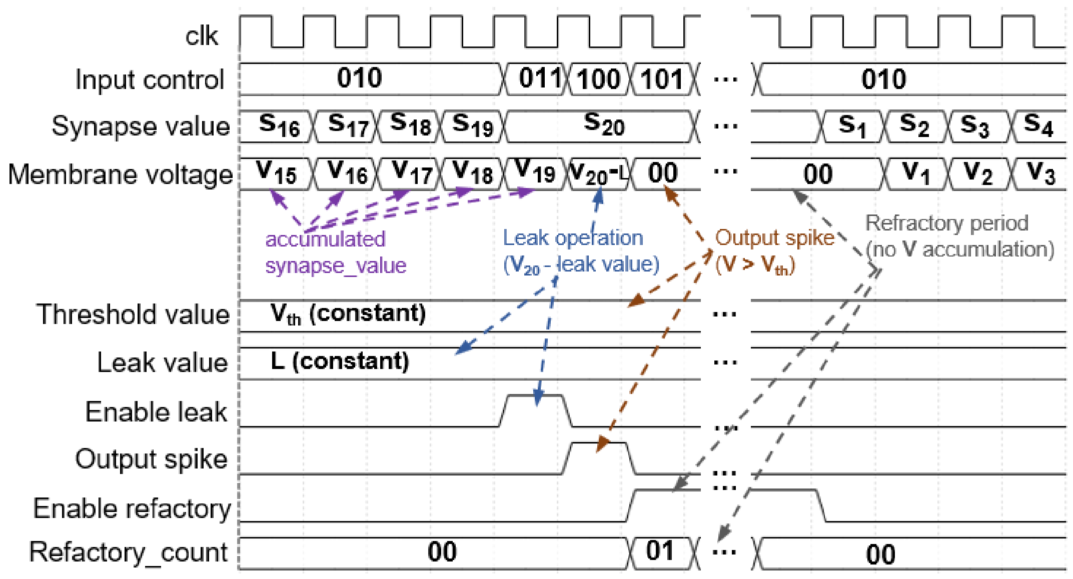


Figure 5.4: Signal diagram of LIF neuron operation.

Fig. 5.4, the refractory count gradually counts down every time step from the set refractory period to zero. While in the refractory period, the neuron cannot accumulate weighted input spikes. However, when the refractory period is over, accumulation of weighted input spikes resume. The LIF neuron model is adopted in this design because it has proved to be effective for most learning applications and is suitable for digital implementation due to its modest hardware cost [103]. The SNPC design enables the embedded 256 neurons to be updated in one cycle using the PNU mechanism.

5.1.4 Learning Module

To achieve on-chip learning in each of the 65k synapses represented in the synapse crossbar, an efficient implementation of the trace-based spike timing dependent plasticity learning rule (STDP) [104] is performed. The update logic of the implemented trace-based STDP is presented in Fig. 5.5. A learning operation requires 16 pre-synaptic spike trace arrays, each from a classification time step. These pre-synaptic spike trace arrays are stored in a circular memory using a 4-bit time step counter. To begin learning, the presence of a postsynaptic spike trace array(s) stored in a memory is verified. Then the pre-

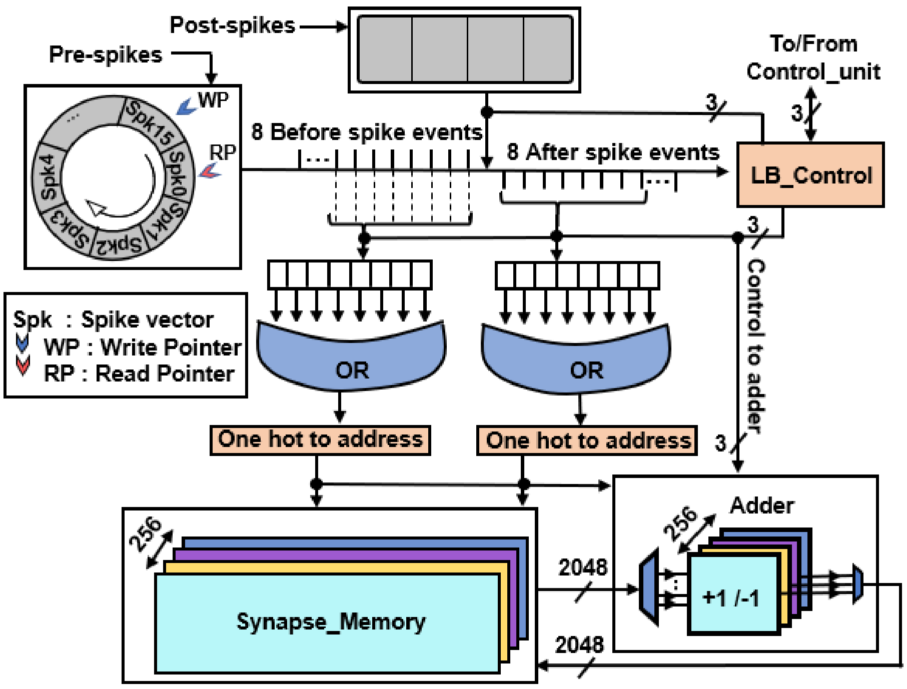


Figure 5.5: An illustration of the trace-based STDP learnig implementation.

synaptic spike trace arrays are grouped into 8 *Before* and 8 *After*, based on their arrival time relative to the postsynaptic spike trace array(s). An OR operation is further performed on the 8 *Before* spike arrays and on the 8 *After* spike arrays to obtain two arrays. Using one hot operation and the postsynaptic spike trace array(s), the associated synapses' memory addresses are obtained from the *Before* and *After* arrays. The corresponding synapse values are then fetched from the synapse memory, increased for the *Before* spike events, decreased for the *After* spike events, and then written back to the synapse memory. The implemented trace-based STDP enables the parallel update of synapses using the PWU (parallel weight update) mechanism.

5.1.5 Core Controller

The SNPC core controller is designed as a state machine shown in Figure 5.6, controlling the SNPC's operation. As described in It starts off Idle, which is the first and default state. At this state the SNPC does nothing while waiting for input spikes. The arrival of

Algorithm 5.1: Core controller operation.

```
Input: input spike (i_spk), start, valid_spike
Output: Output spike (o_spk)
if reset == True then
    spike_buf = 0
    state = state_idle
else
    switch State do
        /* SNPC in idle state                                  */
        case state_idle do
            if start == True then
                next_state = state_dwnld_spike
            else
                next_state = state_idle
        /* Download input spike vector                         */
        case state_dwnld_spike do
            if Valid_spike == True then
                spike_reg = i_spk
                next_state = state_comp._spike
            else
                next_state = state_dwnld_spike
        /* Process input spikes                                */
        case state_comp_spk do
            if xbar_last_spike = True then
                next_state = state_leak
            else
                PNU = True
                next_state = state_comp._spike
        /* Enable decay in neuron membrane potential           */
        case state_leak do
            LIF_neuron(0, leak, 0)
            next_state = state_fire
        /* Check fire condition for output spike               */
        case state_fire do
            o_spk =LIF_neuron(0, 0, fire)
            next_state = state_upld_spk
        /* upload output spike to network interface (NI)       */
        case state_upld_spk do
            NI =o_spk; next_state = state_learn
        /* If learn condition is met, else go to idle          */
        case state_learn do
            if learn_valid == True then
                PWU = True
                next_state = state_idle
            else
                next_state = state_idle
```

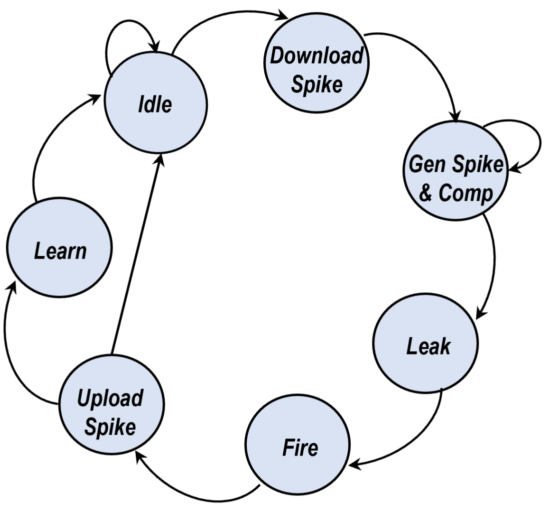


Figure 5.6: State machine of core controller.

a presynaptic spike vector is preceded by a signal which triggers the control unit to change its state to the second state, allowing the presynaptic spike vector to be received. After the presynaptic spike vector is received, the third state is enabled. In the third state, the synapse crossbar is activated to identify and update the postsynaptic neurons by sending the corresponding synapse values stored in the synapse memory. After the last postsynaptic neuron has been updated, a signal is sent from the crossbar to the control unit to move to the fourth state. The fourth state enables decay in the value of the postsynaptic neurons' membrane voltage, and is followed by a fifth state that triggers the postsynaptic neurons to check for an output spike by comparing the value of their membrane potential with that of the set threshold. At the sixth state, the spikes generated by the neurons are sent to the postsynaotic neurons. When the sixth state is complete, the condition for learning is examined. If satisfied, the learning, which is the seventh state, begins, and when finished, a signal is sent to the control unit to return to the default state. If the learning condition is not satisfied, the learning is skipped to the default state.

5.2 Evaluation Results

5.2.1 Evaluation Methodology

In this section, we evaluate the performance and hardware complexity of the SNPC. The SNPC was designed in Verilog-HDL, synthesis and layout were made with Cadence tools, using the NANGATE 45nm open-cell library [105] as the standard cells. Open-RAM [106] was used for generating the system memory and TSV from FreePDK3D45 [107].

For formance evaluation, the SNPC was used to classify the modified national institute of standards and technology database (MNIST) [108]. The MNIST dataset is made up of hand written digits from 0 to 9, and contains a total of 60K training images and 10K inference images. Each MNIST image is a 16×16 grayscale image. The rate coding scheme was with poisson distribution was used to encode the images into spikes. The hardware complexity of the SNPC in terms of power and area is also analyzed.

The poisson model of spike train generation is described in [109] as : Given a time period $(0, T)$, the probability of a spike occurring during that period is given as:

$$P\{1 \text{ spike during } \delta t\} \approx r\delta t \tag{5.1}$$

The period $[0, T]$ is first subdivided into series of intervals each of duration δt. Next, a sequence of random numbers $x[i]$ uniformly distributed between 0 and 1, are generated. Then for each interval, if the value of $r\delta t \geq x[i]$, a spike is generated. If not, no spike is generated.

5.2.2 Performance Evaluation

In evaluating the SNPC performance with MNIST dataset classification, two experiments were performed. In the first experiment, the MNIST images were classified using weights that were trained off-chip. As depicted in Figure 5.7(a), these weights were trained as 256:225:10 ANN using backpropagation on matlab, and afterwards converted to SNN using an approach adopted from [110]. Each of the synaptic weights are first

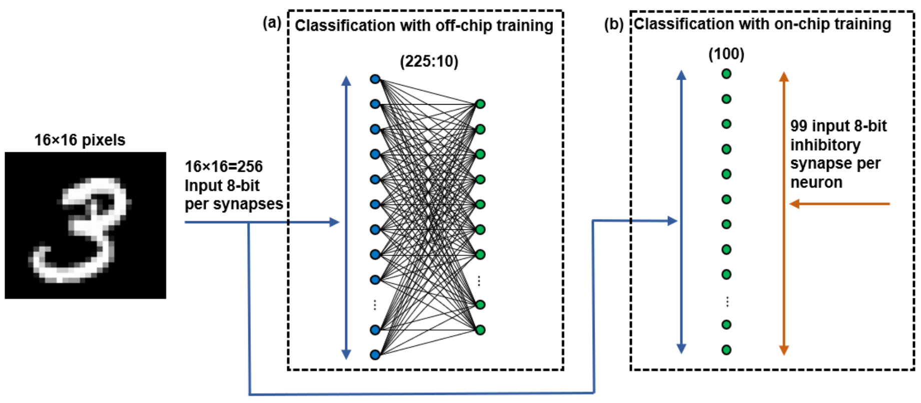


Figure 5.7: MNIST dataset classification on SNPC: (a) Classification with off-chip learning. (b) Classification with on-chip learning.

represented with 8 bits, and subsequently varied to analyze the effect of varying synapse precision on the accuracy. Two SNPC modules were used for this experiment, each representing one layer of the network. The hidden layer of 225 neurons are mapped to the first SNPC while the second layer of 10 neurons are mappeed to the second SNPC. Each of the 10 output neurons is assgned to each MNIST label from 0 to 9. Input spikes are sent directly to the first SNPC, and output spikes from the first SNPC are directly sent to the second SNPC, for processing. Before the classification begins, the trained weights are first mapped to the SNPCs. Afterwards the classification begins and the spike trains which have a total period of 35ms and 35 intervals of duration 1ms for each image, are fed to the SNPC. To determine the output of each image classification, the total number of spikes fired by each output neuron are compared. The label of the neuron with the highest number of output spikes become the selected label. At the end of the classification, the SNPC achieved an accuracy of 96.71%.

For the second experiment, we perform on-chip learning, with a similar network as [110]. As described in Figure 5.7(b), the network has a single layer of 100 neurons with inhibitory recurrent connections to other neurons. With the inhibitory recurrent connections, any neuron that spikes, inhibits the other neurons. As opposed to the off-chip learning network, this network has an output of 100 neurons, and each output neuron is assigned after training to the label in which it spikes the most. At the end of the classification, the SNPC achieved an accuracy of 72.9%.

Each of the synaptic weights are first represented with 8 bits, and subsequently varied to analyze the effect of varying synapse precision on the accuracy, and area of the SNPC

In the second experiment, the MNIST images were trained and classified using the on-chip trace-based STDP learning module on a 256:100 network as described in Figure 5.7(b)

5.2.3 Hardware Complexity Analysis

Table 5.1: Hardware complexity report.

System	SNPC
Power Estimation(mW)	57.55
Area(mm^2)	1.290
FrequencyMHz	142

As described in Table 5.1, the SNPC occupies a silicon area of 1.290mm^2 excluding pads. At 1.1V, 25°C and a clock frequency of 142MHz, the SNPC consumes a total dynamic and leakage power of 57.55mW. Because of the substantial amount of memory access required for moving synapse values during learning and classification, most of the energy consumed by the SNPC can be attributed to memory access.

