

Design and FPGA Implementation of Optimized and Digital Multiplexing Spiking Encoders for Neuromorphic Computing

Ruizhe Li

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

Master of Science
in
Computer Engineering

Yang Yi, Chair
Dong S. Ha
Jason S. Thweatt

October 2, 2025
Blacksburg, Virginia

Keywords: Artificial Intelligence, Recurrent Neural Network, Field Programmable Gate Array, Spiking Encoding, Hardware Accelerator.

Copyright 2025, Ruizhe Li

Design and FPGA Implementation of Optimized and Digital Multiplexing Spiking Encoders for Neuromorphic Computing

Ruizhe Li

(ABSTRACT)

Machine Learning (ML) and Artificial Intelligence (AI) have driven rapid progress in wireless communications, particularly in spectrum sensing for cognitive radios. Accurate identification of occupied and unoccupied spectrum bands is vital to address spectrum scarcity and enable secondary users to make efficient use of available frequencies. Liquid State Machines (LSMs), built on spiking neural networks inspired by biological computation, provide a promising framework for real-time spectrum monitoring. A crucial step in this process is spike encoding, which converts continuous input signals into discrete spike trains. This encoding not only preserves temporal features of the spectrum but also allows the reservoir to exploit time-dependent dynamics for richer representations. By combining spike encoding with the inherent temporal processing capabilities of LSMs, specialized accelerators and processors achieve faster, more energy-efficient, and more accurate spectrum classification than conventional sensing methods. Furthermore, the design of training algorithms and optimized learning algorithms within LSM processors plays a pivotal role in maximizing their performance. This thesis is divided into two main parts. The first focuses on the development and implementation of learning algorithms for spectrum classification using an LSM processor as part of my research work during my master's study. The second part, as my primary research work, investigates and optimizes the spiking encoding algorithm to improve input representation and system efficiency. Both parts are realized on Field-Programmable Gate

Arrays (FPGAs), leveraging their programmability and massively parallel computing capabilities to enable high-performance hardware implementation. Lastly, the thesis provides a comprehensive analysis of the proposed spiking encoder and points out the future research direction of the spiking encoding algorithm. The results show that the proposed multiplexing temporal encoding method surpasses existing state-of-the-art rate and temporal encoding algorithms in both performance and noise robustness across most test cases. It achieves up to a $9.95\times$ improvement in signal reconstruction accuracy compared to other encoding techniques. In addition, the optimized rate encoding architectures outperform current models, providing up to a $3.5\times$ accuracy gain across all evaluated signals. Furthermore, the proposed TTFS-based architecture delivers up to a $5.5\times$ improvement in accuracy over existing methods while simultaneously reducing logic resource utilization by 57.1%, highlighting its efficiency for hardware implementation.

Design and FPGA Implementation of Optimized and Digital Multiplexing Spiking Encoders for Neuromorphic Computing

Ruizhe Li

(GENERAL AUDIENCE ABSTRACT)

Wireless communication systems increasingly rely on intelligent technologies such as Machine Learning (ML) and Artificial Intelligence (AI) to use limited radio frequencies more efficiently. One major challenge is identifying which parts of the spectrum are already in use and which remain available—a process known as spectrum sensing. Addressing this challenge helps prevent interference and allows more users to share the same communication resources. This research explores a brain-inspired computing approach called the Liquid State Machine (LSM) to improve real-time spectrum monitoring. An essential part of this system is spike encoding, a method that converts continuous radio signals into brief electrical pulses similar to those used by biological neurons. This process preserves important timing information, enabling the LSM to recognize complex patterns in wireless data with higher accuracy and speed. The work presented in this thesis is divided into two main areas. The first develops and tests learning algorithms that allow the LSM to classify spectrum activity accurately. The second focuses on improving the spike encoding method to represent input signals more effectively. Both components are implemented on Field-Programmable Gate Arrays (FPGAs)—reconfigurable hardware devices that support fast, parallel computation with low power consumption. The study concludes with an analysis of the proposed spike encoding approach and suggests future directions for advancing this technique in next-generation neuromorphic communication systems.

Dedication

To my grandparents, parents, friends, and professors

Acknowledgments

As I reach the completion of my master's studies at Virginia Tech, I would like to express my deepest gratitude to all those who have supported, guided, and inspired me throughout this journey. First and foremost, I am sincerely thankful to my advisor, Yang (Cindy) Yi, for her invaluable guidance, patience, and encouragement. Their insight and dedication have profoundly shaped both my research and my approach to scientific inquiry. I am equally grateful to the members of my committee for their thoughtful feedback and constructive suggestions that strengthened this work. I would also like to thank my lab colleagues and friends in the MICS lab for their collaboration, discussions, and shared enthusiasm for research. Their friendship made long days in the lab both productive and memorable. I am deeply appreciative of the support provided by the Department of Electrical and Computer Engineering and Virginia Tech for offering a stimulating academic environment and the resources that made this research possible. Finally, I owe my heartfelt thanks to my family for their unwavering love, patience, and belief in me. Their encouragement has been my greatest motivation and source of strength throughout my academic journey.

Contents

List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Spectrum Sensing in Wireless Communication	1
1.2 The Challenge of AI in Spectrum Sensing	2
1.3 Spiking Neural Network	4
1.3.1 Biological Neuron	4
1.3.2 Spiking Neuron	5
1.3.3 Encoding Scheme in Spiking Neural Network	7
1.4 Outline	8
2 On-Chip Training of Liquid State Machine for Spectrum Sensing	9
2.1 Reservoir Computing	9
2.1.1 Advantages of Using Liquid State Machine in Spectrum Sensing	12
2.2 Implementation of FPGA-based Liquid State Machine Accelerator for Spectrum Sensing	16
2.2.1 Implementation of Readout Units	16

2.2.2	Implementation of Learning Engine in Readout Unit	17
3	Neural Encoding Algorithms and their Hardware Architecture	20
3.1	Rate Encoding	20
3.2	Time to First Spike Encoding	22
3.3	Inter-Spike-Interval Encoding	24
3.4	Phase Encoding	26
3.5	Multiplexing Encoding	29
4	Architecture of Proposed Digital Encoder	33
4.1	Proposed Digital Encoder	33
4.1.1	Proposed Digital Rate Encoder	33
4.1.2	Proposed Digital TTFS encoder	34
4.1.3	Proposed Digital Multiplexing Encoder	36
5	Experimental Setup and Results	40
6	Discussion	47
7	Conclusions	49
	Bibliography	51

List of Figures

1.1	Demonstration of a biological neuron [1]	5
1.2	Demonstration of LIF neuron in RC circuit	6
2.1	Demonstration of Echo State Network(ESN)	11
2.2	Demonstration of Liquid State Machine (LSM)	12
2.3	Overall digital architecture of proposed LSM	15
2.4	Digital architecture of a RUs	17
2.5	Digital architecture of a Learning Engine (LE) in RU	19
3.1	Rate encoding demonstration	21
3.2	State-of-art digital hardware rate encoder [2]	22
3.3	TTFS encoding demonstration	23
3.4	State-of-art digital hardware TTFS encoder [2]	24
3.5	ISI encoding demonstration	26
3.6	State-of-art digital hardware ISI encoder [2]	27
3.7	Illustration of phase encoding	28
3.8	State-of-art digital hardware phase encoder [2]	28
3.9	Illustration of PWM-based phase encoding	29

3.10	Illustration of multiplexing encoding. (a) TTFS-phase encoding. (b) ISI-phase encoding	31
4.1	Hardware implementation of the proposed rate encoder.	34
4.3	Demonstration of PWM alignment	37
4.2	Hardware implementation of the proposed TTFS encoder	38
4.4	(a) Hardware implementation of the proposed multiplexing encoder (b) Proposed PWM alignment module	39
5.1	Testing signals used for analysis. (a) Sine component with random power and phase lag with white noise (b) 3Hz smooth sinusoidal signal with white noise (SS) (c) Constantly rising and falling signal (CRF) (d) Step-wise signal(SW)	41

List of Tables

5.1	Accuracy comparison for rate encoder	44
5.2	Accuracy comparison for TTFS encoder	44
5.3	Accuracy comparison for multiplexing encoder	45
5.4	Resource utilization, Power consumption and Fmax for all the encoders . . .	45

List of Abbreviations

AI Artificial Intelligence

ANN Artificial Neural Network

CNN Convolutional Neural Network

FFT Fast Fourier Transform

FPGA Field Programmable Gate Array

LIF Leaky Integrate and Fire model

LSM Liquid State Machine

MIMO Multiple Input Multiple Output

ML Machine Learning

OFDM Orthogonal Frequency Division Multiplexing

RC Reservoir Computing

RNN Recurrent Neural Network

RSTDP Reward-modulated Spike Timing Dependent Plasticity

SNN Spiking Neural Network

Chapter 1

Introduction

1.1 Spectrum Sensing in Wireless Communication

Spectrum sensing is a fundamental technique of cognitive radio which identifies the idle or busy bandwidths [3]. Under an unforeseen spectrum situation, users within the intelligent network need to detect the spectrum and identify available frequencies for their use. To improve the spectral efficiency, Multiple-Input-Multiple-Output (MIMO) and Orthogonal Frequency-Division Multiplexing (OFDM) systems are proposed by highlighting arrays of antennae used for data transmission and receiving simultaneously, and divides a wide-band channel into many closely spaced orthogonal sub-channel. Fast Fourier Transformations (FFTs) are employed for modulating symbols(i.e.,data) into Radio Frequency(RF) waves on the transceiver side, and for demodulating these waves back into symbols at the receiving side of the system. Thus, the combination of MIMO-OFDM system improves the spatial multiplexing gain and avoids frequency selective fading accordingly [4].

In the process of spectrum sensing, the energy detected by the antennas of a MIMO-OFDM receiver is utilized to ascertain the occupancy status of a sub-carrier frequency. Therefore, spectrum sensing can be recognized as a binary classification task in machine learning, whether the sub-carrier is idle or busy. The dataset compiled following the procedure from K.Hamedani et al.[5] was employed in our study, comprising measurements of static spectrum occupancy, detailing Primary User (PU) activity across a specific range of frequency bands

and time intervals. The occupancy status of each sub-carrier is determined using frequency occupancy models derived from a database. More importantly, OFDM symbols are conveyed using the q_{th} sub-carrier frequency and then are modulated through Quadrature Phase Shift Keying (QPSK). The output signal can be defined as the equation (1.1).

$$R_q(m) = Y_q(m) + N_q(m) \quad (1.1)$$

where $Y_q(m)$ is the transmitted signal in frequency domain and $N_q(m)$ is the Discrete Fourier Transform (DFT) format of Additive White Gaussian Noise (AWGN). And $m = 1, \dots, N$ where N is the number of OFDM symbol received by the system. When the q_{th} sub-carrier remains unused, and no signal is present on it, the representation of the received signal is as follows: $R_q(m) = N_q(m)$. Furthermore, we take into account signals received across multiple antennae by employing the following equation (1.2).

$$R_q^j(m) = Y_q^j(m) + N_q^j(m) \quad (1.2)$$

where $j = 1, \dots, X$ and X is the number of antennae configured in the system.

1.2 The Challenge of AI in Spectrum Sensing

Spectrum sensing is essential in cognitive radio systems for determining whether a frequency band is occupied or available. The occupancy of a given spectrum band exhibits temporal correlation, meaning its usage at a specific moment is influenced by past and future states. Among various spectrum sensing methods, energy detection remains the most widely adopted due to its simplicity, low data requirements, and energy efficiency [6].