5.3 Conclusion

In this chapter, we introduced the architecture and design of spiking neuron processing core (SNPC), desribed its components, and their operation. The hardware complexity in terms area and power was evaluated, and a performance evaluation with MNIST dataset classification was also performed with on-chip and off-chip learning. In the next chapter, an integration of the SNPC into a scalable 3D-NoC-based fault-tolerant interconnect is presented.

6

On The Design of Adaptive Scalable 3D-NoC-based Neuromorphic System

Recent progress in tract-tracing connectomics has helped deepen our understanding of the topology of the brain [111] and has buttressed findings which reveal that the anatomical topology of the brain network is organized as a three-dimensional small world networks. Such a network is typified by dense local clustering of neurons with short connection lengths, and a few long-range connections between clusters [112]. As described in previous chapters, the brain is formidably connected, and to attemt realizing a fraction of that level of connection while keeping a reasonable communiction cost, a scalable architecture which also assumes the 3D structure

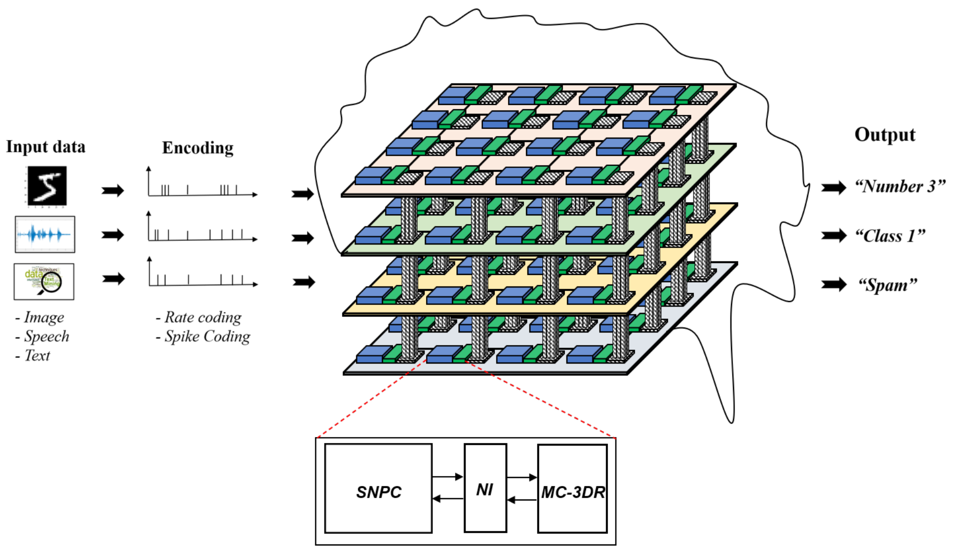


Figure 6.1: High level view of the proposed NASH system.

of the brain is imperative. As a result, we present in this chapter, a scalable spike-based neuromorphic system (NASH), based on three-dimensional network on chip communication architecture. We begin by describing the overall architecture, then the architecture of the individual components. Next, we present the various routing algorithms employed in routing spikes within NASH system. Afterwards, we present the network performance, system performance, and hardware complexity evaluation of NASH. Finally, we present the conclusion.

6.1 Overall Architecture

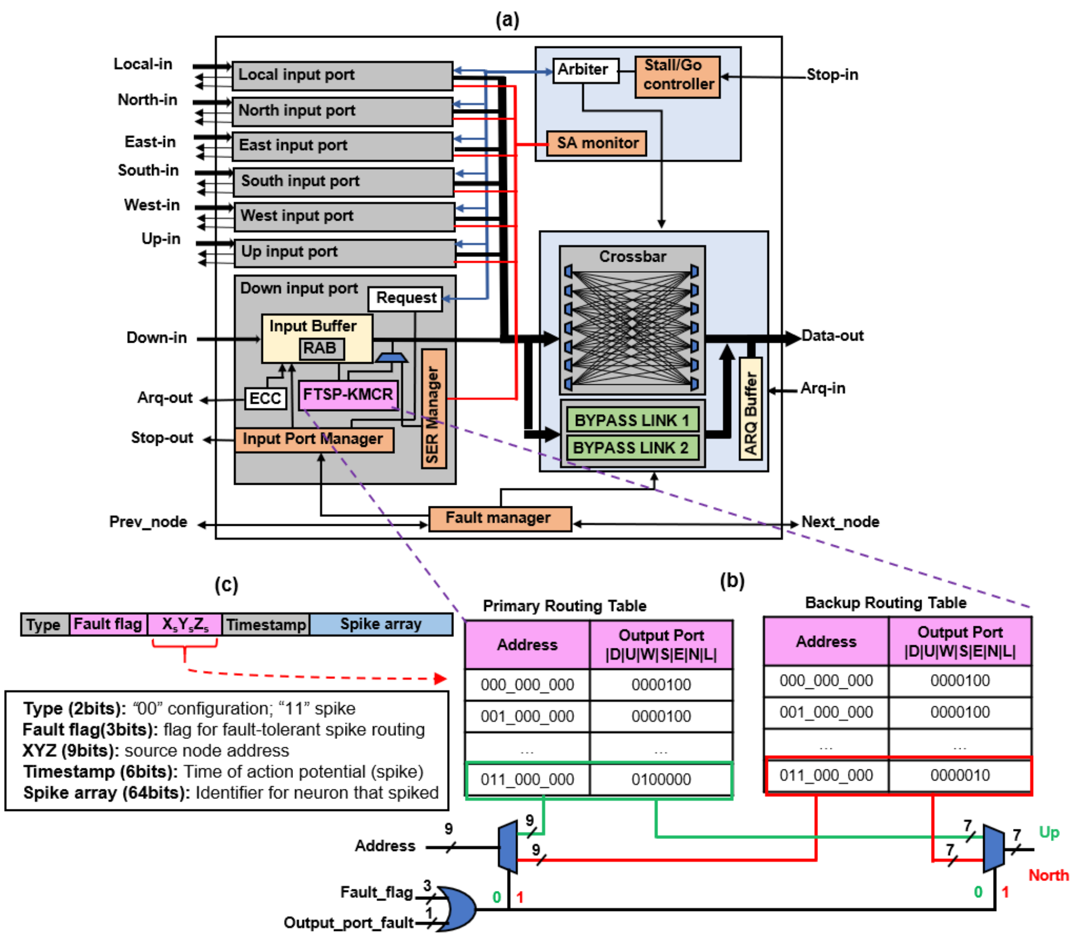
Figure 6.1 shows the overall architecture of the proposed NASH in a $4 \times 4 \times 4$ configuraton [113]. NASH is a 3D mesh-based architecture composed of several nodes. Each node is made up of a spiking neuron processing core (SNPC) adopted from chapter 4, a multicast 3D router (MC-3DR) and a network interface (NI). An output spike from a LIF neuron while performing a task is sent to post-synaptic neurons, which could be in another node within the 3D network. The spike is encoded at the NI as a packet and sent to the local MC-3DR, which routes it to the destination node(s) where the post-synaptic neurons reside. At the destination node(s), the packet is decoded into spikes at the destination NI, and sent to the local SNPC. In the following subsections, we describe in details the various components of proposed adaptive scalable 3D-NoC-based neuromorphic system.

6.1.1 Topology

The NASH architecture, employs a 3D-Mesh topology to enable scalability and high performance [114]. The authors in [115] analyzed several 2D communication architecture (i.e., mesh NoC, tree, shared bus, and point-to-point) for neural networks using various spike routing algorithms. Their evaluation result show that the 2D mesh NoC with multicast routing is good for SNNs. As described in [116], the 3D-mesh communication architecture outweighs 2D counterpart, and this is verified in the evaluation results described in subsequent sections.

6.1.2 Fault-Tolerant Three-Dimensional Multicast Router

Figure 6.2 illustrates the architecture of the FTMC-3DR, which is based on [79, 117]. It consists of seven input and seven output ports: one port for connecting to the local network interface with which spike packets can be injected into or received from the network, four for connecting to neighboring routers in the north, east, south, and west direction using the intra-layer links, and the remainder for up and down ports are used for connecting to the routers in the closest layers, through the TSVs. Each FTMC-3DR routes multicast packets using four pipeline stages. In the first stage, which is buffer writing (BW), the spike flit received either from the NI or neighboring routers are stored in the buffer of an input port [118]. In the second pipeline stage, routing calculation (RC) obtains the source address of the stored flit, and uses it to derive the next address where the flit is to be delivered, which is either in the X, Y, or Z dimension. After this address is derived, the third stage begins. In this stage, the switch-allocator (SA), which handles a stall/go flow-control, and the matrix arbitration (matrix-arbiter scheduler) are triggered to allocate an output port through which the flit, will be delivered to the next routeror local SNPC. After the right output port has been allocated, the fourth stage, crossbar traversal



Type (2bits): "00" configuration; "11" spike
Fault flag(3bits): flag for fault-tolerant spike routing
XYZ (9bits): source node address
Timestamp (6bits): Time of action potential (spike)
Spike array (64bits): Identifier for neuron that spiked

Primary Routing Table

Address	Output Port |D|U|W|S|E|N|L|
000_000_000	0000100
001_000_000	0000100
...	...
011_000_000	0100000

Backup Routing Table

Address	Output Port |D|U|W|S|E|N|L|
000_000_000	0000100
001_000_000	0000100
...	...
011_000_000	0000010

Figure 6.2: Architecture of multicast 3D spike router.

(CT) begins, and the packet traverses the crossbar through the allocated output port, to the next destination.

6.1.3 Fault tolerance

The reliability if neuromorphic systems is considered important, especially for critical applications like biological, space and autonomous systems. As a result, incorporating fault-tolerance in neuromorphic system design is imperative. There are a number of possible faults that can occur in such a 3D-NoC based neuromorphic architecture and they are generally classified as soft errors, hard faults and TSV defects [119, 120]. Some possible hard faults and soft errors are illustrated in Figure 6.3, where Figure 6.3(a), describes a faulty open wire with a crack in it. This fault leads to an increase in resistance, and prevents

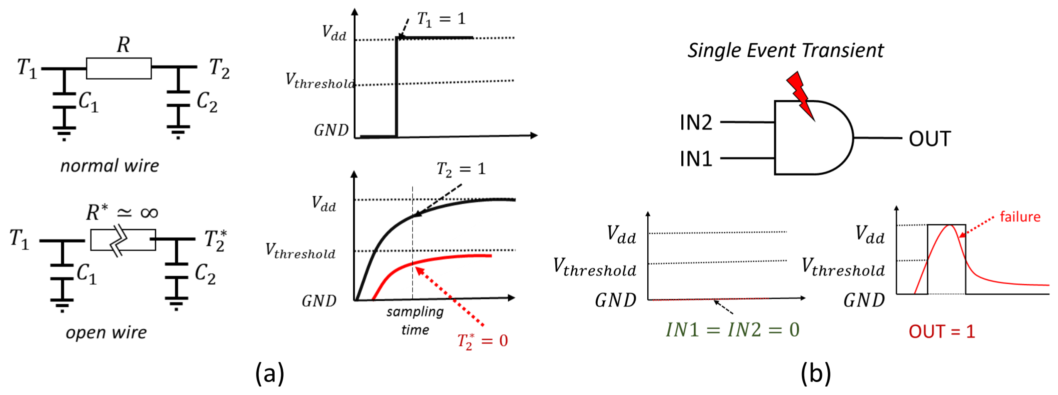


Figure 6.3: An Illustration of hard faults and soft errors [8]: (a) Open wire defect (hard fault); (b) Single event transient (soft error).

the wire from reaching the threshold votage, which results in the transmission of wrong signals. In Figure 6.3(b), shows the effect of a single-event particle transient fault on an AND gate, where a wrong value is sampled due to a glitch [121].

Hard faults such as permanent and intermittent, mostly occur at manufacturing. However, intermittent faults can also recurrently occur during operation, and cease after some time. At high temperatures, this fault can lead to timing violations, and at cooler temperatures, maintain normal operation, leading to inconsistency in operation. As a result, they are often not detected during testing. For transient faults, repeating the faulty operation, and code-based techniques such as error correction code [122] can be employed to tackle them. Soft errors on the other hand emerge as a result of electron-holes caused by charged particles from cosmic rays [123]. When accumulated, these charges may change the state of a logic, leading to faulty operation. The effect of soft errors are not so adverse because they occur within a short period. However, they are unpredictable and unavoidable. In general, these faults are recorded be the cause of about 80% of the failures that occur in systems [8, 124], and

The FTMC-3DR utilizes a complex recovery technique which reconfigures the system using redundant structural resources to handle hard faults that may occur in the input-buffers, crossbar, and links [125, 126]. In this subsection, we discuss these techniques and how they are realized in the router

6.1.3.1 Fault-tolerant buffer

To handle deadlock problems that may occur in the input buffers, a mechanism called Random Access Buffer (RAB) was inherited from [10]. As shown in Figure 6.4, this mechanism enabled by the buffer controller in each input port, which tasked with detecting deadlock. It does this by handling the assignment of read and write addresses where a timer is used to observe a request being processed. When this request is not seved after a certain amount of time, a deadlock notice is flagged [127]. The buffer controller then proceeds to read the head of the next packet in the buffer, and if its requested output port is different, a request is then made to the switch allocator to grant it. When this request is granted, the corresponding packet is then read from the buffer, and free slots are created to store other incoming packets [127]. After the incoming packets have been written to the buffer, a recheck is performed for the blocked packet, and served.

The RAB along with detecting deadlock, is also able to detect transient, intermittent, and permanent faults in the input-buffer [9], which enable it to recover from transient, intermittent, and permanent faults that may occur in the input buffer.

6.1.3.2 Fault-tolerant crossbar:

In handling faults that may occur in the router crossbar, we employed a mechanism called Bypass-Link-on-Demand (BLoD) [10] which was proposed in our previous work. As described in Figure 6.5, the BLoD provides additional escape links which are used to bypass the crossbar in the event of faults.

6.1.3.3 Fault-tolerant TSV

The routers employed in the NASH system require inter-layer connection, and this is achieved using TSVs [128] that serve as inter layer link. For enable each inter-layer connection, a router needs a set of TSVs. Consequently, a router is equipped with four TSV-clusters, and can also utilize a maximum of four nearby TSV-clusters. Therefore, a total of eight TSV clusters are employed to handles inter-layer connections for a router. because of high fault rate and clustering distribution, which occur in TSV, fault-tolerance

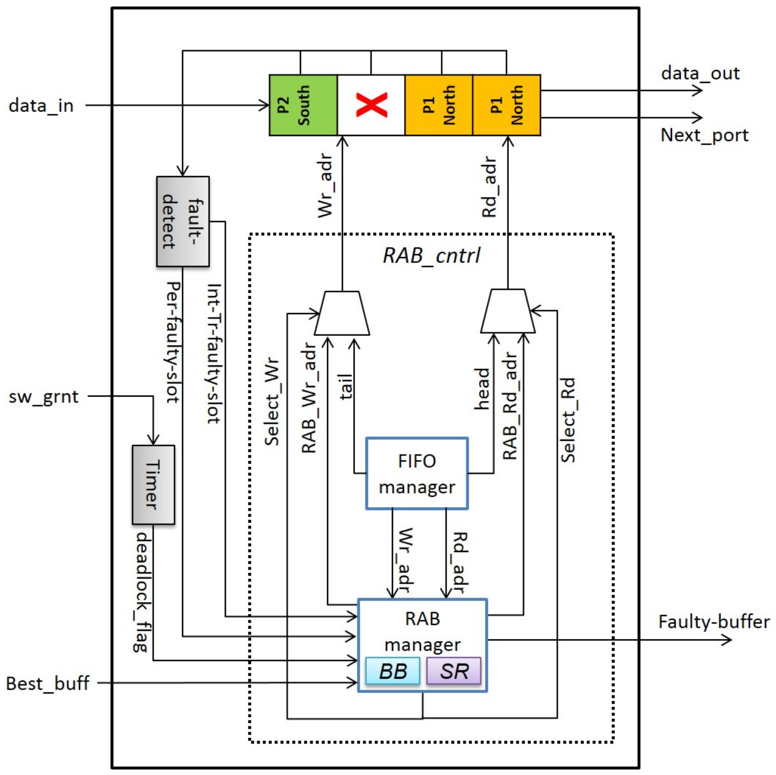


Figure 6.4: Block diagram of random access buffer (RAB) [9].

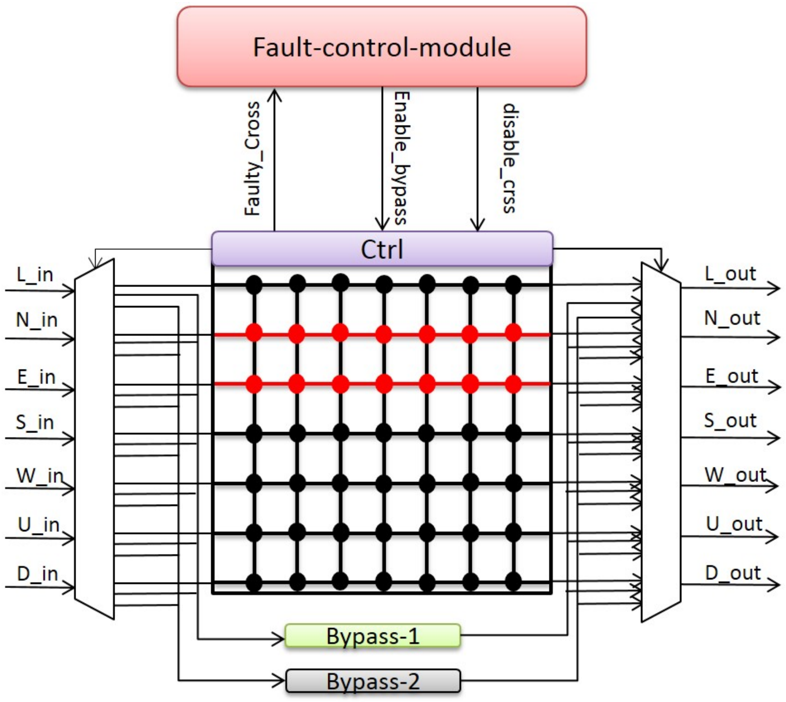
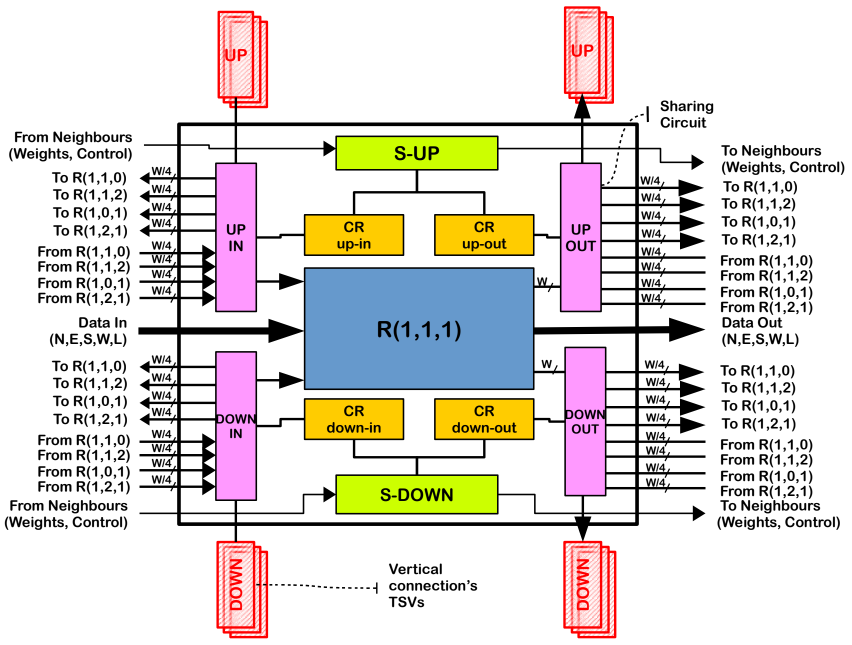


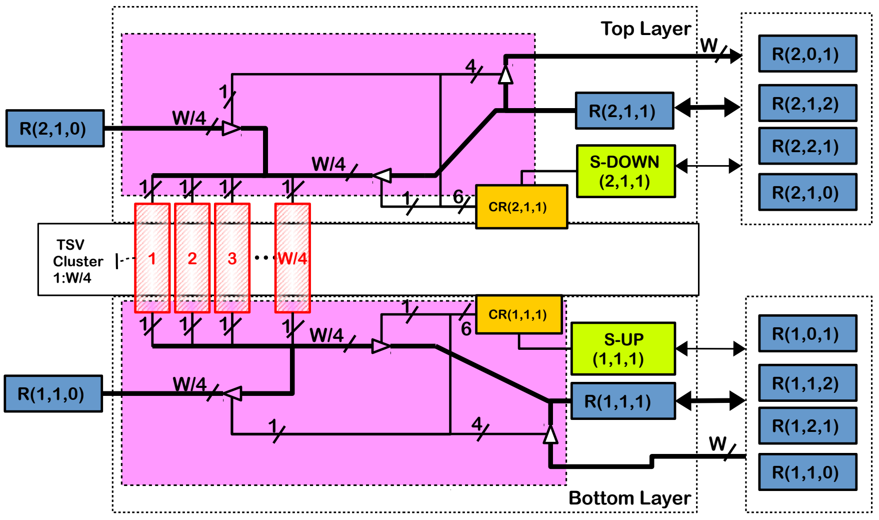
Figure 6.5: Block diagram of bypass-link-on-demand [10].

in TSVs have become imperative. To handle faults that may occur in the TSV, we adopt from our previous work, an architecture sharing TSV clusters [129][130]. These TSVs do not follow the conventional method where they are grouped together, rather, they are divided into four groups, and in the event that a TSV-cluster of a router becomes faulty, to satisfy the SNN timing constraint, one out of its four neighboring clusters can be used as a replacement with the help of supporting modules that implement a sharing algorithm, instead of having redundant ones. To enable sharing, the verticval connections of each router are assigned weights, and these weighs help to decide its priority during sharing/borrowing. These weight are ususally assigned during the design process. However, they can also be updated by a dedicated module. By changing the weights of the routers, different TSV mappings can be created. The sharing algorithm first enables all routers in the system share information regarding their weighs, and the status of their TSV clusters with neighboring routers [131]. When this is completed, a mapping process begins where routers with defected TSV clusters determine which of its neighbors is a possible candidate to borrow from. This choice however, depends on whether the candidate's weight is smaller than its own weight, the TSV's of the candidate are healthy and has not already been borrowed by another router, and the it has the least weight among all the considered candidates. These criterias help each router determine a candidates to borrow from. The borrowing process is completed when the router sends a borrow request to the determined to utilize its TSV cluster as a substitute for its faulty one. However, if no possible candidate is found, the inter-layer connection of the router is disabled. As illustrated in Figure 6.6, there are two similar supporting modules; S-UP and S-DOWN that are employed to manage this sharing process. For both inter-layer connections; up and down, a total of four configuration (two each) are adopted for the input and output ports.

In the event that a TSV-cluster is borrowed, the borrowing router manages this TSV until it is disabled, after which the borrowed TSV-cluster becomes free, and is returned to the lending router. Consequently, a similar chain created in borrowing the TSV cluster, is also created in also created in returning it.



(a)



(b)

Figure 6.6: Architecture of fault-tolerant TSV: (a) router wrapper (b) sharing TSV architecture with TSV cluster (red rectangles) and sharing arbitrators (*S-UP*, *S-DOWN*) [8].

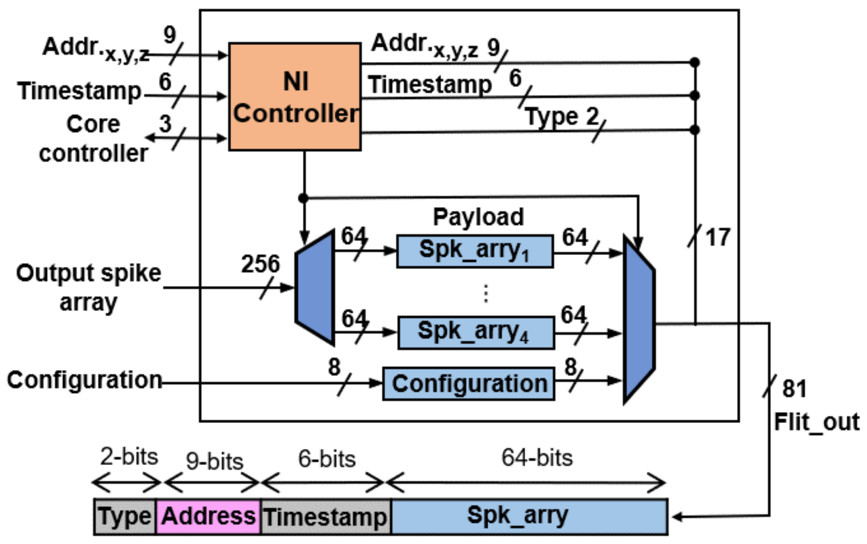


Figure 6.7: Network interface encoder.

6.1.4 Network Interface

The Network interface is the communication link between the SNPC and the FTMC-3DR. It is composed of two modules: Encoder and Decoder. The encoder which is described in Figure 6.7 receives output spike vector from the SNPC and encodes it into flits of packets using an 81-bit flit format described in Figure 6.2 before sending to the 3D-MCR to be routed to the destination SNPC. The first 2 bits of the flit indicate the "*Type*" of the flit: "00" for configuration and "11" for the spike. The next 9 bits (3 bits each for X, Y, and Z dimensions) are used to represent the address of the source neuron. The following 6 bits are a record of the time in which the source neuron fired the spike. The last 64 bits are used for the spike vector. In contrast to the encoder, the decoder which is described in Figure 6.8 receives packets from the MC-3DR and decodes it into spikes before sending to the SNPC.

6.2 Spike Routing Algorithms

To ensure efficient communication of spikes within NASH, the K-means based multicast routing algorithm (KMCR), and the shortest path K-means based routing algorithm

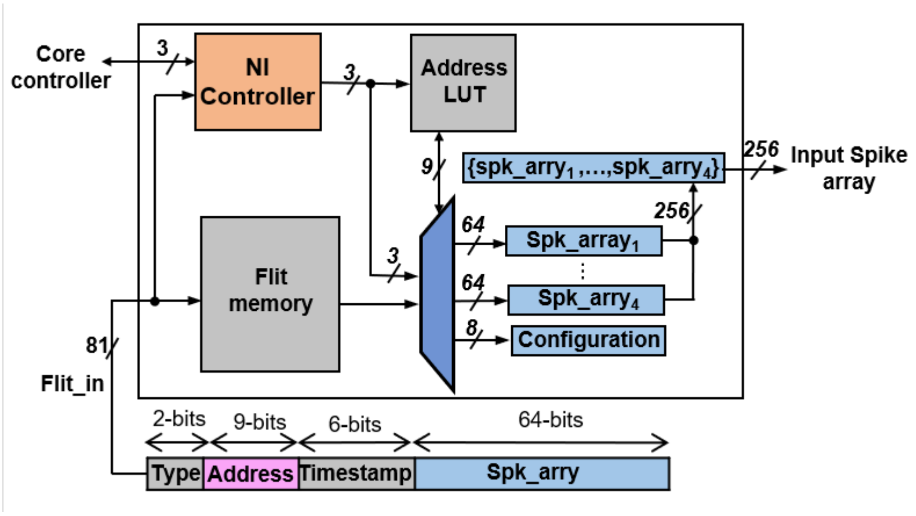


Figure 6.8: Network interface decoder.

were adopted from a previous work [117]. In this section, we describe these algorithms and their operation.

6.2.1 K-means Based Multicast Spike Routing Algorithm (KMCR)

The KMCR is a combination of K-means clustering and tree-base routing. The algorithm routes packets by first partitioning destination nodes into subgroups. From each subgroup, a centroid a node which has a minimum mean distance to other nodes in the subgroup is selected. This distance is calculated using the manhattan distance. After the centroids have been decided, a routing tree is formed connecting the source node and the centroids. To complete the routong path, a spanning sub-tree is formed from the centroid to the destinations in each subgroup.