In classical spectrum sensing, the energy detector is a simple model-based method that

operates on a single feature: the received signal energy. It does not rely on transmitter-specific information to decide whether a band is occupied by a primary user (PU). In practice, the detector compares the measured energy to a threshold to discriminate between idle and occupied. This approach is attractive for hardware because it is compact and power-efficient [6]. However, its accuracy degrades sharply in noisy channels due to the SNR wall, where increasing observation time no longer improves detection performance [7, 8].

Recent advances in AI offer a path forward by learning discriminative structure from limited inputs, so that spectrum vacancies can be classified more reliably. Yet conventional artificial neural networks (ANNs), especially deep models, are often ill-suited to low-cost devices like FPGAs: their training depends on backpropagation, which incurs intensive multiply-accumulate operations and severe memory bandwidth pressure for gradient updates across layers [9]. These constraints motivate hardware-aware learning schemes (e.g., event-driven or locally updated methods) that retain the efficiency of simple detectors while improving robustness at low SNR.

Reservoir computing, proved its potential for various real-world applications [10, 11, 12, 13, 14, 15, 16, 17, 18], has recently also gained attention in the wireless domain [19, 20, 21]. Its ability to efficiently process temporal dependencies makes it particularly suited for spectrum sensing, where identifying patterns in spectrum occupancy is crucial for optimizing dynamic spectrum access. Moreover, there have been recent researches on exploring in-memory computing for hardware efficient reservoir computing at device level in analog domain using different resistive ram based technology [22, 23, 24, 25, 26, 27, 28].

1.3 Spiking Neural Network

The human brain, with its remarkable efficiency and adaptability, has long been an inspiration for Artificial Intelligence (AI) and Machine Learning (ML) [29]. Unlike traditional Artificial Neural Networks (ANNs), which rely on continuous numerical representations and high-power computations, biological neurons communicate through discrete electrical impulses known as spikes. This has led to the development of Spiking Neural Networks (SNNs)—a class of neural models that more closely emulate biological processes by utilizing time-dependent spike-based communication. SNN has widely adopted in various applications due to its high energy efficiency compare with ANNs [30, 31, 32]. The cornerstone of SNNs is the spiking neuron model, which governs how information is encoded, processed, and transmitted in a biologically plausible manner [33].

1.3.1 Biological Neuron

The biological neuron consists of dendrites, nucleus, cell body, axon and axon terminal as shown in Figure 1.1 where dendrites conduct electrical impulses toward the cell body, playing a crucial role in integrating incoming signals, nucleus regulates the neuron’s activities, including protein synthesis and cell maintenance, cell body integrates incoming signals from dendrites and, if the combined signal is strong enough, generates an action potential that travels down the axon, axon serves as the main transmission pathway for neural signals, ensuring efficient communication over long distances and axon terminals enabling signal transmission to the next neuron.

In biological neurons, spikes, also known as action potentials, are the primary means of transmitting information. These spikes are brief electrical signals that travel along the axon to communicate with other neurons. The generation and transmission of a spike follow a well-

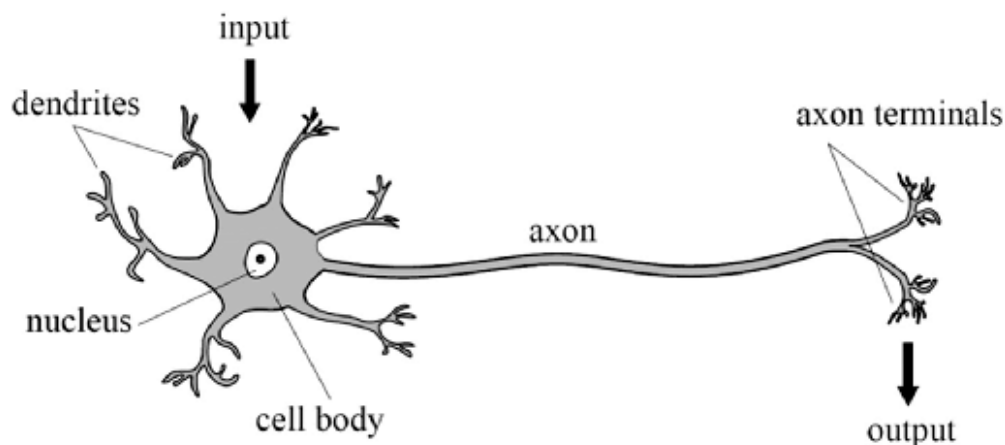


Figure 1.1: Demonstration of a biological neuron [1]

defined process based on the neuron’s membrane potential and ion exchange. The working process of biological neurons can be briefly summarized as that neuron will fire spikes once it reach a certain threshold and then the neuron will be reset and enter a refractory period to prevent excessive firing to the next neuron.

1.3.2 Spiking Neuron

To gain a deeper understanding of biological neurons, researchers have developed various spiking neuron models, including the Hodgkin-Huxley (HH) model [34], Leaky Integrate-and-Fire (LIF) model [35], Izhikevich model [36], and Adaptive Exponential Integrate-and-Fire (AdExIF) model [37]. These models differ in complexity and biological accuracy, offering different trade-offs for computational efficiency and fidelity.

Among them, the LIF model is one of the most widely adopted in Spiking Neural Networks (SNNs) due to its simplicity and computational efficiency. It serves as a foundation for both hardware and software implementations, making it a preferred choice for neuromorphic computing applications. In this study, the spiking neuron is modeled using the Leaky

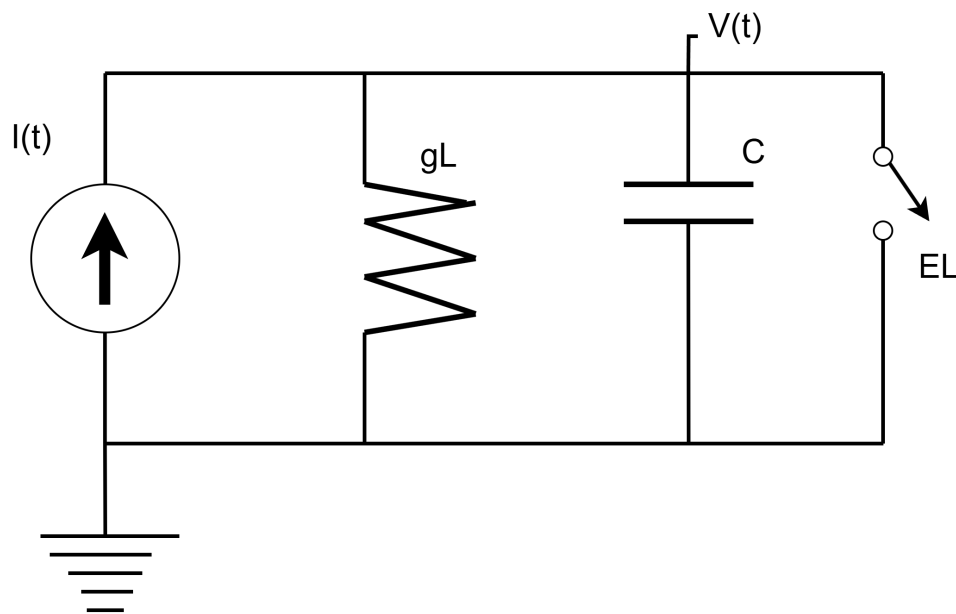


Figure 1.2: Demonstration of LIF neuron in RC circuit

Integrate-and-Fire (LIF) framework, which provides a balance between computational efficiency and biological relevance. LIF neuron is often modeled as a parallel Resistor-capacitor (RC) circuit with a "leaky" resistor as shown in Figure 1.2.

The output voltage $V(t)$ is mathematically defined as:

$$C \frac{dV}{dt} = -gL(V(t) - EL) + I(t) \quad (1.3)$$

where gL represents the conductance of the resistor, C denotes the capacitance of the capacitor, and $I(t)$ is the input current that charges the capacitor, influencing the membrane potential $V(t)$. When $V(t)$ surpasses a defined threshold, the capacitor discharges to its resting potential EL through a voltage-controlled switch, triggering the generation of a spike.

1.3.3 Encoding Scheme in Spiking Neural Network

Neural encoding schemes are fundamental to the operation of neuromorphic computing systems, which seek to mimic the brain's working mechanism and architecture by processing information through Spiking Neural Networks (SNNs) [2, 38]. These algorithms are responsible for converting continuous sensory input, such as visual or auditory data, into discrete electrical spikes that neurons can process [39, 40]. Thus, the neural encoding algorithm serves as the interface between the real-world signal and the digital processing mechanisms of artificial neural systems.

Historically, the encoding algorithm in neuromorphic computing has drawn inspiration from biological neurons, where information is represented not by the intensity of electrical signals but by the timing and frequency of spikes. This shift from rate-based encoding, where information is conveyed by the firing rate of neurons, to temporal encoding, which leverages the precise timing of spikes, has provided a new perspective for developing an efficient, high-performance computational model.

Two widely used encoding schemes in this domain are rate encoding and temporal encoding algorithms. Rate encoding is computationally simple, mapping the intensity of the input to the number of spikes within a given time window. While this method has been effective in early applications, it lacks the data density and precision needed for complex tasks. Temporal encoding, on the other hand, encodes information using the precise timing of individual spikes. This method allows for a more subtle representation of data, capturing both the rate and timing of neural spikes, and is known to improve the efficiency and accuracy of neuromorphic systems, particularly in noisy environments.

Recent advancements have led to the development of more sophisticated encoding algorithms such as multiplexing encoding [41, 42, 43, 44]. These methods integrate multiple timescales

of data encoding, combining rate and temporal encoding [45]. By leveraging multiplexing, neural systems can transmit significantly more information using fewer spikes, which enhances the robustness and efficiency of neuromorphic systems in various applications.

As neuromorphic computing becomes more prominent in applications like robotics [46], Artificial Intelligence [47], and wireless communication [48, 49], neural encoding algorithms continue to evolve. They not only help bridge the gap between biological and artificial intelligence but also enable the creation of systems that are faster, more energy-efficient, and capable of handling complex, real-world tasks.

1.4 Outline

The structure of the thesis is organized as the following chapters: Chapter 2 first introduce the Reservoir Computing in general and explain the advantage of using Liquid State Machines (LSMs) in Spectrum Sensing. Then, it talks about the FPGA implementation of LSM as well as it's learning algorithm on hardware. This chapter serves as the secondary works in my master's study. More details can be found in [50, 51, 52]. Chapter 3 explains the existing spiking neural encoding scheme such as rate, Time-To-First-Spike (TTFS), Inter-Spike-Interval (ISI), Phase, and Multiplexing encoding as well as their hardware implementation. Chapter 4 discuss the proposed hardware rate, TTFS and multiplexing encoder. Chapter 5 explores the experimental setup and results. Chapter 6 and 7 discuss the existing finding and concludes the thesis by summarizing the results and insights. It also outlines potential research direction within the domain.

Chapter 2

On-Chip Training of Liquid State Machine for Spectrum Sensing

2.1 Reservoir Computing

Reservoir Computing (RC) is a brain-inspired computational paradigm derived from Recurrent Neural Networks (RNNs). Unlike conventional RNNs, the recurrent connections within the reservoir are fixed after initialization, while only the readout layer weights are trained to produce the desired output [53]. The static internal connections create a nonlinear dynamic system that projects low-dimensional inputs into a high-dimensional feature space. The resulting complex temporal responses are then processed by the readout layer, where lightweight learning algorithms can efficiently perform the final mapping.

There are two typical structure of RC, one is Echo State Network (ESN), which is normally implemented with non-spiking, discrete-time artificial neurons as shown in Figure 2.1. When no feedback is applied from the output to the reservoir, the temporal evolution of the neuronal states within the reservoir can be expressed as follows [54]:

$$\mathbf{x}(n) = f(W^{\text{in}}\mathbf{u}(n) + W\mathbf{x}(n-1)) \quad (2.1)$$

Here, n represents the discrete time step, $x(n)$ denotes the state vector of the reservoir

neurons, and $u(n)$ is the input vector. W^{in} corresponds to the input-to-reservoir weight matrix, while W represents the recurrent weight matrix within the reservoir. The function f is an element-wise nonlinear activation applied to each reservoir unit, commonly implemented as a sigmoid-type function. Equation 2.1 defines a non-autonomous dynamical system driven by the external input $u(n)$. The corresponding output is typically obtained as a linear combination of the reservoir states, expressed as follows:

$$\mathbf{y}(n) = W^{out}\mathbf{x}(n) \quad (2.2)$$

where $y(n)$ is the output vector and W^{out} is the weight matrix in the readout layer. The ESN consists of three main components: the input layer, the reservoir, and the readout layer. The input layer receives external signals and projects them into the reservoir through fixed weighted connections. The reservoir, built as a recurrent neural network with randomly connected neurons, forms the core of the system. Its internal connections remain unchanged after initialization, creating a nonlinear dynamic environment. This allows the reservoir to transform simple input sequences into complex, high-dimensional patterns that capture temporal dependencies. Finally, the readout layer processes these rich internal states. Unlike the reservoir, the readout layer is trainable, typically through simple and efficient learning methods. This separation of fixed internal dynamics and lightweight training makes the echo state machine a powerful yet computationally efficient model for handling sequential data.

The other type of RC is Liquid State Machine (LSM), which is more biologically plausible as it is based on the Spiking Neural Networks (SNNs) with recurrent reservoir structures as shown in Figure 2.2 and the reservoir units are typically given by excitatory and inhibitory spiking neurons with Leaky Integrate-and-Fire (LIF) neurons, other biologically plausible spiking neuron models can also be used [55, 56]. The topology and connectivity of the RNN in a LSM are designed to mimic the structural properties of biological neural systems. In

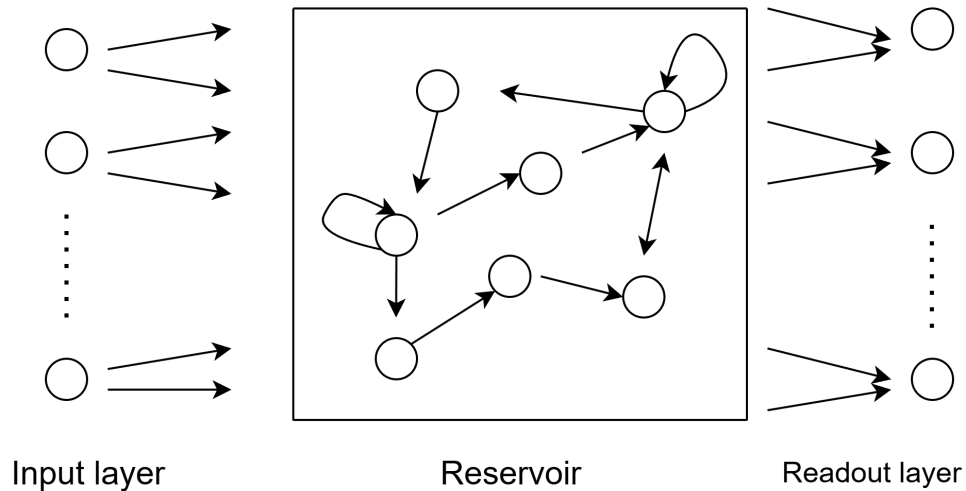


Figure 2.1: Demonstration of Echo State Network(ESN)

particular, the likelihood of a connection between two neurons decreases with increasing distance between their spatial positions. This dynamic network, often referred to as a “liquid,” gives rise to the term liquid computing, as its behavior resembles excitable media that generate ripples in response to external stimuli. The evolution of the reservoir’s dynamics can be generally expressed as follows [57]:

$$\mathbf{x}^m(t) = (L^m u)(t) \quad (2.3)$$

Here, t denotes continuous time; x^m is the reservoir state (neural activity pattern); $u(\cdot)$ is the input encoded as a spike train; and L^m is the operator (filter) that maps the input to the reservoir state. The output is given by:

$$\mathbf{y}(n) = f^m(\mathbf{x}^m(t)) \quad (2.4)$$

where $y(t)$ denotes the output and f^m is a memory-less readout function that maps the reservoir state to $y(t)$. The LSM has a similar structure as ESN, including the input layer, the

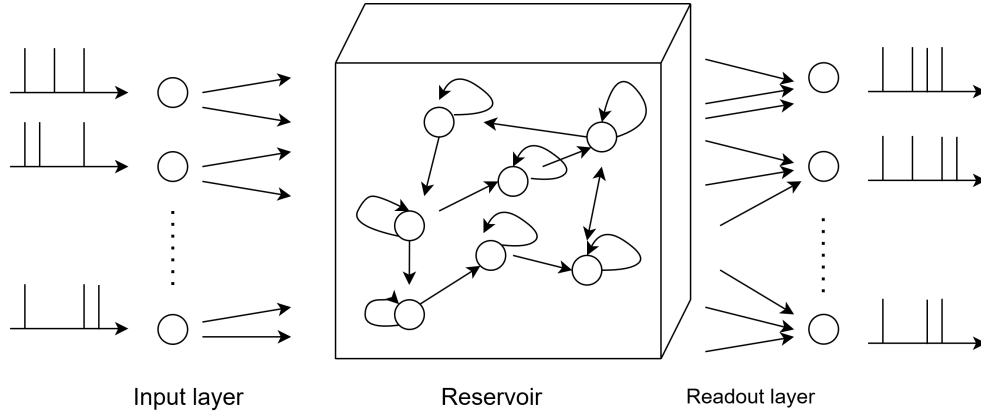


Figure 2.2: Demonstration of Liquid State Machine (LSM)

reservoir, and the readout layer. The input layer delivers external signals—often encoded as spikes—into the reservoir through fixed connections. The reservoir, also called the “liquid,” is a recurrent network of spiking neurons. Its neurons are densely and randomly interconnected, allowing the network to generate complex, time-varying activity in response to incoming spikes. This dynamic activity acts as a rich representation of the input, spreading information across both space and time, much like ripples in a liquid when disturbed. The readout layer receives these dynamic states and extracts meaningful outputs. Only the readout connections are trained, typically with simple supervised learning rules. This design allows the liquid state machine to capture temporal patterns in spiking inputs efficiently, while keeping the training process computationally lightweight.

2.1.1 Advantages of Using Liquid State Machine in Spectrum Sensing

The traditional energy detector remains widely used for spectrum sensing because it requires little data, has low resource cost, and is highly energy efficient [58]. However, its performance degrades in high-SNR channels with impulsive noise, where it lacks robust noise

handling [7, 8]. Recent AI approaches, such as Convolutional Neural Networks (CNNs), have surpassed detector-based baselines in accuracy [59, 60]. Yet deep AI models and most non-SNN accelerators are less hardware-friendly due to extensive Multiply-and-Accumulate (MAC) operations, whereas SNNs rely primarily on accumulation [61]. Moreover, on-chip learning with SNN accelerators—such as those inspired by LSMs—avoids backpropagating gradients through deep stacks to update every neuron, reducing memory contention [61]. Consequently, conventional ANN-based accelerators are a weaker fit for edge spectrum sensing scenarios that require real-time weight adaptation under changing conditions.

On the other hand, Liquid State Machines (LSMs) are well suited to spectrum sensing when both high accuracy and real-time adaptation are required. By exploiting local learning, an LSM can capture the spatio-temporal structure of incoming signals and perform competitive classification [62, 63]. A learning rule is considered local when its computations can be decomposed into small, independent steps that rely only on information available at each processing element, enabling parallel, distributed updates [64]. An efficient hardware implementation of an LSM can mitigate SNR-wall effects in noisy channels and surpass existing ML accelerators for spectrum sensing, while also outperforming the advanced LSM design in [65] with higher accuracy and shorter training time.

Learning in Spiking Neural Networks is driven by the precise timing of spikes rather than continuous activation values. Various approaches have been proposed, spanning unsupervised plasticity, supervised gradient-based optimization, and reinforcement learning. In this work, Reward-Modulated Spike-Timing-Dependent Plasticity (R-STDP) is employed as an efficient and biologically inspired learning mechanism. This rule extends conventional timing-based plasticity by introducing a global reward signal that reinforces or suppresses synaptic changes based on task performance. When pre- and postsynaptic spikes coincide within a defined window, they generate short eligibility traces indicating potential weight updates; the arrival

of a reward or penalty then determines whether these traces are consolidated or diminished. R-STDP integrates naturally with the Liquid State Machine framework, where the reservoir remains static and only the readout layer learns. By combining local spike-based adaptation with a simple global feedback signal, R-STDP enables the readout to translate the reservoir's dynamic spatiotemporal activity into accurate decisions. This produces a compact and fully online learning process ideal for hardware deployment. Implemented on FPGA, R-STDP leverages localized computation, minimal arithmetic, and bounded weight updates to deliver fast, energy-efficient spectrum classification while avoiding the complexity and memory demands of gradient-based backpropagation.

In this thesis, we use Reward-based STDP(R-STDP) as the supervised learning method for training the readout layer weights in the LSM. The equation for the R-STDP learning is written in the following :

Reward-based STDP equation:

$$\text{Weight}_{ij}^{new} = \text{Weight}_{ij}^{old} + \Delta w_{ij}^{\text{potentiation/depression}} \quad (2.5)$$

Where:

$$\Delta w_{ij}^{\text{potentiation}} = A_{\text{pos}} \cdot \exp\left(-\frac{\Delta t_{\text{potentiation}}}{\tau_{\text{pos}}}\right) \quad (2.6)$$

$$\Delta w_{ij}^{\text{depression}} = -A_{\text{neg}} \cdot \exp\left(-\frac{\Delta t_{\text{depression}}}{\tau_{\text{neg}}}\right) \quad (2.7)$$

Weight $_{ij}$ is trainable synapse between i and neuron j ,

$\Delta w_{ij}^{\text{potentiation}}$ is the reward amount for potentiation,

$\Delta w_{ij}^{\text{depression}}$ is the punishment amount for depression,

Δt timing difference between spike pairs

A_+ & A_- are strength parameters, $\tau_{\text{pos/neg}}$ are time constants

The magnitude of synaptic modification during potentiation and depression can be finely controlled by adjusting the corresponding constants in equations 2.6 and 2.7, allowing the learning rule to be optimized for various application requirements. The overall LSM architecture is shown in Figure 2.3, where there are 16 Learning Units (LUs) in reservoir and 2 Readout Units (RUs) in the readout layer. Each LU receives 1 external spike from input sample spike-train and 2 recurrent spikes from spike record register at each time-step to it's internal module. The internal module of both RU units receive all the 16 spike-record events at each time step. CT1 and CT2 are two classification teacher signals for updating the related RU according to target label. This thesis will focus on explaining the highlight part, which is the readout layer of the LSM as more details can be found in [50].