6.2.2 Shortest Path K-means Based Multicast Spike Routing Algorithm (SP-KMCR)

The SP-KMCR algorithm operates in a similar manner as the KMCR. However, after destination subsets have been generated, the distance from source to all the destination nodes in the subsets are calculated. Then as illustrated in Figure 6.10, a node in each subgroup with the shortest path from the source is selectedand labelled SP nodes. Unlike the KMCR which sends to the centroids, the SP-KMCR first routes packets to the SP

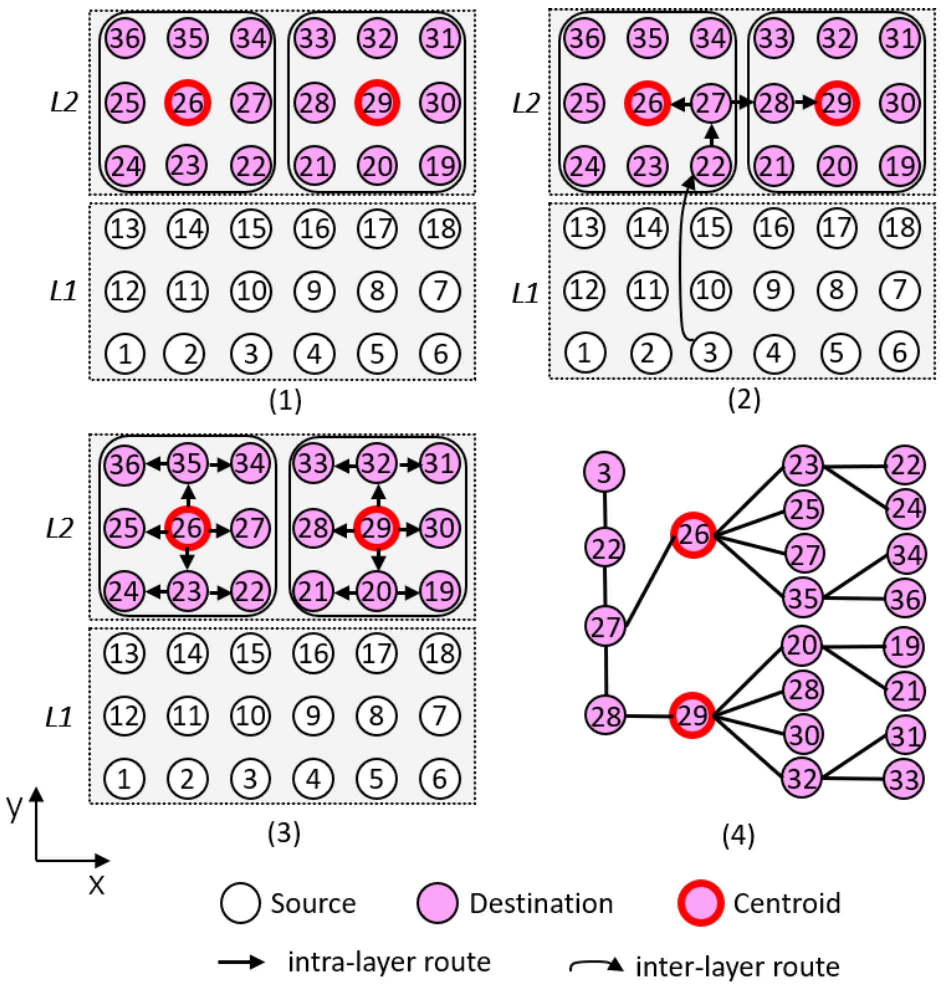


Figure 6.9: Illustration of the k-means based multicastRrouting algorithm.

nodes from the source as shown in Figure 6.10. From these SP nodes, spikes are then routed to destination nodes as described in Figure 6.10. The SP-KMCR despite using shortest path, requires more computations than the KMCR, and since computations for both algorithms are done offline, their runtime overhead ends up the same.

6.2.3 Fault-Tolerant Shortest Path K-means Based Multicast Spike Routing Algorithm (FTSP-KMCR)

The FTSP-KMCR algorithm [132] design is based on the SP-KMCR, and routes packets by first computing offline, primary routing tree and backup branches from a source to destinations. When the offline computations have been completed, the routing tables

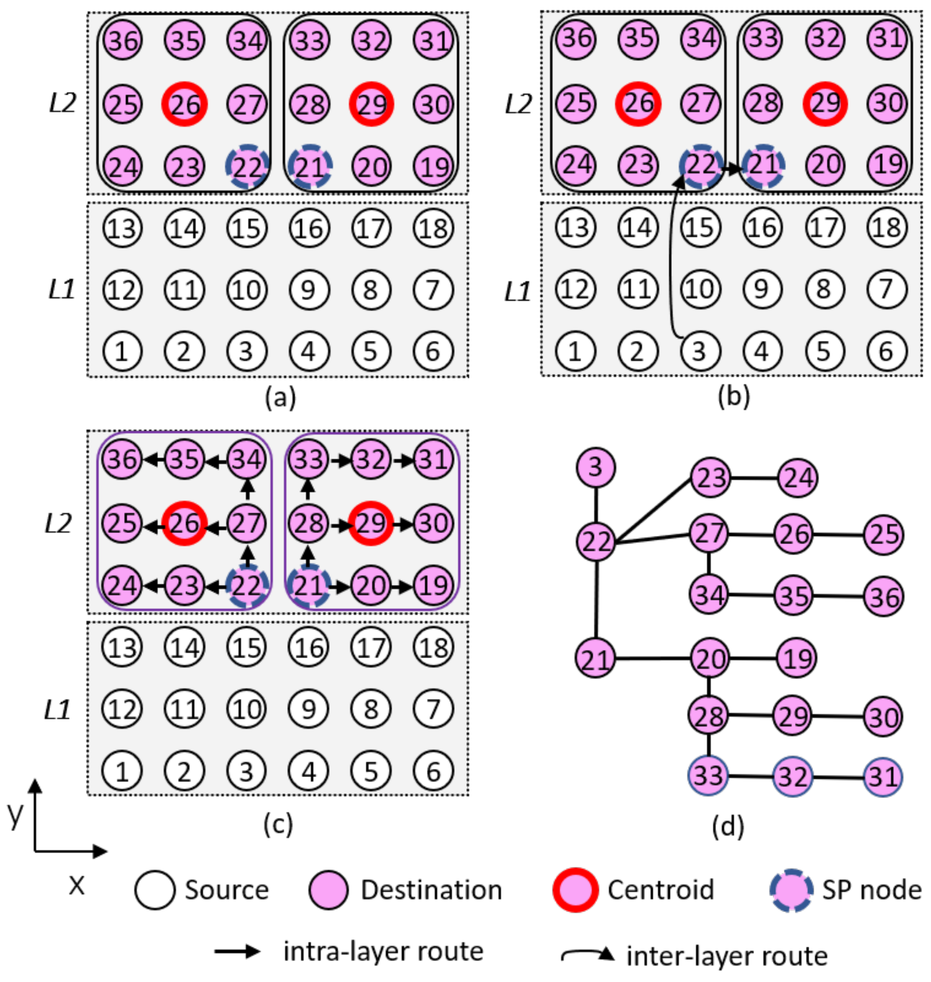


Figure 6.10: Illustration of the shortest Path k-means based multicast rrouting algorithm.

are configured with the computation results. An illustration of the primary route and backup branches are described in Figure 6.10. The primary routes are computed using the SP-KMCR, while the backup branches are alternative routes from the primary route. These backup branches are used as a bypass in the event of faults in the primary link. The aim of the FTSP-KMCR is to mitigate timing violations due to faulty links in SNNs.

In addition to the fault tolerant routing of spikes, a fault management algorithm is also employed by the router to handle received packets. After a packet has been received, the *fault_flag* value is extracted to verify if it is in the primary route or backup branch. The source address of the packet is also extracted to determine its expected primary output port. If the expected primary output port is not faulty, the packet is sent through it to the next router. In the event that the primary output port is faulty, the output port is changed to use the backup branch, and the *fault_flag* value of the packet is updated to inform the next router that the packet is on a backup branch.

6.2.4 Analytical Assessment

The efficiency of NASH given a randomly connected (RNDC) SNN and multicast algorithm can be determined by the distance from source to destination, the efficient bandwidth, the average spike rate (SR) and the maximal spiking frequency described in equations 6.1-6.6 which are expressed in [79] as:

The total number of functional link given as:

$$TL = 3(1-\alpha)\sqrt[3]{n^2}(\sqrt[3]{n}-1). \tag{6.1}$$

Number of hops for each packet from source to destination:

$$TotalDist^{RNDC}_{3DMesh,MC} = C + \overline{Dist}^{RNDC} = C + 3\lambda \tag{6.2}$$

The efficient bandwidth:

$$BW^{RNDC}_{eff,MC} \cong \frac{\overline{w}TL}{\sqrt{n}+3\sqrt[4]{n}}.f_{NoC}.U_{NoC} \quad . \tag{6.3}$$

The average spike rate for each SNPC:

$$\begin{aligned} f^{MC}_{p,out} &= \frac{BW^{RNDC}_{eff,MC}}{n} \\ &= \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)}{\sqrt[3]{n}(\sqrt{n}+3\sqrt[4]{n})}.f_{NoC}.U_{NoC}. \end{aligned} \tag{6.4}$$

The maximal spiking frequency:

$$f^{MC}_{spike,max} = \frac{s}{T_{cycle}.\overline{Dist}^{RNDC}} = \frac{s.f_{NoC}}{3\lambda}$$

. (6.5)

The K ratio derived from (6.4) and (6.5):

$$K = \frac{f^{MC}_{p,out}}{f^{MC}_{spike,max}} = \frac{3\overline{w}(1-\alpha)(\sqrt[3]{n}-1)3\sqrt[4]{n}}{s.\sqrt[3]{n}(\sqrt{n}+3\sqrt[4]{n})}.U_{NoC} \tag{6.6}$$

Where $\overline{Dist}$ is the mean distance between two nodes, $\overline{w}$ the number of wires contained in a link, f_{NoC} the link frequency, U_{NoC} the link utilization factor, T_{cycle} the delay in the link, s the number of neurons in one SNPC, $C \cong \sqrt{n}$ is the number of connections per neuron, $\lambda \cong \sqrt[4]{n}$ is a spatial connectivity constant, and α is the fault rate in the links [133].

6.3 Mapping and Integration

It needs to be noted that the method employed in mapping SNNs onto NoC based systems play a vital role in deploying SNN applications. It not only affects the overall performance of the system, but also its power consumption. Two mapping techniques were proposed by the authors in [134]. The first technique, uses a relatively conventional approach which maps highly communicating tasks together, and the second technique uses an approach that depends on degree of active neurons. In this evaluation, we mapped SNNs onto NASH in a layer-to-layer fashion to take full advantage of the utilized routing algorithm and the 3D mesh NoC topology, as illustrated in Figure 6.11. In this mapping technique, the neurons in the same network layer are mapped to the same NASH layer, and output spikes are sent only to neurons in the next layer. This approach offers multiple parallel connections between layers (vertical connections), less congestion, and low spike latency when compared to the baseline system [125]. In [135] a mapping method named

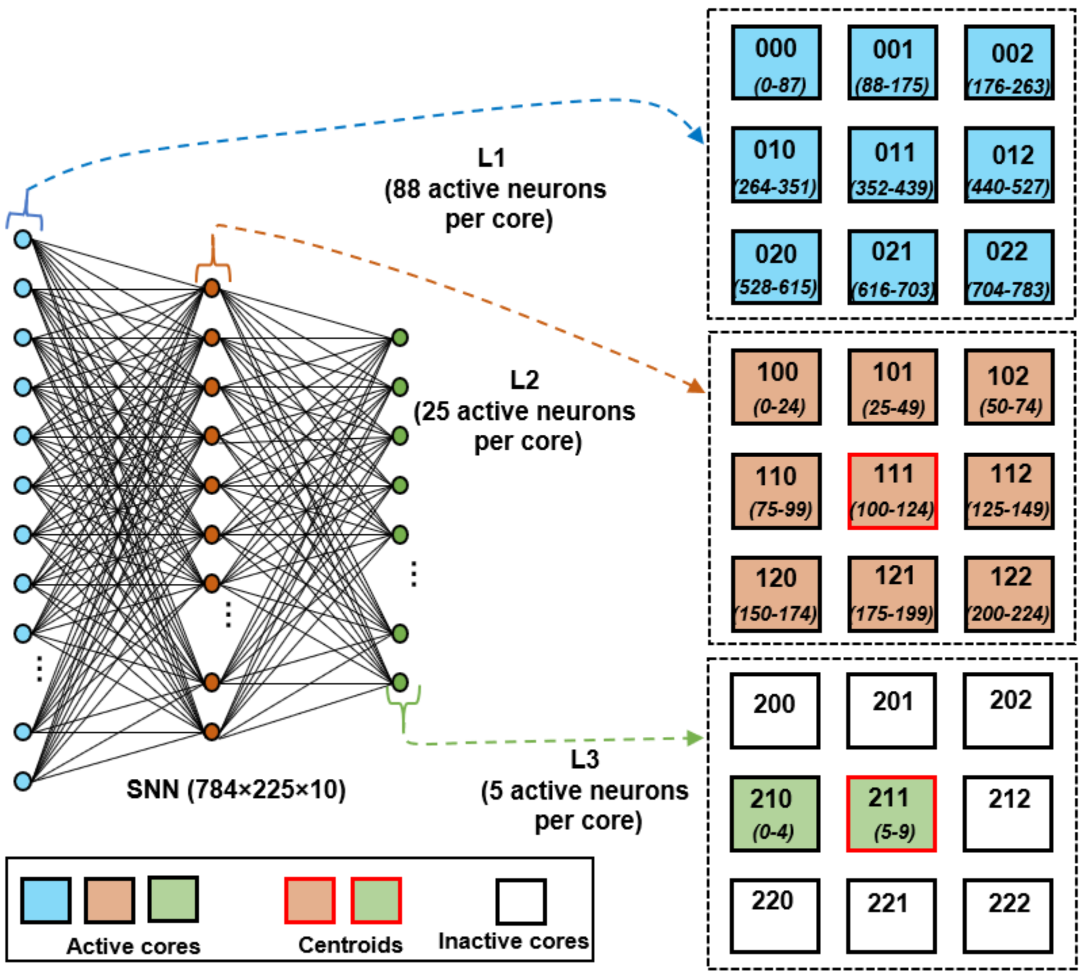


Figure 6.11: SNN mapping for MNIST classification on 3 × 3 × 3 NASH: The first layer of 784 neurons without neural computation is mapped to L1 on 9 cores (88 neurons each for cores 000-021, and 80 neurons on core 022). The second layer of 225 neurons is mapped to L2 on 9 cores (25 neurons each for cores 100-122). The third layer is mapped to L3 on 2 cores (5 neurons each for cores 210 and 211).

MigSpike which uses enhanced migration methods and built in hardware architecture to place spare cluster of neurons, where faulty neurons in the system migrate to in the event of failure. It achieves this by creating chains of migrations for unmapped neurons from their nodes to suitable ones within the system. Also, a max-flow min-cut adaptation and a genetic algorithm approach are employed in in resolving this migration problem.