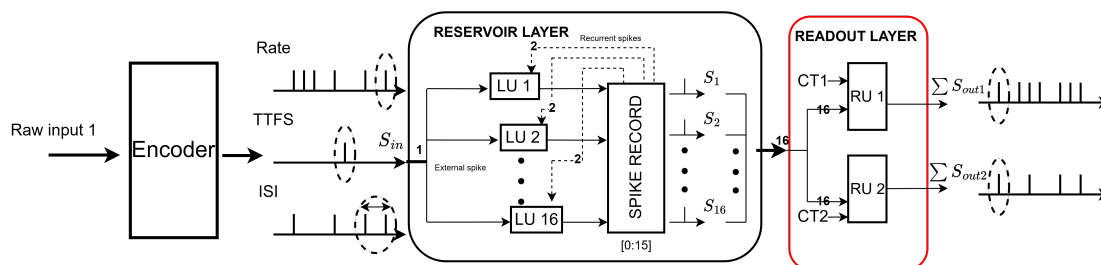


Figure 2.3: Overall digital architecture of proposed LSM

2.2 Implementation of FPGA-based Liquid State Machine Accelerator for Spectrum Sensing

2.2.1 Implementation of Readout Units

The implementation of RU follows Leaky Integrate-and-Fire (LIF) neuron [66], a widely used neuron model in SNN where dynamics can be characterized by

$$V_{mem}(t) = V_{mem}(t-1) - \frac{V_{mem}(t-1)}{\tau_{mem}} + \sum W_i * S_i \quad (2.8)$$

where $V_{mem}(t)$ means the membrane potential of LIF model at time step t , τ_{mem} is decay constant, W_i and S_i are the afferent presynaptic weight and spiking event from i_{th} presynaptic neuron respectively. W_i is added to $V_{mem}(t)$ if i_{th} presynaptic neuron fires at time $t-1$ and S_i is 1 as a result. If the $V_{mem}(t)$ is higher than the threshold value V_{th} at a timestep t , then the current neuron spikes and the $V_{mem}(t)$ is set to rest at 0. The V_{th} threshold is governed by an adaptive model in equation 2.9, where the threshold increases by a constant C_{th} if spikes are generated from the current neuron at consequent timesteps.

$$V_{th}(t) = V_{th}(t-1) - \frac{V_{th}(t-1)}{\tau_{th}} + C_{th} \quad (2.9)$$

The adaptive threshold mechanism stabilizes learning by preventing weight saturation and suppressing repetitive spike patterns that contribute little new information. Figure 2.4 illustrates the architecture of the Readout Unit (RU), which is composed of two main modules: the Learning Engine (LE), responsible for training the afferent synaptic weights of the neuron, and the *Spike Generator*, which performs computations based on equations 2.8 and 2.9.

During operation, each neuron retrieves presynaptic spike inputs through a parallel-in-serial-

includes sixteen pre-synaptic shift registers, each loading one spike bit into its Most Significant Bit (MSB) position, as shown in Figure 2.5. In parallel, a post-synaptic shift register captures the spike event produced by the RU’s leaky-integrate-and-fire (LIF) comparator. All shift registers are 12 bits deep and shift only when the comparator generates a spike (‘1’) event. A 16-to-1 Multiplexer (MUX) then selects one pre-synaptic register for the timing-difference computation module, which measures the temporal difference between pre- and post-synaptic spikes. When the MSB of any register becomes ‘1’, a priority encoder identifies the first active bit within twelve indices of the paired register to compute the spike-time difference Δt . A positive Δt (post-before-pre) results in synaptic potentiation, whereas a negative Δt induces depression.

The absolute value of Δt is mapped to a probability through a lookup table (LUT) following the activity-dependent STDP function reported by Jin *et al.* [67]. Both the STDP reward LUT and the probability LUT are pre-calibrated in fixed-point precision using software simulations to ensure hardware stability. Smaller timing differences yield higher update probabilities, while larger differences correspond to lower probabilities. This probability is compared against a random number generated by the enable (ENA) signal, forming an activity-dependent Hebbian update condition.

The sign bit of Δt determines whether the corresponding weight is increased (potentiation) or decreased (depression). Training is enabled only when both the ENA and classification-teacher (CT) signals are asserted, allowing weight modification for the target neuron based on the reservoir state derived from the current training sample. Weight updates are stored in the main weight memory implemented with dual-port RAM in a pipelined structure, allowing one update per clock cycle. The memory read address is synchronized with the MUX selection bit to retrieve the corresponding weight for the chosen pre-synaptic input. After each iteration, the adjusted weights are temporarily held in a dedicated register before

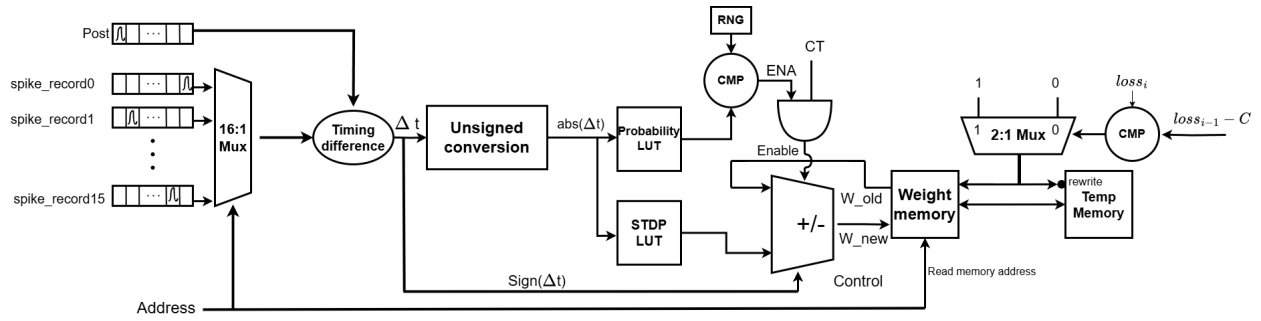


Figure 2.5: Digital architecture of a Learning Engine (LE) in RU

being written back to memory, ensuring continuous and efficient online learning across the entire readout layer.

Chapter 3

Neural Encoding Algorithms and their Hardware Architecture

3.1 Rate Encoding

In rate encoding, the strength of the input signal, whether it's from sensory data like sound or light, is converted into the number of spikes generated within a certain time frame. In other words, a stronger input results in a higher firing rate, while a weaker input results in fewer spikes. This approach is inspired by how biological neurons communicate in the brain. They adjust the frequency of their firing based on the intensity of the stimuli they receive. For example, imagine a neuron responding to an increase in light intensity. When the light is dim, the neuron might fire only a few spikes per second. As the light becomes brighter, the neuron fires more frequently, conveying the intensity of the light through its firing rate.

In rate neural encoding, the quantity of spikes that occur within a specific time interval reflects the strength or magnitude of the input signal. This results in a direct proportionality between the firing rate of the neuron and the intensity of the stimulus. Consequently, a Poisson spike train, as illustrated in Figure 3.1, can be generated using this approach. The Poisson spike train is produced by comparing a scaled version of the input signal to a randomly generated number. Rate encoding is widely adopted due to its straightforwardness and efficiency, making it highly suitable for fast implementation in both hardware and

software systems.

In advanced hardware architectures, a common approach for modeling a Random Number Generator (RNG) that follows a Poisson probability distribution involves the use of an n -bit linear feedback shift register, incorporating a specific number of XOR gate feedbacks [2]. In a rate encoder as shown in Figure 3.2, the number of RNGs corresponds to the total number of timesteps, denoted as t , in the input signal. The input values for each timestep are compared simultaneously with their respective RNG-generated numbers, which serve as thresholds in the comparators. These comparators then determine whether a spike should be generated based on the comparison. The total number of comparators in the system is also equal to the number of timesteps, t . Thus, we get a certain length of spike train equivalent to the total timestep t of an input signal. All the LFSR bit widths are n such that $2^n - 1 \approx$ maximum amplitude of the scaled signal.

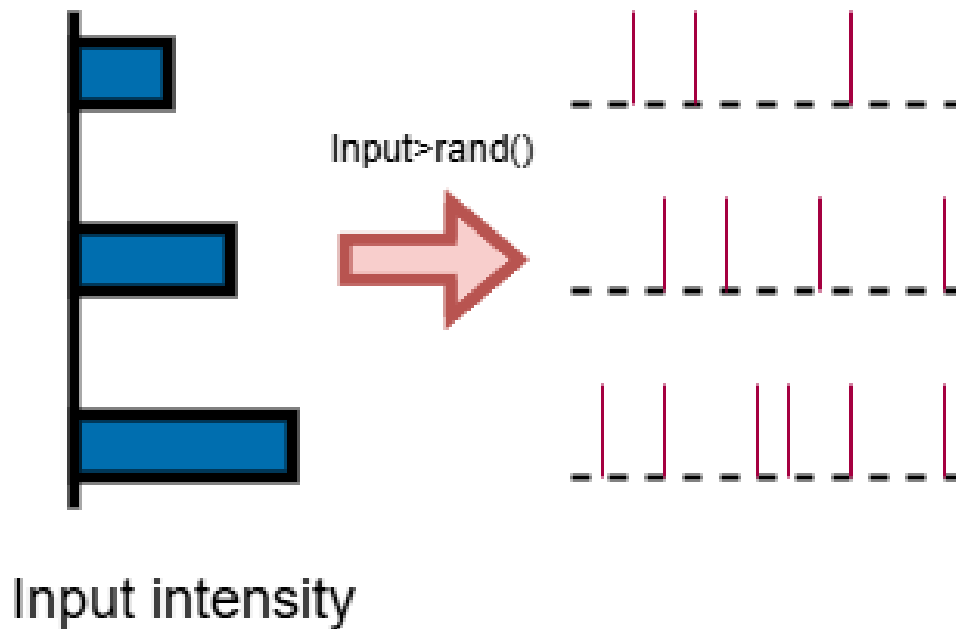


Figure 3.1: Rate encoding demonstration

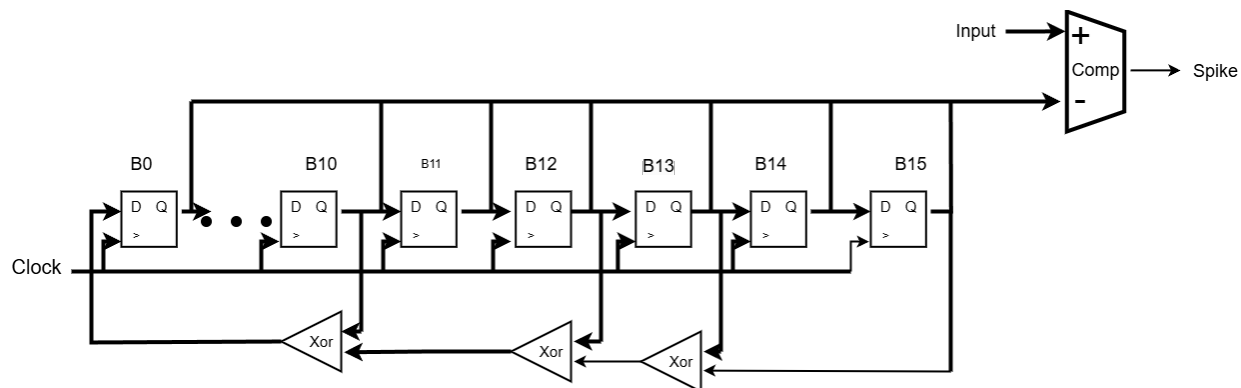


Figure 3.2: State-of-art digital hardware rate encoder [2]

3.2 Time to First Spike Encoding

The time-to-first-spike (TTFS) encoding is a neural coding algorithm where the timing of the first spike from a neuron is used to encode information about an input stimulus. Unlike rate encoding, where the number of spikes over a period represents the input magnitude, TTFS relies on the latency or delay of the first spike to convey information. In this method, stronger or higher-intensity stimuli cause neurons to fire earlier, while weaker inputs result in delayed firing.

In TTFS encoding, each neuron responds to an input stimulus by generating a spike after a certain delay. This delay is inversely related to the stimulus intensity—the higher the input signal, the shorter the delay before the neuron fires its first spike. The neuron produces only a single spike within the encoding window, and the time of this spike relative to the stimulus onset carries the entire information about the stimulus magnitude. Additionally, a dynamic threshold that decays exponentially is applied. The higher the input value, the more information it represents, leading to an earlier spike. To process the input, it is often normalized by dividing it by the maximum input value. The threshold over time is then modeled by an exponential function, which can be expressed as $Threshold(t) = \theta * e^{-t/\tau_{th}}$, where θ is a threshold constant and usually set to 1, and τ_{th} is a time constant which is

defined by the user. A spike is generated when the input is larger than the calculated threshold, and more spikes are discouraged as shown in Figure 3.3.