The NASH configuration of 3 × 3 × 3, and an SNN size of 784:225:10 was used in the evaluation. The input layer of 784 neurons is mapped to the first layer of NASH, utilizing 88 neurons from each of the 9 nodes in the layer. The hidden layer of 225 neurons is also mapped onto the second layer of NASH and utilized 25 neurons from each node in the layer. Finally, the output layer of 10 neurons is mapped to the third layer and utilized five neurons each from two of the nodes in the layer. For a baseline architecture of (9×3) used for comparison, the input layer was mapped to the first nine vertical cores utilizing 88 neurons from each core, the second layer to the second 9 vertical cores utilizing 25 neurons from each core, and the third layer on 2 of the third nine vertical cores using five neurons from each core.

6.4 Evaluation Results

6.4.1 Evaluation Methodology

The proposed system was designed in Verilog-HDL, functional simulation was done with Modelsm, synthesis and layout were made with Cadence tools. For ASIC implementation, we use NANGATE 45nm open-cell library [105] as the standard cells, Open-RAM [106] for generating the system memory and TSV from FreePDK3D45 [107].

6.4.2 Performance Evaluation

6.4.2.1 Evaluation with hand witten digit classification

To explore the efficiency of NASH, we evaluate its performance by classifying Modified National Institute of Standards and Technology (MNIST) dataset [108] using SNN of 784:225:10 that was trained offline, and 784:100, trained on-chip. The MNIST benchmark was chosen for this evaluation because of its wide use, providing a basis for comparison with existing works. The MNIST images were converted to spikes with Poisson neural coding.

In carrying out the evaluation, we perform two experiments. The first experiment evaluates the classification accuracy and average classification time (ACT) on both NASH and the baseline architecture using the SP-KMCR, the XYZ-UB and the XY-UB algorithms without faults, over various spike arrival windows (SAWs). The ACT is the average time taken to classify one MNIST image, and the SAW is the time duration given for all flits (spikes) from source SNPCs to arrive at a destination SNPC. The countdown of SAW begins after the first flit arrives, and any flit that arrives after it reaches zero is not decoded. After all flits that arrived within the SAW have been decoded, the SAW is reset. For the second experiment, we evaluate the accuracy and ACT on both NASH and the baseline architecture using the FTSP-KMCR algorithm over various fault rates.

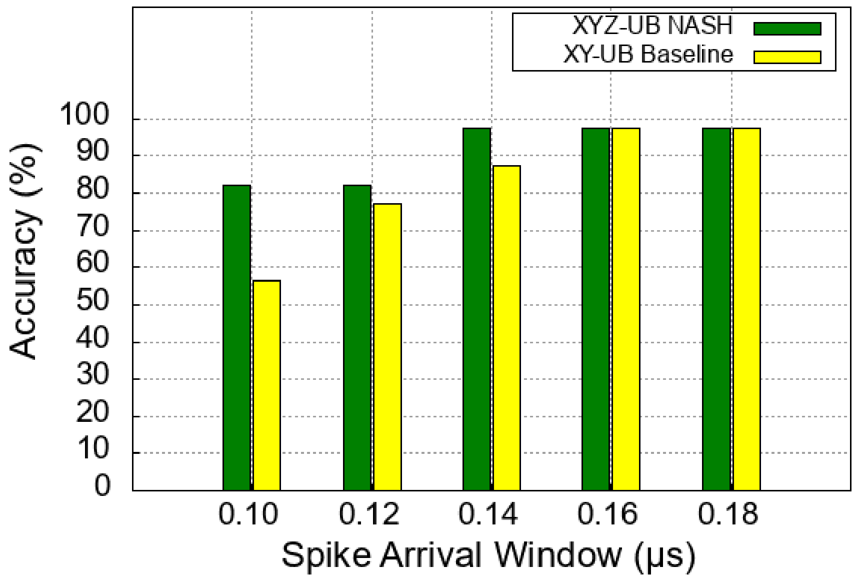


Figure 6.12: MNIST classification accuracy over various SAW on NASH and baseline architecture using the XYZ-UB and XY-UB algorithms.

In the first experiment, the accuracy over various SAWs are presented for the XYZ-UB in Fig. 6.12 and the SP-KMCR in Fig. 6.14. For the XYZ-UB, NASH at SAWs 0.1, 0.12, and 0.14, show 25.2%, 5.04%, and 10.3% better accuracy respectively, than the baseline architecture. Also, for the SP-KMCR, NASH at SAWs 0.1, 0.12, and 0.14, show 25.6%, 20.5%, and 10.2% better accuracy respectively, than the baseline architecture. This difference in accuracy is because more spikes arrived before the end of the SAWs on NASH, enabling more spikes to be processed, which resulted in better accuracy. At SAW 0.16, the accuracy on both algorithms reach 97.6% and saturates. This is because, at this SAW, all spikes for both architecture arrive, and further increasing the SAW as seen at SAW 0.18 does not cause any change in the accuracy.

For the XYZ-UB in Fig. 6.13, the ACT of the baseline architecture at SAWs 0.1, 0.12,

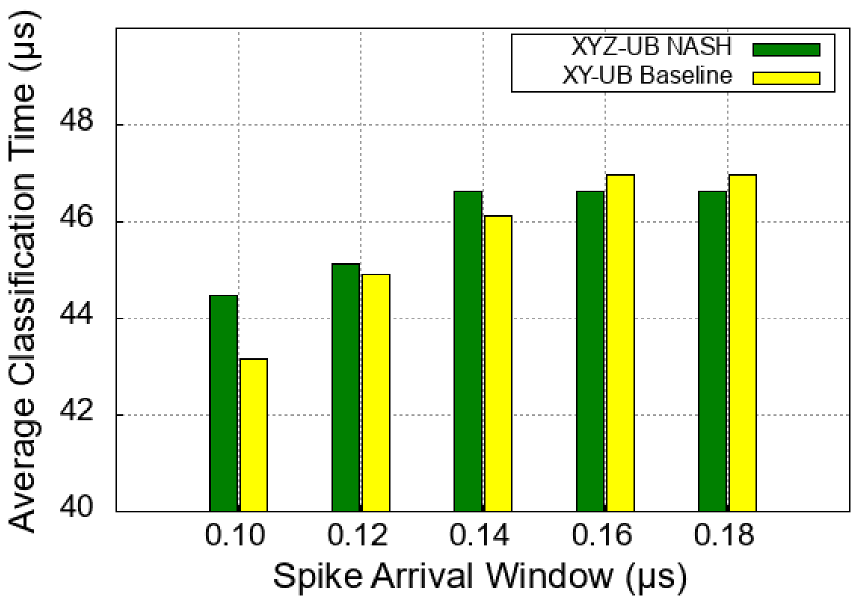


Figure 6.13: MNIST average classification time over various SA. with the XYZ-UB and XY-UB algorithms.

and 0.13 despite having lower accuracy, are 2.9%, 0.6%, and 1.2% respectively lower than that of NASH. For the SP-KMCR in Fig. 6.15, the ACT of the baseline architecture at SAWs 0.1 and 0.12 with also lower accuracy, are 3.2% and 2.3% respectively lower than that of NASH. This can be attributed to the time taken by the destination SNPC on NASH to process the increased number of spikes that arrived. However, for the XYZ-UB, at the accuracy saturation point of SAW 1.6 when all spikes arrive for NASH and the baseline architecture, NASH shows 1.1% lower ACT than the baseline architecture. For the SP-KMCR, even though NASH had reached an accuracy saturation point at SAW 0.14 and had to process more spikes than the baseline architecture that had not, it still shows 0.4% lower ACT. At SAW 1.6 when both NASH and the baseline architecture had reached saturation, NASH shows an even lower ACT of 2.5% than the baseline architecture.

For the second experiment, the accuracy and ACT of NASH and the baseline architecture over various fault rates using the FTSP-KMCR algorithm are evaluated. A SAW of 0.2 is chosen for this experimentbecause both NASH and the baseline architecture have reached accuracy saturation, giving enough time for the changes in the ACT to be

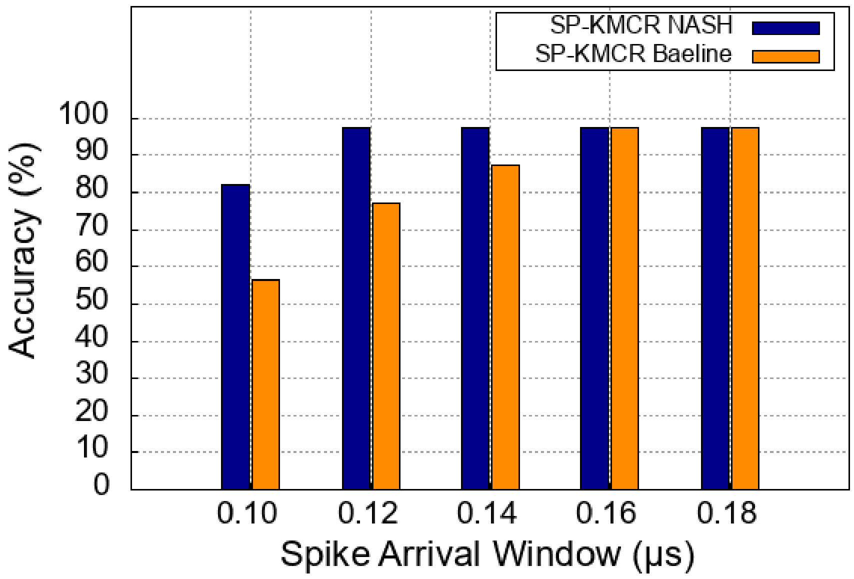


Figure 6.14: MNIST classification accuracy over various SAW with the SP-KMCR algorithms.

monitored when fault rate is varied. As described in Fig. 6.16, NASH and the baseline architecture both maintain the saturation accuracy of 97.6% from 0 to 5% fault rate. However, at 10% fault, the accuracy of the baseline architecture drops to 90%, and further reduces to 55.77% and 41.05% as the fault rate reaches 20% and 30% respectively. NASH on the other hand, maintains the saturation accuracy all through. The differences in accuracy between NASH and the baseline architecture is because a NASH node has higher path diversity than the baseline architecture, so in the event of faulty links, NASH has more links that can be utilized as backup compared to the baseline architecture, therefore delivering more spikes to the destination within the SAW than the baseline architecture.

In Fig. 6.17 the ACT for NASH and the baseline architecture starts at 45.96 and 46.98 microseconds respectively with zero fault rate, and slightly increases by 2% as fault rate reaches 5%. However, as the fault rate further increases, the ACT of NASH increases, while that of the baseline decreases. At 30% fault rate, NASH utilizes 40% more ACT compared to the baseline architecture. The increase in ACT is due to the time taken to process the increased number of spikes that arrived within the SAW, which led to better accuracy in Fig. 6.16. While the decrease in ACT for the baseline is because fewer spikes

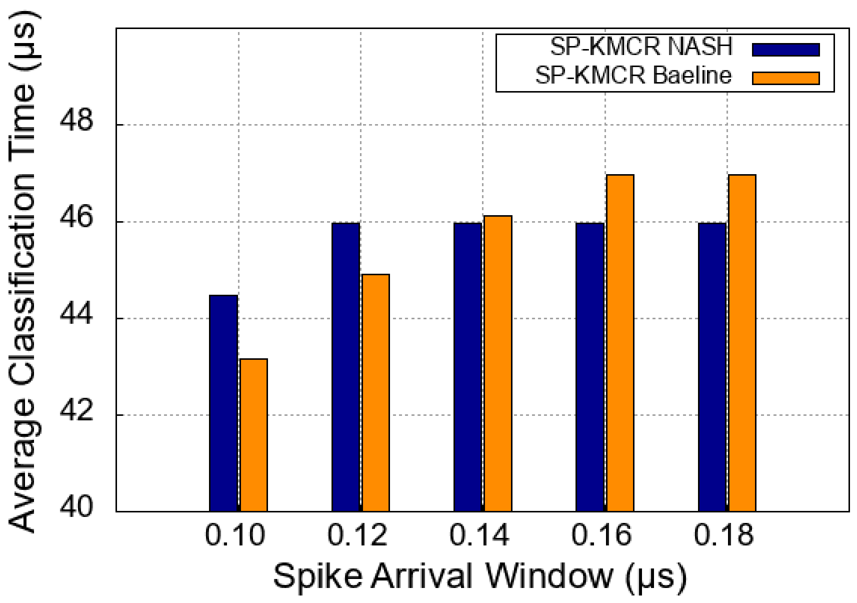


Figure 6.15: Average classification time of MNIST image over various SAW with the XYZ-UB and XY-UB algorithms.

arrive within the SAW, reducing the number of spikes processed by the its destination SNPCs, which led to lower accuracy, as shown in Fig. 6.16.

6.4.3 Hardware Complexity Analysis

In this subsection, we evaluate and analyze NASH node's hardware complexity, and then compare with the baseline node using the XY-UB, XYZ-UB, SPKMCR, and FTSP-KMCR algorithms.

Table 6.1: Hardware Complexity Comparison of NASH and the Baseline Nodes.

architecture	XY-UB	XYZ-UB	SP-KMCR		FTSP-KMCR	
Architecture	Baseline	NASH	Baseline	NASH	Baseline	NASH
Area (mm^2)	1.312	1.316	1.316	1.322	1.320	1.325
Power (mW)	66.16	66.63	66.50	66.84	68.22	70.10

As described in Table 6.1 and Fig. 6.21, an FTSP-KMCR NASH node occupies a silicon area of 1.325mm^2 excluding pads. Fig. 6.18 shows that the synapse crossbar and SRAM-based synapse memory occupy a significant portion of the chip area at 90.3%. This is because of the considerable amount of stored synapses. All 256 neurons occupy 1.3% of the node area, the STDP learning module occupy 7.8%, and the network interface and

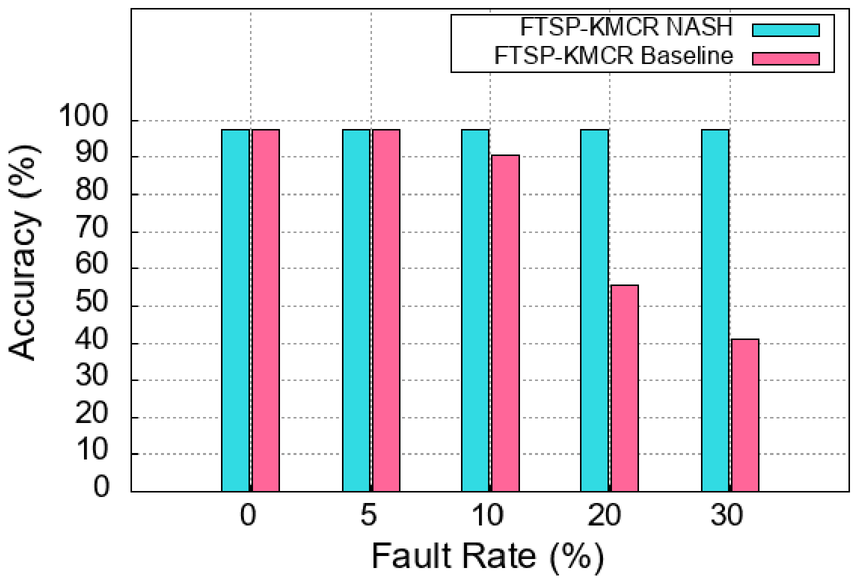


Figure 6.16: MNIST classification accuracy over various fault rates using the FTSP-KMCR.

router occupy 0.6%. At 1.1V, 25°C and a clock frequency of 142MHz, an FTSP-KMCR NASH node consumes 70.10mW. Assuming that all 256 neurons spike at the same rate, each spike event results to 256 synaptic operations in one cycle. With cortical fast spiking neurons able to generate spikes at rates up 582Hz, the corresponding rate of synaptic operation in NASH is 37M synaptic operations per second. Therefore, a clock frequency of 142MHz is sufficient to operate at biological real time. Because of the substantial amount of memory access required for moving synapse values during learning and classification, most of the energy consumed by NASH can be attributed to memory access.

A comparison of NASH and the baseline architecture of XY-UB, XYZ-UB, SP-KMCR, and FTSP-KMCR algorithms presented in Table 6.1 show that the NASH node of these algorithms occupy larger footprint and has higher power consumption when compared to the baseline architecture nodes. This is due to NASH's increased design complexity and its higher degree of path diversity enabled by TSVs, whose diameter also add to the footprint. Figure 6.20 evaluates how changes in synapse precision affect both the hardware complexity and the performance of NASH. With increased precision, the area, power and performance increases, and reduces otherwise. The energy per synaptic operation (SOP)

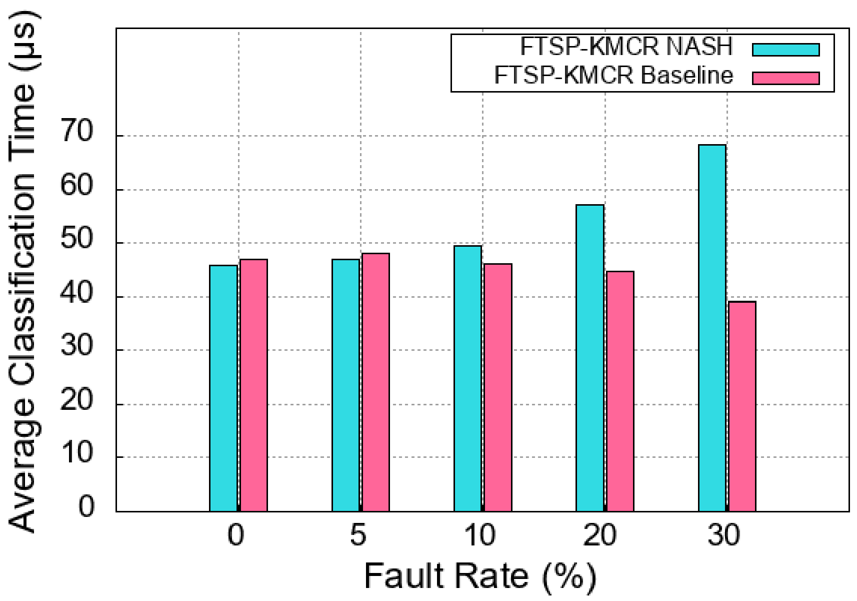


Figure 6.17: Average classification time of MNIST image over various fault rates with theFTSP-KMCR algorthm.

of NASH is a division of the power consumption of the SNPC and the rate of synaptic operation. NASH performs a maximum of 256 operations in one cycle, except for the STDP synapse update, which takes two cycles.

6.5 Conclusion

In summary, this chapter presented the architecture, hardware design, and evaluation of an adaptive scalable 3D-NoC-based neuromorphic system with on-chip learning, named NASH. The proposed system leverages the high scalability and parallelism, low communication cost, and high throughput available in 3D-NoC to present a neuromorphic architecture capable of supporting large SNN with massive number of synapses. To handle challenges that may arise in spike communication and lead to performance degradation, we employed the FTSP-KMCR routing algorithm. We presented the network and performance evaluation of NASH with and without faults. From the evaluation results, we found that NASH achieved an accuracy of 79.4% and 97.6% on MNIST data set classification with on-chip and off-chip learning, respectively. Moreover, the experiments show that the proposed NASH achieves better accuracy with less ACT when compared to the

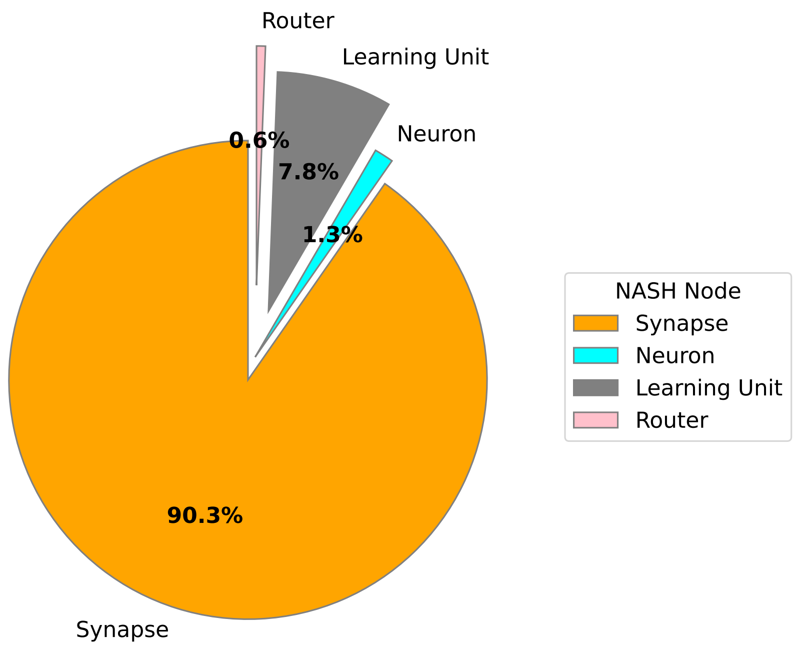


Figure 6.18: Area analysis of a NASH node.

baseline architecture and sustains a 97.6% accuracy with up to 30% fault rate while experiencing a 40% increase in the ACT as opposed to the baseline architecture. In the next chapter, we present the conclution of this thesis, and describe directions that will guide our future work.

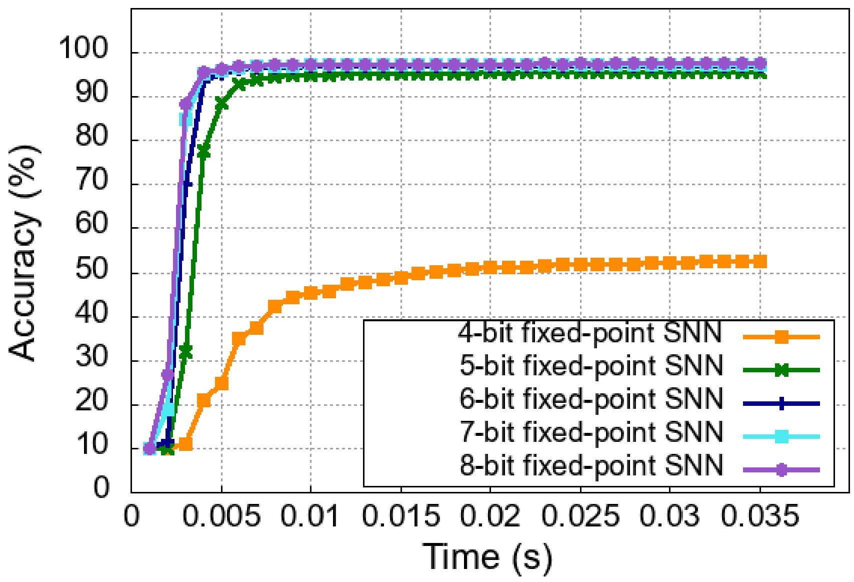


Figure 6.19: Accuracy evaluation over various synapse precision

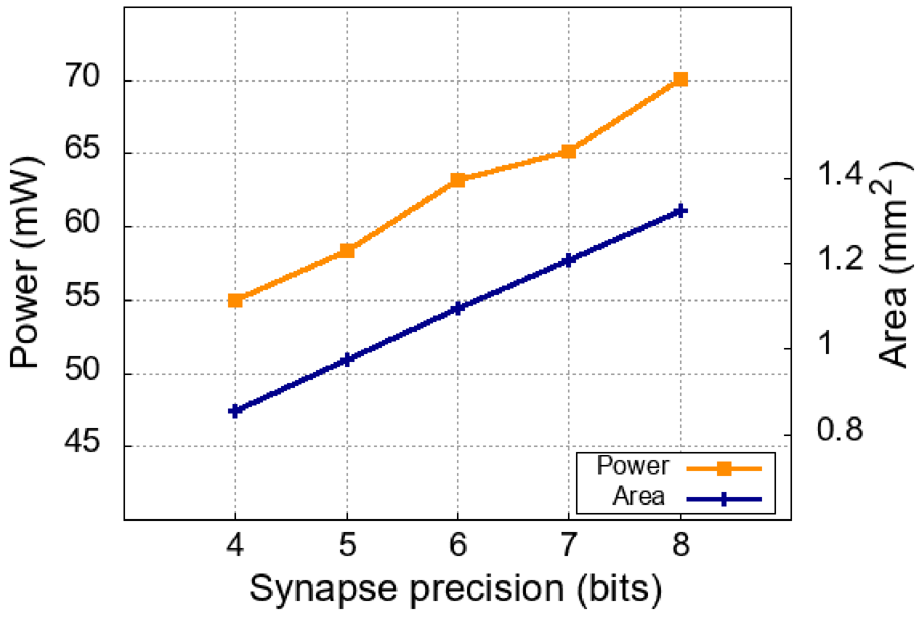


Figure 6.20: Hardware complexity over various synapse precision.

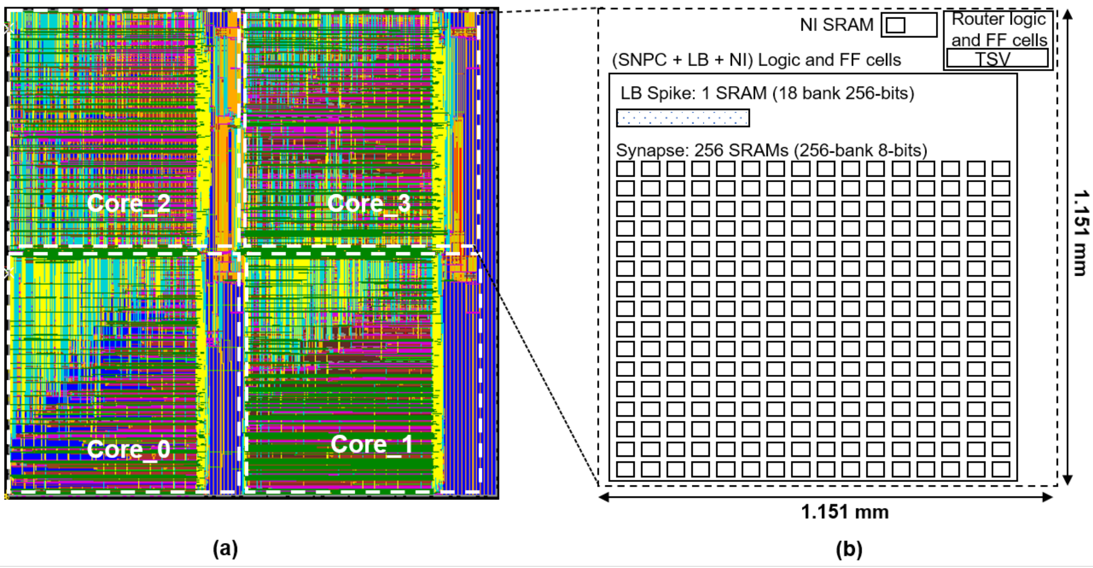

Core_2
Core_3
Core_0
Core_1
(a)
NI SRAM
Router logic and FF cells
TSV
(SNPC + LB + NI) Logic and FF cells
LB Spike: 1 SRAM (18 bank 256-bits)
Synapse: 256 SRAMs (256-bank 8-bits)
1.151 mm
1.151 mm
(b)

Figure 6.21: Layout of a 2 × 2 NASH system.

7 Conclusion and Future Works

In this dissertation, we presented an adaptive scalable 3D-NoC-based neuromorphic system (NASH). NASH shows essential characteristics, such as low latency, high throughput, high reliability, and low footprint which make it suitable for large-scale SNN-based embedded artificial intelligence (AI) implementations.

Prior to presenting the NASH design, we presented the architecture of a light-weight spiking neuron processing core (SNPC) with on-chip learning, which is the backbone of NASH. Embedding 256 physical neuron and 65K synapses in 256 distributed SRAMs of 256-bank, the SNPC is able to perform parrallel update of neurons and synapses using the PNU and PWU mechanisms. The SNPC was evaluated by classifying MNIST 16×16 images with off-chip and on-chip learning. From the evalution, the SNPC was able to achieve an accuracy of 96.71 and 72.9% with off-chip and on-chip learning respectively.

NASH then integrates several SNPCs in a 3D-NoC communication architecture which enable it to provide high scalability, parallelism, low communication cost, and high throughput. To handle challenges that may arise in spike communication and lead to performance degradation, NASH enables adaptivity, by adoopting the FTSP-KMCR routing algorithm. The evaluation result from classifying 28×28 MNIST images on NASH achieved an accuracy of 79.4% and 97.6% for on-chip and off-chip learning, respectively. Moreover, the experiments show that the a better accuracy was achieved with less ACT when compared to the baseline processor, and the accuracy of 97.6% was sustained with up to 30% fault rate while experiencing a 35% increase in the ACT as opposed to the baseline system.