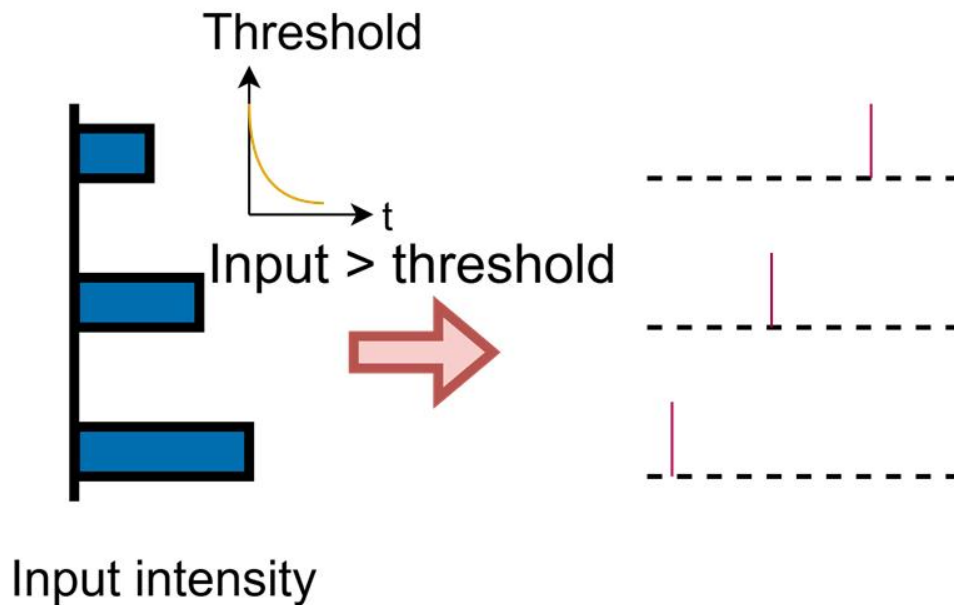


Figure 3.3: TTFs encoding demonstration

The existing TTFs encoder aims to generate one spike where the latency of the spike depends upon the input signal amplitude at a discrete time-step. The higher the amplitude, earlier the spike is generated. The input signal data is compared to an exponentially decaying threshold by using a combination of a timing counter output, a ROM-based memory table storing the exponential threshold values, and a comparator as illustrated in Figure 3.4. Whenever a spike is generated at a certain latency, an inhibition mechanism is activated through another comparator against the timing counter to ensure that no more spikes are generated for the same data sample.

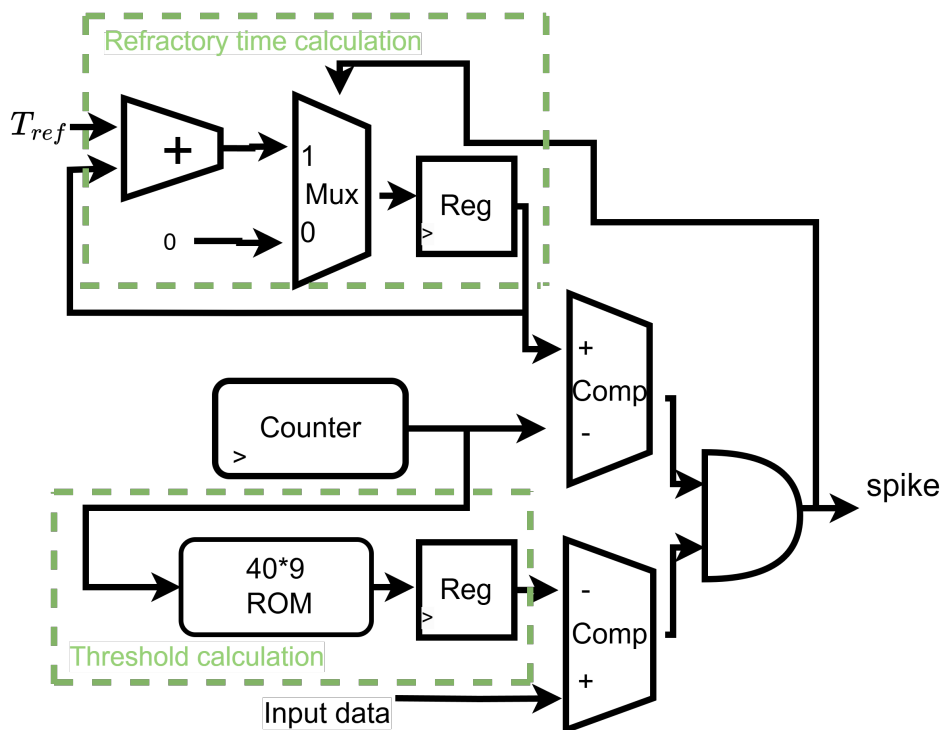


Figure 3.4: State-of-art digital hardware TTFS encoder [2]

3.3 Inter-Spike-Interval Encoding

In the realm of neuroscience and artificial neural networks, the efficient representation and transmission of information is a critical challenge. One approach to tackling this challenge is through spike-based encoding mechanisms, which are inspired by the way biological neurons communicate. Among these mechanisms, Inter-Spike Interval (ISI) encoding stands out as a powerful algorithm compare with rate and TTFS encoding for translating information into the timing patterns of neural spikes. By focusing on the time intervals between successive spikes, ISI encoding captures dynamic and temporal features of stimuli in a way that mimics the brain's neural coding strategies.

The concept of ISI encoding is grounded in the observation that neurons convey information not only through the rate of spiking activity but also through the precise timing of spikes.

This method emphasizes the temporal structure of neural signals, which is crucial for tasks that require high precision, such as sensory processing, motor control, and cognitive functions. Unlike traditional rate coding, where information is encoded in the average firing rate of neurons which is number of spikes (N_s), ISI encoding focuses on the intervals between individual spikes which is Inter-Spike Interval (ISI) as illustrated in Figure 3.5 , offering a more detailed and efficient way to represent data. For more detailed, the input intensity is usually normalized in the range from 0 to 1. The following equations are used to generate the ISI encoded spikes from each discrete signal amplitude (A) value:

The equation for $N_s(A)$ is given by:

$$N_s(A) = \lceil N_{\max}A \rceil \quad (3.1)$$

where N_{\max} is a constant parameter as maximum number of spikes allowed for the encoder, and $\lceil \cdot \rceil$ represents the ceiling function.

The conversion of ISI is given by:

$$\text{ISI}(A) = \begin{cases} \lceil -(T_{\max} - T_{\min})A + T_{\max} \rceil, & N_s(A) > 1, \\ T_{\max}, & \text{otherwise.} \end{cases} \quad (3.2)$$

Here, T_{\max} and T_{\min} are the maximum and minimum time intervals permitted between two spikes within the spike train and are constant parameters. In the existing ISI hardware encoder, all the constant parameters are defined as terms of 2^n where n is an integer for easier multiplication through shift operation in the hardware. The spike burst is generated through calculating the $N_s(A)$ for a specific signal data of A with shifting, addition and truncation operation using equation 3.1 which if higher than 1 for a specific amplitude value is then used to calculate equation 3.2 for ISI interval. A set of shifters and adders are used

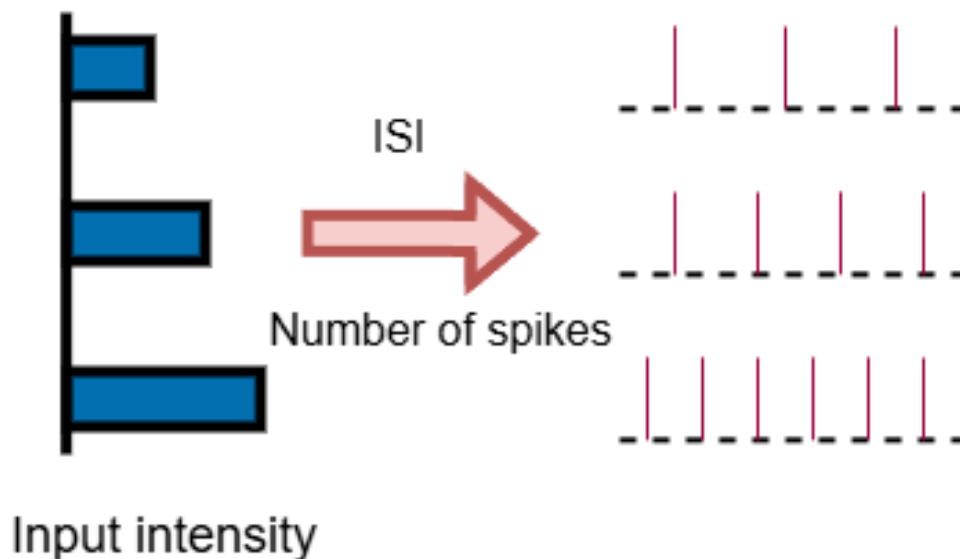


Figure 3.5: ISI encoding demonstration

to calculate the ISI for the specific A and then a timing counter is used to ensure that ISI interval is maintained between every two spikes in the encoded spike train for that specific input sample as shown in Figure 3.6. A spike counter is also used to ensure that the total generated spikes in the spike train are limited within the calculated parameter $N_s(A)$ for a specific data sample A . Thus, the existing ISI encoder design functions for converting each discrete amplitude (A) value of the signal to a burst set of spike trains.

3.4 Phase Encoding

In phase encoding, information is encoded in the relative phase differences of oscillatory signals. This approach draws inspiration from observations of biological systems, where neurons synchronize with oscillatory rhythms to process information [68]. Thus, this approach uses an external periodic oscillation signal as a reference and spike will be fired depending on the phase shift of the oscillation signal [69, 70]. In [71], a simple phase encoding method

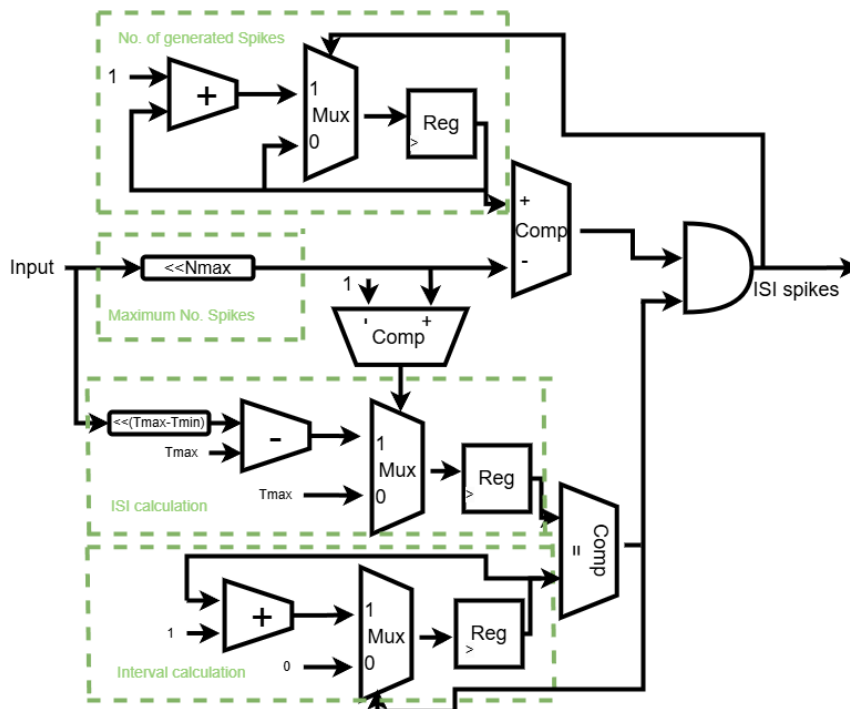


Figure 3.6: State-of-art digital hardware ISI encoder [2]

is proposed, where input intensity values are converted into their binary representation. In this approach, each bit with a value of 1 corresponds to a spike, as illustrated in Figure 3.7. The information is encoded by assigning distinct weights to each bit, with these weights varying periodically which can be expressed as equation 3.3.

$$W(t) = 2^{-(1+\text{mod}(t-1,n))} \quad (3.3)$$

Where t represents the number of time steps, and n denotes the number of phases within one period of the oscillation signal. The number of phases is determined by maximum input intensity. The larger the input intensity, the more significant spikes it fires, and the more information it carries. The advanced digital hardware design for phase encoding is both efficient and straightforward. It relies solely on multiplexers and n -bit registers to produce a periodic binary representation of the input signal, as illustrated in Figure 3.8.

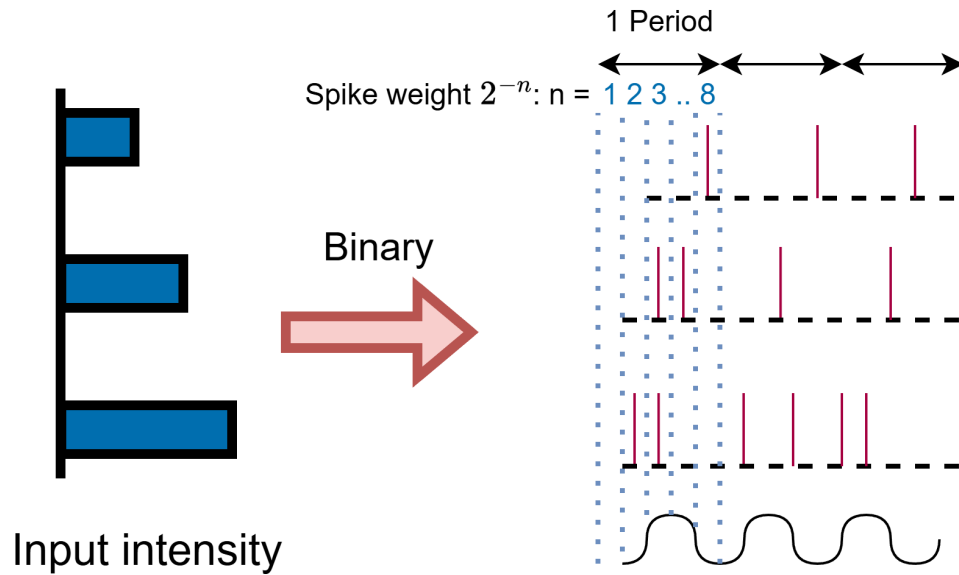


Figure 3.7: Illustration of phase encoding

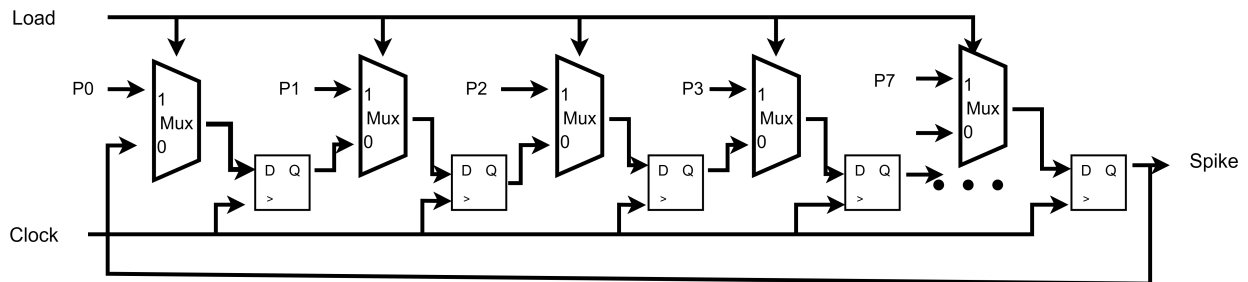


Figure 3.8: State-of-art digital hardware phase encoder [2]

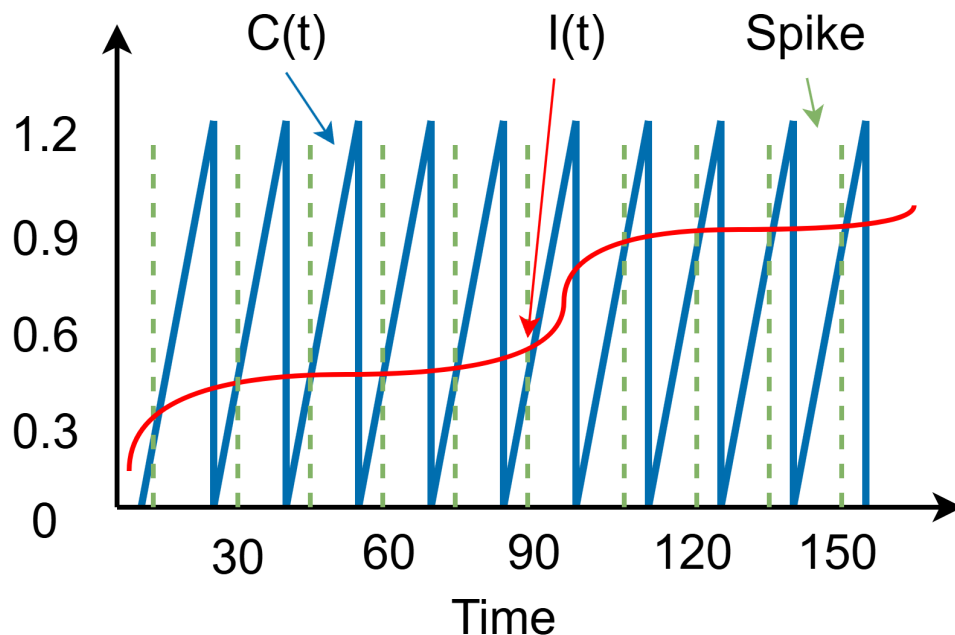


Figure 3.9: Illustration of PWM-based phase encoding

Recent researchers also proposed a robust and efficient phase encoding which uses Pulsewidth Modulation (PWM) principles to encode the analog data into spikes due to its simplified hardware implementation, adaptability in parameters and precise reconstruction of analog signal compared with traditional phase encoding [72]. Figure 3.9 demonstrates this encoding algorithm, which triggers a spike whenever the carrier signal $C(t)$ surpasses the input signal $I(t)$. The effectiveness of this approach in conveying information depends significantly on the resolution of the carrier signal, defined as its ability to detect subtle variations in the input. Higher resolution enables more accurate signal reconstruction.

3.5 Multiplexing Encoding

Recent advances in neuroscience have also revealed a unique encoding strategy within biological neural systems. First introduced in [73], this approach integrates multiple encoding

methods that operate on varying timescales, enhancing the system's data processing capabilities. Referred to as multiplexed encoding, it combines different encoding schemes to capture complementary temporal and spatial input features, thereby increasing data capacity and improving noise resilience [74].

In [75], researchers evaluated the information density and noise resilience of various encoding schemes. Among them, multiplexed encoding demonstrated the highest data capacity and robustness compared to rate and temporal encoding methods. Multiplexing encoding consists of two primary steps [76]. The first step involves encoding the input signal into spikes using either rate-based or temporal encoding. In the second step, known as the transformation phase, the generated spikes are aligned with the next local maximum of an external reference signal, as illustrated in Figure 3.10. In TTFS-phase encoding (Figure 3.10a), each spike is shifted to the subsequent local maximum of a sinusoidal reference signal. If multiple spikes coincide at the same local maximum, they are compressed into a single spike. Similarly, in ISI-phase encoding (Figure 3.10b), spikes are adjusted to align with the next local maximum of the reference signal. In [45], researchers introduced the implementation of TTFS-phase and ISI-phase encoding techniques using analog integrated circuits with subthreshold membrane oscillation(SMO) as a sinusoidal reference signal.

However, the implementation of sinusoid signals in digital circuits is not easy. Generating precise sinusoidal waveforms in digital circuits is challenging due to the need for digital-to-analog conversion and filtering, which add complexity and cost. The sinusoid signals are also more prone to noise and quantization errors, compromising signal quality. Additionally, The inherent continuity of sinusoidal waveforms results in reduced efficiency and increased power dissipation, as generating such signals necessitates gradual transitions and continuous output variation. This approach introduces intermediate states between fully on and off, failing to leverage the binary nature of digital circuits optimized for discrete switching. In [72], a

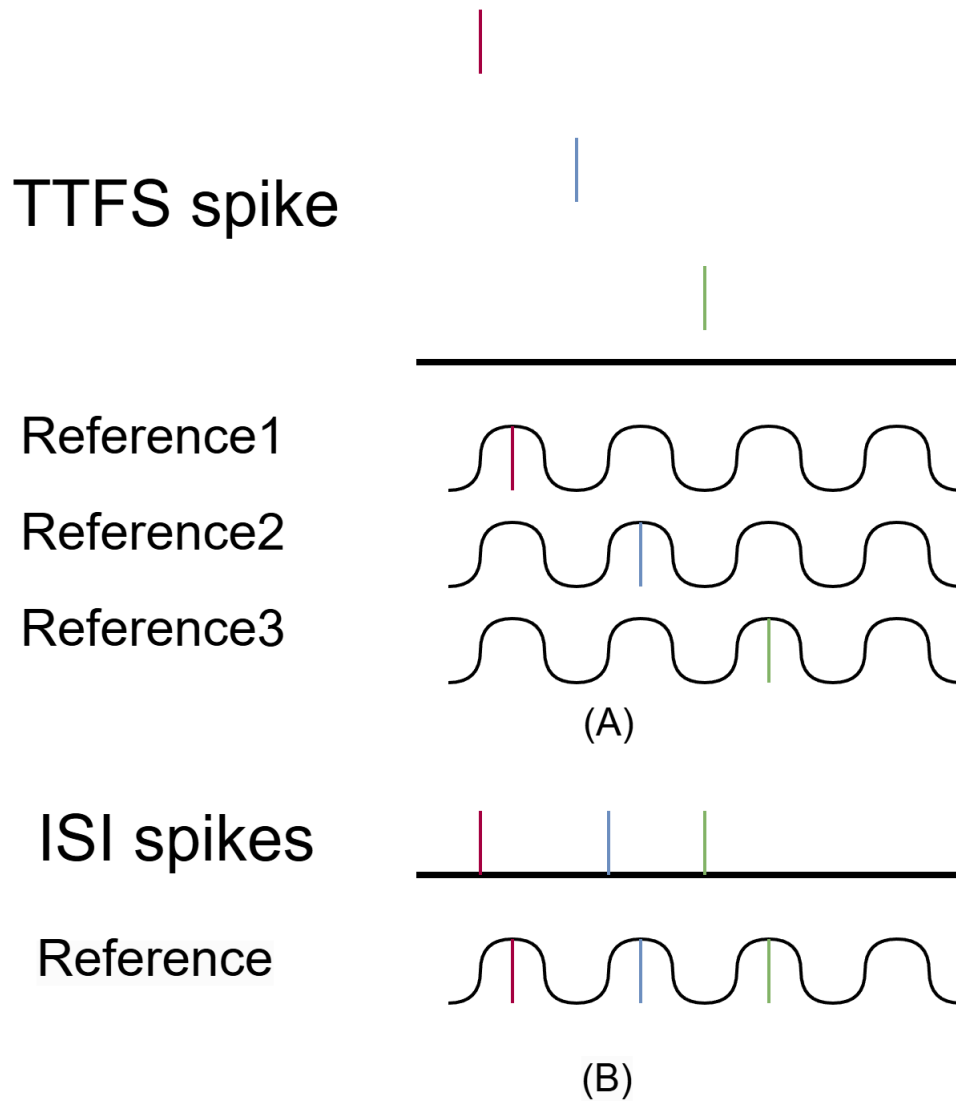


Figure 3.10: Illustration of multiplexing encoding. (a) TTFS-phase encoding. (b) ISI-phase encoding

novel phase encoding method, developed using Pulse Width Modulation (PWM) signaling, is introduced to tackle these challenges, offering a solution optimized for efficient hardware integration.