It is important to note that there are several other issues that need to be addressed in the NASH system. One of these issues is mapping. In this dissertation, we used the layer based mapping approach. Howwver a comprehensive mapping technique needs to be ivestigated because mapping influences the overall performance of the neuron communication architecture. Furthermore, other forms of adaptivity need to be investigated. This research focused only on permanent faults that may occur in the communication links among neurons, however; other parts such as neurons and memory, need to be considered. Another issue that needs to be addressed is application. The evaluation of NASH was done by classifying MNIST dataset, but while MNIST is a popular dataset for evaluating machine learning systems, It does not fullly exploit the advantages of NASH. Therefore applications (eg. biological) that will fully harness the advantages of NASH for improved performance and energy efficiency need to performed. These issues in all, provide a roadmap that will guide future works.

List of Publications

Major Journals

1. **Ogbodo Mark Ikechukwu**, Khanh N. Dang and Abderazek Ben. Abdallah, ”On the Design of a Fault-Tolerant Scalable Three Dimensional NoC-Based Digital Neuromorphic System With On-Chip Learning,” in IEEE Access, vol. 9, pp. 64331-64345, 2021, doi: 10.1109/ACCESS.2021.3071089.

2. Zhishang Wang, **Ogbodo Mark Ikechukwu**, Huakun Huang, Chen Qiu, Masayuki Hisada, Abderazek Ben Abdallah, “AEBIS: AI-Enabled Blockchain-Based Electric Vehicle Integration System for Power Management in Smart Grid Platform,” in IEEE Access, vol. 8, pp. 226409-226421, 2020, doi: 10.1109/ACCESS.2020.3044612.

3. The H. Vu, **Ogbodo Mark Ikechukwu**, and Abderazek Ben Abdallah, Fault-tolerant Spike Routing Algorithm and Architecture for Three Dimensional NoC-Based Neuromorphic Sysstems, IEEE Access, vol. 7, pp. 90436-90452, 2019.

Major conferences

1. **Ogbodo Mark Ikechukwu**, Khanh N. Dang and Abderazek Ben. Abdallah, Energy-efficient Spike-based Scalable Architecture for Next-generation Cognitive AI Computing Systems, Ubiquitous Networking. UNet 2021. Lecture Notes in Computer

Science, vol 12845. Springer, Cham. https://doi.org/10.1007/978-3-030-86356-2_19, *(Best student paper award)*.

2. **Ogbodo Mark Ikechukwu**, The H. Vu, Khanh N. Dang and Abderazek Ben Abdallah, Light-Weight Spiking Neuron Processing Core for Large-Scale 3D-NoC Based Spiking Neural Network Processing Systems, 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), 2020, pp. 133-139.

Refereed conferences

3. **Ogbodo Mark Ikechukwu**, Khanh N. Dang, Tomohide Fukuchi and Ben Abdallah, Abderazek. (2020). Architecture and Design of a Spiking Neuron Processor Core Towards the Design of a Large-scale Event-Driven 3D-NoC-based Neuromorphic Processor. SHS Web of Conferences. 77. 04003.

4. Huakun Huang, **Ogbodo Mark Ikechukwu**, Zhisheng Wang, Chen Qiu, Masayuki Hisada, Abderazek Ben Abdallah, ”Smart Energy Management System based on Reconfigurable AI Chip and Electrical Vehicles”, 2021 IEEE International Conference on Big Data and Smart Computing (BigComp 2021), January 17-20, 2021, Jeju Island, Korea

5. Okada Yuuki, Jiangkun Wang, **Ogbodo Mark Ikechukwu**, Abderazek Ben Abdallah, ”Hardware Acceleration of Convolution Neural Network for AI-Enabled Realtime Biomedical System,” 3rd ETLTC2021-ACM Chapter Int. Conference on Information and Comm. Technology, January 27-30, 2021, Aizu-Wakamatsu, Japan

6. Tomohide Fukuchi, **Ogbodo Mark Ikechukwu**, Abderazek Ben Abdallah. Design and Optimization of a Deep Neural Network Architecture for Traffic Light Detection, ACM Chapter International Conference on Educational Technology, Lan-

guage and Technical Communication (ETLTC), January 27-31, 2020, Aizuwakamatsu, Japan

References

[1] R. Courtland, "Transistors could stop shrinking in 2021," *IEEE Spectrum*, vol. 53, pp. 9–11, Jul. 2016.

[2] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[3] Macrovector, "Cartoon human body organs composition," https://www.freepik.com/vectors/frame', (Accessed on 2022/01/05).

[4] V. H. The, "Algorithms and architectures for spiking neuromorphic systems," Ph.D. dissertation, University of Aizu, Sep. 2019.

[5] W. Gerstner and W. K. M., *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Jun. 2002.

[6] K. Mark, "Github - mark-kramer/case-studies-python: An introduction to the practicing neuroscientist to data analysis in python," https://github.com/Mark-Kramer/Case-Studies-Python, (Accessed on 02/05/2022).

[7] I. M. Eugene, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[8] D. N. Khanh, "Development of on-chip communication fault-resilient adaptive architectures and algorithms for 3d-ic technologies," Ph.D. dissertation, University of Aizu, Sep. 2017.

[9] B.-A. Akram, "High-throughput architecture, routing algorithms, reliable mesh-based, many-core, network-on-chip systems," Ph.D. dissertation, University of Aizu, Mar. 2015.

[10] B.-A. Akram and B.-A. Abderazek, "Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3d-noc systems," *Journal of Parallel and Distributed Computing*, vol. 93-94, pp. 30–43, Jul. 2016.

[11] B. Marek, L. Fujcik, and R. Vrba, "Field programmable neural array for feed-forward neural networks," in *2013 36th International Conference on Telecommunications and Signal Processing (TSP)*, Jul. 2013, pp. 727–731.

[12] S. A. McKee, "Reflections on the memory wall," in *Proceedings of the 1st Conference on Computing Frontiers*. ACM, Apr. 2004, pp. 162–167.

[13] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng, "Building high-level features using large scale unsupervised learning," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. USA: Omnipress, Jul. 2012, pp. 507–514.

[14] J. Rodrigues-de Oliveira-Neto, J. P. Cerquinho-Cajueiro, and J. Ranhel, "Neural encoding and spike generation for spiking neural networks implemented in fpga," in *2015 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, Feb 2015, pp. 55–61.

[15] N. Fourcaud-Trocmé, *Encyclopedia of Computational Neuroscience: Integrate and Fire Models, Deterministic*. New York, NY: Springer New York, May 2014, pp. 1–9.

[16] B. F. Mark, C. W. Barry, and P. A. Michael, *Neuroscience: Exploring the Brain, 4th Edition*. Lippincott Williams and Wilkins, 351 W Camden St, Baltimore, MD 21201, United States: Lippincott Williams and Wilkins, Apr. 2016, pp. 81–108.

[17] A. Hodgkin and A. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, Aug. 1952.

[18] M. Henry, "The blue brain project," *Nature Reviews Neuroscience*, vol. 2, no. 7, pp. 153–159, Feb. 2006.

[19] A. David and L. B. Simon, "An energy budget for signaling in the grey matter of the brain," *Journal of Cerebral Blood Flow & Metabolism*, vol. 21, no. 10, pp. 1133–1145, Oct 2001.

[20] "Sparsity enables 100x performance acceleration in deep learning networks," White Paper, Numenta, May 2021.

[21] M. Wolfgang, "On the relevance of time in neural computation and learning," in *Proceedings of the 8th International Conference on Algorithmic Learning Theory*. London, UK, UK: Springer-Verlag, 1997, pp. 364–384.

[22] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in Neuroscience*, vol. 13, p. 95, Mar. 2019.

[23] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," Oct. 2015.

[24] B.-A. Abderazek, *3D Integration Technology for Multicore Systems On-Chip*. Singapore: Springer Singapore, Sep. 2017, pp. 175–199.

[25] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, Aug. 2017.

[26] A. P. Johnson, J. Liu, A. G. Millard, S. Karim, A. M. Tyrrell, J. Harkin, J. Timmis, L. McDaid, and D. M. Halliday, "Fault-tolerant learning in spiking astrocyte-neural networks on fpgas," in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, Jan. 2018, pp. 49–54.

[27] F. Jean-Marc, "Discovering spike patterns in neuronal responses," *Journal of Neuroscience*, vol. 24, no. 12, pp. 2989–3001, Mar. 2004.

[28] D. S. Broomhead and L. David, "Radial basis functions, multi-variable functional interpolation and adaptive networks," *Royal Signals and Radar Establishment Malvern (United Kingdom)*, vol. RSRE-MEMO-4148, Mar. 1988.

[29] J. J. Hopfield, *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, Apr. 1988, ch. Neural Networks and Physical Systems with Emergent Collective Computational Abilities, pp. 457–464.

[30] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, Jun. 2021.

[31] E. L. Jeffrey, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Apr. 1990.

[32] V. H. The, R. Murakami, Y. Okuyama, and B.-A. Abderazek, "Efficient optimization and hardware acceleration of cnns towards the design of a scalable neuro inspired architecture in hardware," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jan. 2018, pp. 326–332.

[33] Y. Berg, R. L. Sigvartsen, T. S. Lande, and A. Abusland, "An analog feed-forward neural network with on-chip learning," *Analog Integrated Circuits and Signal Processing*, vol. 9, no. 1, pp. 65–75, Jan. 1996.

[34] G. Carvajal, M. Figueroa, D. Sbarbaro, and W. Valenzuela, "Analysis and compensation of the effects of analog vlsi arithmetic on the lms algorithm," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1046–1060, Jul. 2011.

[35] H. Tamukoh and M. Sekine, "A dynamically reconfigurable platform for self-organizing neural network hardware," in *Neural Information Processing. Models and Applications*, K. W. Wong, B. S. U. Mendis, and A. Bouzerdoum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Sep. 2010, pp. 439–446.

[36] A. I. Oludare, J. Aman, A. E. Omolara, K. D. V., A. M. Nachaat, and A. Humaira, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, Nov. 2018.

[37] I. M. Eugene, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[38] S. S. Harris, "Neural coding in primary motor cortex," in *Encyclopedia of Neuroscience*, S. R. Larry, Ed. Oxford: Academic Press, Jan. 2010, pp. 105–115.

[39] W. Gerstner, *Neuronal dynamics : from single neurons to networks and models of cognition*. Cambridge, United Kingdom: Cambridge University Press, Sep 2014.

[40] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems," *Frontiers in Neuroscience*, vol. 15, Mar. 2021.

[41] J. O'Keefe, "Hippocampus, theta, and spatial memory," *Current Opinion in Neurobiology*, vol. 3, no. 6, pp. 917 – 924, Dec. 1993.

[42] S. Thorpe and J. Gautrais, "Rank order coding," in *Computational Neuroscience*. Springer US, 1998, pp. 113–118. [Online]. Available: https://doi.org/10.1007/978-1-4615-4831-719

[43] B. Chen and K. Y. Kwan, Eds., *Neural circuit and cognitive development*, 2nd ed. San Diego, CA: Academic Press, Jun. 2020.

[44] N. B. Larry, Abbott F.and Sacha, "Synaptic plasticity: taming the beast," *Nature Neuroscience*, vol. 3, no. S11, pp. 1178–1183, Nov. 2000.

[45] C. Frenkel, J.-D. Legat, and D. Bol, "Morphic: A 65-nm 738k-synapse/mm 2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, pp. 999–1010, Oct. 2019.

[46] N. Zheng and P. Mazumder, "Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 9, pp. 4287–4302, Nov. 2018.

[47] S. Bohte, J. Kok, and J. Poutré, "Spikeprop: backpropagation for networks of spiking neurons." in *ESANN 2000, 8th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 26-28, 2000, Proceedings*, vol. 48, Jan. 2000, pp. 419–424.

[48] R. Gütig and H. Sompolinsky, "Tempotron learning," in *Encyclopedia of Computational Neuroscience*. Springer New York, 2014, pp. 1–3.

[49] S. B. M., J. K. N., and H. L. Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, Oct. 2002.

[50] B.-A. Abderazek and D. N. Khanh, *Neuromorphic computing principles and organization*, 1st ed. New York, United States: Springer Nature, Mar. 2022.

[51] T. Zhang, W. Zhou, X. Jiang, and Y. Liu, "Fpga-based implementation of hand gesture recognition using convolutional neural network," in *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, Oct. 2018, pp. 133–138.

[52] Y. Fan and P. Michael, "Implementation of an rbf neural network on embedded systems: real-time face tracking and identity verification," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1162–1175, Sep. 2003.

[53] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[54] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, Dec. 2013.

[55] Holler, Tam, Castro, and Benson, "An electrically trainable artificial neural network (etann) with 10240 'floating gate' synapses," in *International 1989 Joint Conference on Neural Networks*, vol. 2, Aug. 1989, pp. 191–196.

[56] J. Liu, M. A. Brooke, and K. Hirotsu, "A cmos feedforward neural-network chip with on-chip parallel learning for oscillation cancellation," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1178–1186, Sep. 2002.

[57] D. U. Peter, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.

[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 *2011 IEEE Custom Integrated Circuits Conference (CICC)*, Sep. 2011, pp. 1–4.

[59] A. Bhaskar, "Design and analysis of low power sram cells," in *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, Apr. 2017, pp. 1–5.

[60] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2016, pp. 380–392.

[61] A. F. Vincent, J. Larroque, N. Locatelli, N. Ben Romdhane, O. Bichler, C. Gamrat, W. S. Zhao, J.-O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 2, pp. 166–174, Apr. 2015.

[62] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *2011 International Electron Devices Meeting*, Dec. 2011, pp. 4.4.1–4.4.4.

[63] T. Luo, X. Wang, C. Qu, M. K. F. Lee, W. T. Tang, W.-F. Wong, and R. S. M. Goh, "An fpga-based hardware emulator for neuromorphic chip with rram," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 2, pp. 438–450, Feb. 2020.

[64] S. Yu, Y. Wu, and P. Wong, "Investigating the switching dynamics and multi-level capability of bipolar metal oxide resistive switching memory," *Applied Physics Letters*, vol. 98, no. 10, p. 103514, Feb. 2011.

[65] C.-Y. Chen, H.-C. Shih, C.-W. Wu, C.-H. Lin, P.-F. Chiu, S.-S. Sheu, and F. T. Chen, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, Jan. 2015.