Thus, we introduce a novel multiplexing encoder that combines an Inter-Spike Interval (ISI) or burst encoding with a PWM-based phase encoding which will be demonstrated in a later section, and is easy and straightforward to implement using digital circuits while offering enhanced control and efficiency.

Chapter 4

Architecture of Proposed Digital Encoder

4.1 Proposed Digital Encoder

4.1.1 Proposed Digital Rate Encoder

All the computations are performed using 16-bit fixed-point arithmetic to balance accuracy and cost. The rate encoding module is implemented with an Xor gate, two pseudo-random number generators, and a comparator. Two random number generators are implemented with 16-bit Fibonacci and Galois LFSRs separately as shown in Figure 4.1. The limitation of existing hardware rate encoders arises from their deterministic and predictable nature, which can negatively impact the accuracy and robustness of rate encoding in certain applications. Thus, the proposed hardware rate encoder targets to mitigate those problems by combining two types of LFSRs which reduces predictability and increases the period of the random number sequence. Furthermore, reduces the correlation between two LFSRs and thus improves the quality of the randomness which potentially increases the ability to carry information. The random number will be generated every 16 clock cycles and then compared with input data. The spike will be fired only if the input data is larger than the random number.

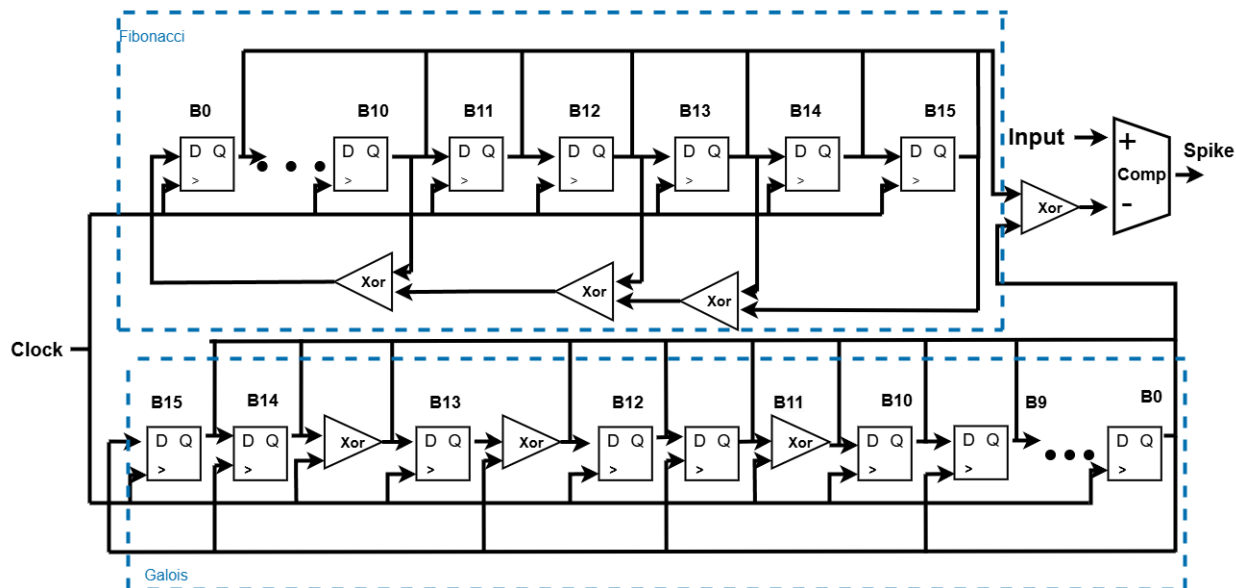


Figure 4.1: Hardware implementation of the proposed rate encoder.

4.1.2 Proposed Digital TTFS encoder

As shown in section 3.2, the implementation of the exponentially decaying threshold is the most important component of the TTFS encoder. The precision will largely affect the timing of firing a spike.

The SoTA TTFS hardware encoder typically utilizes memory tables (ROMs) to store pre-computed exponential function values, owing to their straightforward and rapid implementation. However, it usually demands significant memory space, especially for high precision or wide input ranges, which can consume considerable hardware resources. Additionally, this method lacks flexibility, as increasing the precision or range means increasing the ROM size, which may lead to higher area and power usage. More importantly, the latency may increase due to address decoding and longer memory access time as ROM size grows.

While SoTA architecture uses 40x16 ROM-based memory tables for storing pre-calculated exponential values in the TTFS encoder, it becomes less practical for high-resolution, wide-

range, or power-sensitive applications. Thus, an exponential approximate module is introduced to the TTFS hardware encoder for the first time to address the above problems.

In [77], an exponential function can be transformed into a base-2 function, an approach that only takes negative input has been utilized and expressed in equation 4.1:

$$\text{Exp}(x) \approx 2^{x+(x \cdot 2^{-1})-(x \cdot 2^{-4})} \approx 2^{1.4375x} \quad (4.1)$$

In our study, the input x is the time within a sampling window which is always positive, thus evaluating the sign bit is unnecessary. Therefore, a refined exponential approximation module has been developed exclusively for positive inputs in our study, achieving even fewer hardware resources compared to the exponential approximation model detailed in [77]. Upon processing the input, a scaled value of $-1.4375X$ is computed through shift-and-add operations, with both the integer and fractional segments then stored in a designated register. The integer segment is analyzed to determine the count of 'true' bits, while simultaneously, the fractional part is appended with a leading '1' and followed by thirteen '0' bits. This combined result is subsequently right-shifted according to the identified true bit count as demonstrated in Figure ??.

Our proposed TTFS encoder is implemented in two parts as shown in Figure ?. One is used to generate the time-decaying threshold and compare it with input data, while the other part is used to generate the refractory period which is the time that prevents the firing of the second spike after firing the first spike. The counter serves as a time calculation and input for the exponential approximate module. The encoding process requires a total of 40 clock cycles to complete.

4.1.3 Proposed Digital Multiplexing Encoder

As illustrated in section 3.5, Multiplexing encoding demonstrates superior information processing capabilities, with evidence indicating that its robustness against noise surpasses all other encoding algorithms. This advantage is attributed to the use of complementary temporal patterns, which enable the transmission of greater amounts of information at the same sampling frequency, thereby improving system performance [76]. Hence, the incorporation of SMO as a sinusoidal signal within the multiplexing encoding framework contributes to enhancing system performance and robustness. The multiplexing process involves two primary stages: the encoding phase and the transformation phase. During encoding, analog signals are converted into spikes, which may manifest as a single spike when latency encoding is utilized or as a spike train in the case of ISI encoding. The transformation phase, also known as gamma alignment, then shifts the latency spike or ISI spike train to the subsequent local peak of the SMO signal.

However, the implementation of sinusoid signals in digital circuits is not easy. Generating precise sinusoidal waveforms in digital circuits is challenging due to the need for digital-to-analog conversion and filtering, which add complexity and cost. The sinusoid signals are also more prone to noise and quantization errors, compromising signal quality. Additionally, The inherent continuity of sinusoidal waveforms results in reduced efficiency and increased power dissipation, as generating such signals necessitates gradual transitions and continuous output variation. This approach introduces intermediate states between fully on and off, failing to leverage the binary nature of digital circuits optimized for discrete switching. In [72], a novel phase encoding method, developed using Pulse Width Modulation (PWM) signaling, is introduced to tackle these challenges, offering a solution optimized for efficient hardware integration.

The proposed multiplexing encoder consists of three primary components as shown in Figure 4.4 part a. The first component generates the ISI spike train within a specified sampling window which is similar to the SoTA ISI hardware architecture. The second component, incorporating a NOT gate and an AND gate, identifies the rising edge of the PWM signal, which can be produced by the clock signal. Both the ISI spike train and PWM signal use 16-bit wide space. The third component which is called PWM alignment as demonstrated in Figure 4.3, aligns the ISI spike train with the detected PWM signal rising edge. The architecture of the alignment process is shown in Figure 4.4 part b by using some left shifter, AND and OR logic gates. In each clock cycle, the ISI spike train and the PWM alignment signal will be left-shifted to facilitate spike detection. Upon identifying an ISI spike, it will be synchronized with the subsequent rising edge of the PWM signal. It takes 34 clock cycles to complete the whole encoding process for one input sample. The incorporation of PWM in multiplexed encoding enhances data capacity and noise resilience while also providing ease of implementation within digital circuits.