[66] M. Glesner and W. Pochmuller, *Neurocomputers: An overview of Neural Networks in VLSI*, 1st ed. Andover, England: Chapman and Hall, London, Sep. 1994.

[67] P. Ienne, "Digital hardware architectures for neural networks," *n SPEEDUP Journal*, vol. 9, no. 1, pp. 18–25, 1995.

[68] D. Yiping and W. Takahiro, "High performance noc architecture for two hidden layers bp neural network," in *2008 International SoC Design Conference*, vol. 01, Nov 2008, pp. I–269–I–272.

[69] M. A. Sivilotti, "Wiring considerations in analog vlsi systems, with application to field-programmable networks," Ph.D. dissertation, California Institute of Technology, Apr. 1991.

[70] J. Park, T. Yu, C. Maier, S. Joshi, and G. Cauwenberghs, "Live demonstration: Hierarchical address-event routing architecture for reconfigurable large scale neuromorphic systems," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 707–711.

[71] L. John and W. John, "A multi-sender asynchronous extension to the aer protocol," in *Proceedings Sixteenth Conference on Advanced Research in VLSI*, Mar. 1995, pp. 158–169.

[72] F. Steve, "Large-scale neuromorphic computing systems," *Journal of Neural Engineering*, vol. 13, no. 5, p. 051001, Aug. 2016.

[73] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neurophic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[74] B. Belhadj, A. Valentian, P. Vivet, M. Duranton, L. He, and O. Temam, "The improbable but highly appropriate marriage of 3d stacking and neuromorphic accelerators," in *2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, Oct. 2014, pp. 1–9.

[75] S. Yang, J. Wang, B. Deng, C. Liu, H. Li, C. Fietkiewicz, and K. A. Loparo, "Real-time neuromorphic system for large-scale conductance-based spiking neural networks," *IEEE Transactions on Cybernetics*, vol. 49, no. 7, pp. 2490–2503, Jul. 2019.

[76] H. An, M. S. Al-Mamun, M. K. Orlowski, and Y. Yi, "A three-dimensional (3d) memristive spiking neural network (m-snn) system," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, May 2021, pp. 337–342.

[77] E. M. Amimul, Z. Zhen, and Y. Yang, "Modeling and analysis of neuronal membrane electrical activities in 3d neuromorphic computing system," in *2017 IEEE International Symposium on Electromagnetic Compatibility Signal/Power Integrity (EMCSI)*, Aug. 2017, pp. 745–750.

[78] S. Yang, B. Deng, J. Wang, H. Li, M. Lu, Y. Che, X. Wei, and K. A. Loparo, "Scalable digital neuromorphic architecture for large-scale biophysically meaningful neural network with multi-compartment neurons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 1, pp. 148–162, Mar 2020.

[79] V. H. The, M. I. Ogbodo, and A. Ben-Abdallah, "Fault-tolerant spike routing algorithm and architecture for three dimensional NoC-based neuromorphic systems," *IEEE Access*, vol. 7, pp. 90 436–90 452, Jun. 2019.

[80] M. Alessandro, E. Vittoz, and P. Venier, "A communication scheme for analog vlsi perceptive systems," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 6, pp. 660–669, Jun. 1995.

[81] J. Schemmel, A. Grübl, S. Hartmann, A. Kononov, C. Mayr, K. Meier, S. Millner, J. Partzsch, S. Schiefer, S. Scholze, R. Schüffny, and M.-O. Schwartz, "Live demonstration: A scaled-down version of the brainscales wafer-scale neuromorphic system," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 702–702.

[82] B.-V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, May. 2014.

[83] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in Neuroscience*, vol. 9, pp. 39–55, Apr. 2015.

[84] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, Aug. 2017.

[85] C. S. H. and R. D. Clay, "Fault tolerance in artificial neural networks," in *1990 IJCNN International Joint Conference on Neural Networks*, vol. 1, Jun. 1990, pp. 703–708.

[86] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

[87] W. Naihong, Y. Shiyuan, and T. Shibai, "A modified learning algorithm for improving the fault tolerance of bp networks," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 1, Jun. 1996, pp. 247–252.

[88] A. Hashmi, H. Berry, O. Temam, and M. Lipasti, "Automatic abstraction and fault tolerance in cortical microachitectures," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2011, pp. 1–10.

[89] J. Deng, Y. Fang, Z. Du, Y. Wang, H. Li, O. Temam, P. Ienne, D. Novo, X. Li, Y. Chen, and C. Wu, "Retraining-based timing error mitigation for hardware neural networks," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2015, pp. 593–596.

[90] L. Chu and B. W. W., "Fault tolerant neural networks with hybrid redundancy," in *1990 IJCNN International Joint Conference on Neural Networks*, vol. 2, Jun. 1990, pp. 639–649.

[91] E. D. Martin and D. Robert, "Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 788–793, Sep. 1993.

[92] K. C., V. Kanonkluk, and L. Chidchanok, "Weight shifting techniques for self-recovery neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 651–658, Jul. 1994.

[93] J. Liu, J. Harkin, L. P. Maguire, L. J. McDaid, and J. J. Wade, "Spanner: A self-repairing spiking neural network hardware architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 1287–1300, Apr. 2018.

[94] J. Wu and S. Furber, "A multicast routing scheme for a universal spiking neural network architecture," *The Computer Journal*, vol. 53, no. 3, pp. 280–288, Apr. 2009.

[95] P. Stephen, "Robustness of feedforward neural networks," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 2, Jun. 1992, pp. 346–351 vol.2.

[96] C. Ching-Tai, M. G. Kishan, M. K. Chilukuri, and R. S., "Training techniques to obtain fault-tolerant neural networks," in *Proceedings of IEEE 24th International Symposium on Fault-Tolerant Computing*, Jun. 1994, pp. 360–369.

[97] M. Naeem, L. J. McDaid, J. Harkin, J. J. Wade, and J. Marsland, "On the role of astroglial syncytia in self-repairing spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2370–2380, Oct. 2015.

[98] B. Maxence, V. Alexandre, M. Thomas, R. Francois, R. Marina, V. Elisa, and B. Edith, "Spiking neural networks hardware implementations and challenges," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 2, pp. 1–35, Jun. 2019.

[99] Hananel-Hazan, "BindsNet: A Machine Learning-O|riented Spiking Neural Networks Library in Python," Apr. 2021. [Online]. Available: https://github.com/BindsNET/bindsnet

[100] O. Mark, D. H. Khan, T. Fukuchi, and B.-A. Abderazek, "Architecture and design of a spiking neuron processor core towards the design of a large-scale event-driven 3d-NoC-based neuromorphic processor," in *SHS Web of Conferences*, D. Roy, Ed., vol. 77. EDP Sciences, Jan. 2020, p. 04003.

[101] O. I. Mark, D. N. Khanh, and B.-A. Abderazek, "Energy-efficient spike-based scalable architecture for next-generation cognitive ai computing systems," in *Ubiquitous Networking*, H. Elbiaze, E. Sabir, F. Falcone, M. Sadik, S. Lasaulce, and J. Ben Othman, Eds. Cham: Springer International Publishing, Dec. 2021, pp. 225–238.

[102] O. I. Mark, V. H. The, K. D. N., and B.-A. Abderazek, "Light-weight spiking neuron processing core for large-scale 3D-NoC based spiking neural network processing systems," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Apr. 2020, pp. 133–139.

[103] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SAÏGHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, pp. 697–720, May 2011.

[104] M. Rahimi Azghadi, N. Iannella, S. F. Al-Sarawi, G. Indiveri, and D. Abbott, "Spike-based synaptic plasticity in silicon: Design, implementation, application, and challenges," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 717–737, May 2014.

[105] NanGate Inc., "Nangate Open Cell Library 45 nm," http://www.nangate.com/, (accessed 14.02.2022).

[106] M. R. Guthaus, J. E. Stine, S. Ataei, Brian Chen, Bin Wu, and M. Sarwar, "Openram: An open-source memory compiler," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, vol. 34, no. 2, Nov. 2016, pp. 1–6.

[107] NCSU Electronic Design Automation, "FreePDK3D45 3D-IC process design kit," http://www.eda.ncsu.edu/wiki/FreePDK3D45:Contents, (accessed 13.02.2022).

[108] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," http://yann.lecun.com/exdb/mnist/, (accessed 13.02.2022).

[109] A. Asohan, "Spike count reliability and the poisson hypothesis," *Journal of Neuroscience*, vol. 26, no. 3, pp. 801–809, Jan. 2006.

[110] D. U. Peter and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, Aug 2015.

[111] B. S. Danielle and B. T. Edward, "Small-world brain networks revisited," *The Neuroscientist*, vol. 23, no. 5, pp. 499–516, Sep. 2016.

[112] S. Shibata, Y. Komaki, F. Seki, M. Inouye, T. Nagai, and H. Okano, "Connectomics: Comprehensive approaches for whole-brain mapping," *Microscopy*, vol. 64, pp. 57–67, Feb. 2014.

[113] O. I. Mark, D. N. Khanh, and B.-A. Abderazek, "On the design of a fault-tolerant scalable three dimensional noc-based digital neuromorphic system with on-chip learning," *IEEE Access*, vol. 9, no. 1, pp. 64 331–64 345, Apr. 2021.

[114] B.-A. Abderazek and D. N. Khanh, "Toward robust cognitive 3d brain-inspired cross-paradigm system," *Frontiers in Neuroscience*, vol. 15, Jun. 2021.

[115] D. Vainbrand and R. Ginosar, "Network-on-chip architectures for neural networks," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, May. 2010, pp. 135–144.

[116] V. H. The, O. Yuichi, and B.-A. Abderazek, "Comprehensive analytic performance assessment and k-means based multicast routing algorithm and architecture for 3d-NoC of spiking neurons," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 4, pp. 1–28, Dec 2019.

[117] V. H. The, M. I. Ogbodo, and B.-A. Abderazek, "A low-latency tree-based multicast spike routing for scalable multicore neuromorphic chips," in *Proceedings of the 9th International Conference on Information Systems and Technologies*. ACM, Mar 2019.

[118] D. N. Khanh, M. C. Meyer, B.-A. Akram, B.-A. Abderazek, and X.-T. Tran, "A non-blocking non-degrading multiple defects link testing method for 3D-networks-on-chip," *IEEE Access*, vol. 8, pp. 59 571 – 59 589, Mar. 2017.

[119] D. N. Khanh, B.-A. Akram, X.-T. Tran, Y. Okuyama, and B.-A. Abderazek, "A comprehensive reliability assessment of fault-resilient network-on-chip using analytical model," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 11, pp. 3099–3112, Nov. 2017.

[120] D. N. Khanh, B.-A. Akram, A. Ben-Abdallah, and X.-T. Tran, "A thermal-aware on-line fault tolerance method for tsv lifetime reliability in 3d-noc systems," *IEEE Access*, vol. 8, pp. 166 642–166 657, Sep. 2020.

[121] D. N. Khanh, M. Meyer, Y. Okuyama, and B.-A. Abderazek, "A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3d-noc systems," *The Journal of Supercomputing*, vol. 73, no. 6, pp. 2705–2729, Jun. 2017.

[122] D. N. Khanh, B.-A. Akram, B.-A. Abderazek, M. C. Michael, and T. Xuan-Tu, "2d parity product code for TSV online fault correction and detection," *REV Journal on Electronics and Communications*, vol. 10, no. 1-2, Aug. 2020.

[123] D. N. Khanh, M. Meyer, Y. Okuyama, and B.-A. Abderazek, "Reliability assessment and quantitative evaluation of soft-error resilient 3d network-on-chip systems," in *2016 IEEE 25th Asian Test Symposium (ATS)*, Nov. 2016, pp. 161–166.

[124] D. N. Khanh, B.-A. Akram, B.-A. Abderazek, and T. Xuan-Tu, "Thermal distribution and reliability prediction for 3d networks-on-chip," *VNU Journal of Science: Computer Science and Communication Engineering*, vol. 36, no. 1, Jun. 2020.

[125] D. N. Khanh, B.-A. Akram, O. Yuichi, and A. Ben-Abdallah, "Scalable design methodology and online algorithm for TSV-cluster defects recovery in highly reliable 3D-NoC systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 3, pp. 577–590, Oct. 2017.

[126] D. N. Khanh and T. Xuan-Tu, "An adaptive and high coding rate soft error correction method in network-on-chips," *VNU Journal of Science: Computer Science and Communication Engineering*, vol. 35, no. 1, Jun. 2019.

[127] B.-A. Akram and B.-A. Abderazek, "Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 4, pp. 2229–2240, Apr. 2014.

[128] D. N. Khanh, B.-A. Akram, B.-A. Abderazek, and X.-T. Tran, "Tsv-ias: Analytic analysis and low-cost non-preemptive on-line detection and correction method for tsv defects," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2019, pp. 501–506.

[129] D. N. Khanh, A. B. Ahmed, A. B. Abdallah, and X.-T. Tran, "Hotcluster: A thermal-aware defect recovery method for through-silicon-vias towards reliable 3-d ics systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, Mar. 2021.

[130] D. N. Khanh, B.-A. Akram, B.-A. Abderazek, and X.-T. Tran, "TSV-OCT: A scalable online multiple-TSV defects localization for real-time 3-D-IC systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 3, pp. 672–685, Mar. 2019.

[131] D. N. Khanh, B.-A. Akram, Y. Okuyama, and B.-A. Abderazek, "Scalable design methodology and online algorithm for tsv-cluster defects recovery in highly reliable 3D-NoC systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 3, pp. 577–590, Jul 2020.

[132] B.-A. Abderazek, V. H. The, and H. Masayuki, "Neural computing architecture, fault-tolerant algorithm, and design method for spiking neural networks," Japan Patent 2019-124 541, 2019.

[133] V. H. The, Y. Okuyama, and A. Ben-Abdallah, "Analytical performance assessment and high-throughput low-latency spike routing algorithm for spiking neural network systems," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 5367–5397, Mar. 2019.

[134] J. Yu, Z. Youhui, L. He, and Z. Weimin, "Optimized mapping spiking neural networks onto network-on-chip," *Algorithms and Architectures for Parallel Processing*, pp. 38–52, Dec. 2016.

[135] D. N. Khanh, N. A. V. Doan, and A. Ben-Abdallah, "Migspike: A migration based algorithms and architecture for scalable robust neuromorphic systems," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, Dec. 2021.