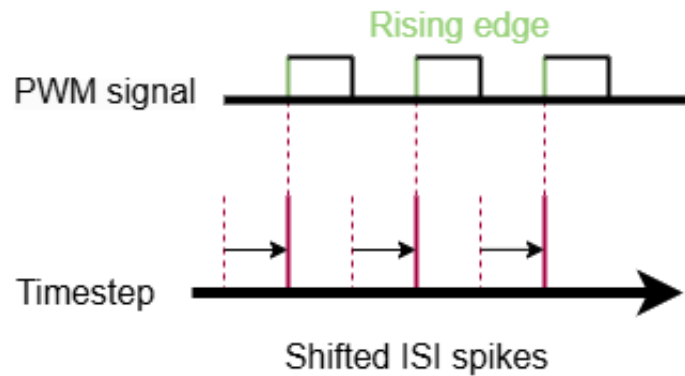
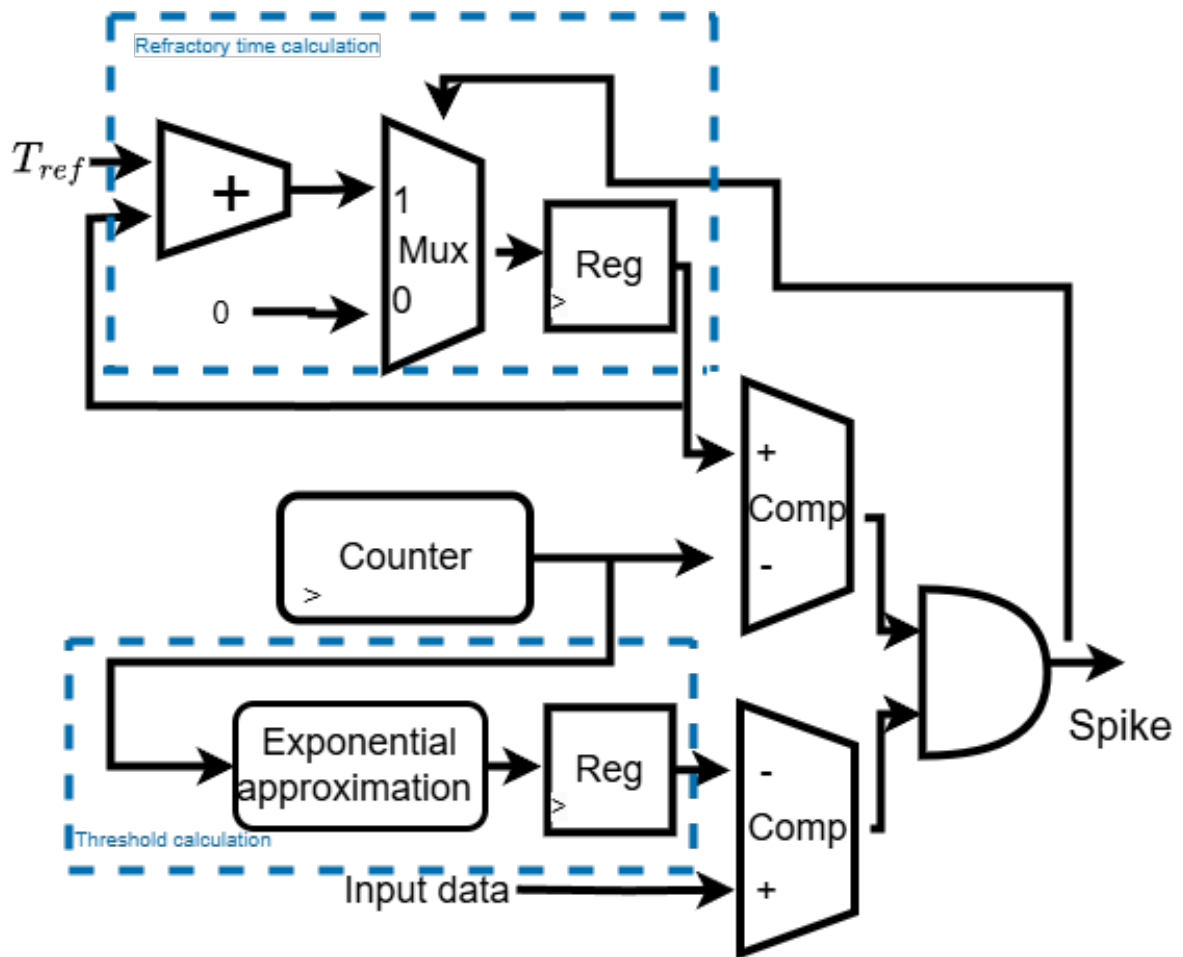
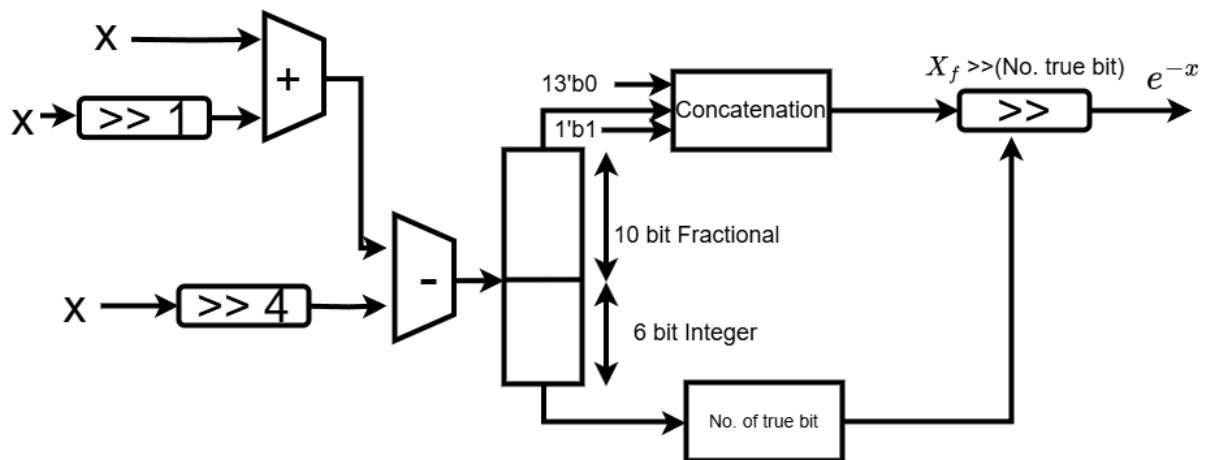


Figure 4.3: Demonstration of PWM alignment



(a) Architecture of the proposed TTFS encoder



(b) Proposed exponential approximation on TTFS encoder

Figure 4.2: Hardware implementation of the proposed TTFS encoder

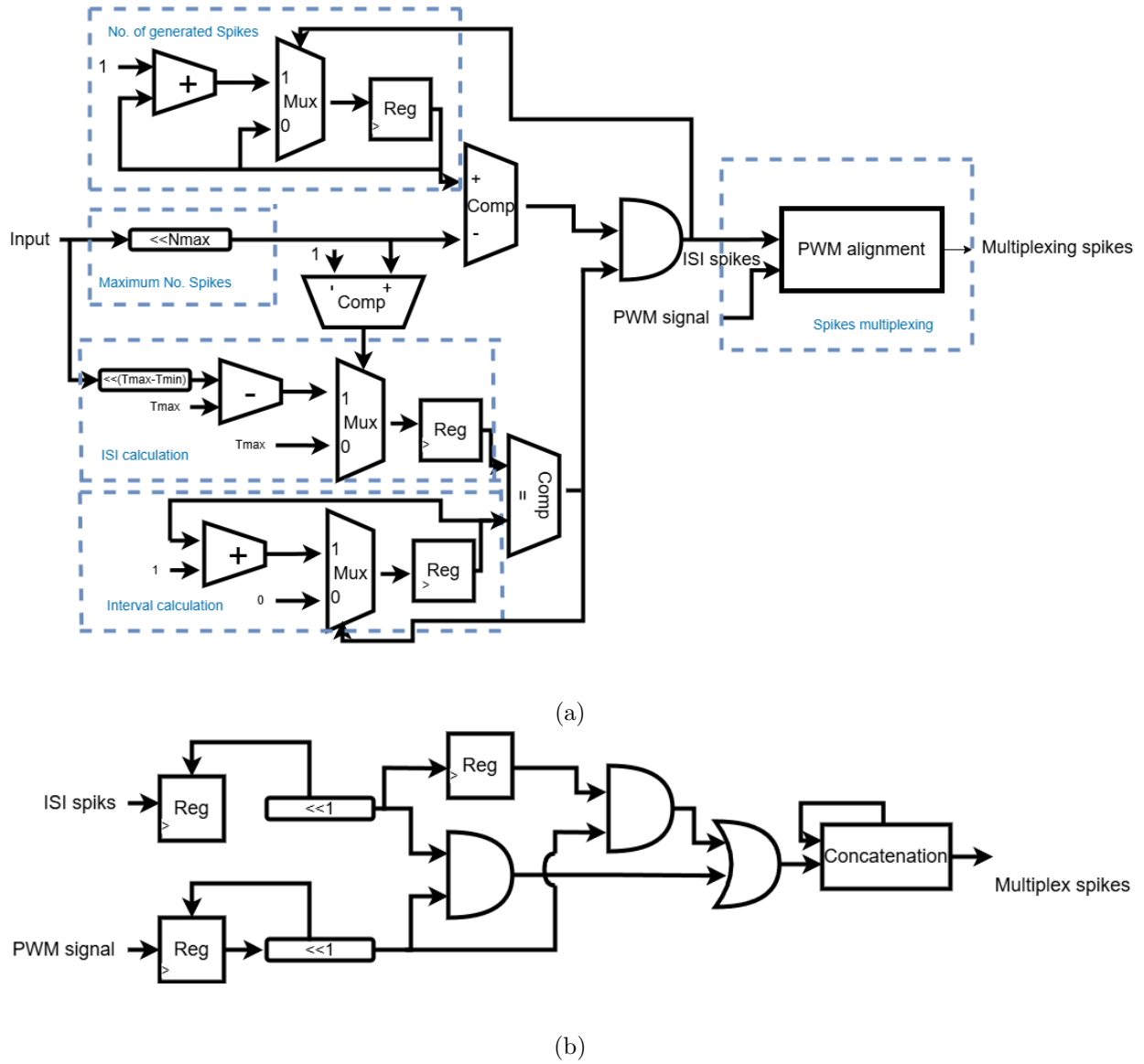


Figure 4.4: (a) Hardware implementation of the proposed multiplexing encoder (b) Proposed PWM alignment module

Chapter 5

Experimental Setup and Results

In this work, all the encoding algorithms are evaluated on the following aspects such as encoding speed, resource consumption, and accuracy. The encoders are implemented on Xilinx ZedBoard Zynq-7000 development board, operating at a frequency of 100 MHz. The speed, resource consumption and the accuracy of encoders are measured by using the input as shown in [5.1](#).

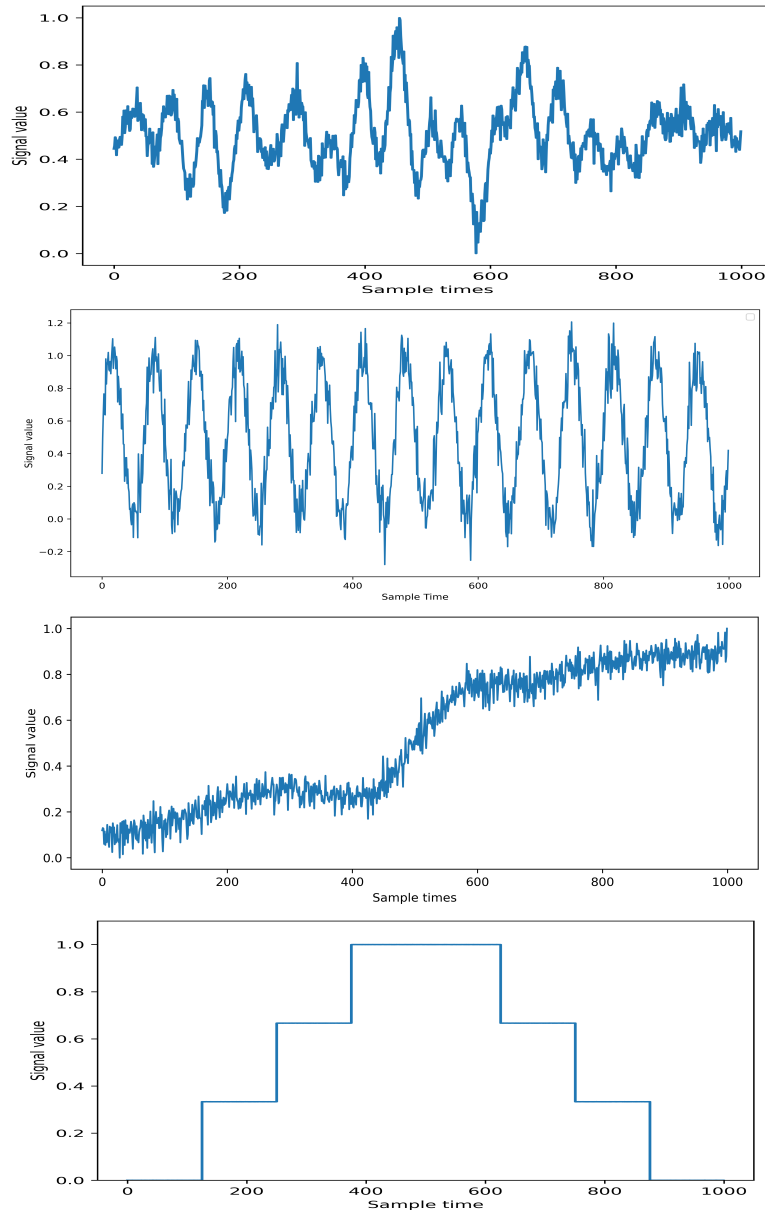


Figure 5.1: Testing signals used for analysis. (a) Sine component with random power and phase lag with white noise (b) 3Hz smooth sinusoidal signal with white noise (SS) (c) Constantly rising and falling signal (CRF) (d) Step-wise signal(SW)

1. Encoding speed: The operational frequency of spike encoding algorithms plays a crucial role in determining the minimum processing speed required for spiking neural networks (SNNs). If the frequency at which these algorithms function on the FPGA is lower

than the processing speed of the SNN itself, the entire system’s performance will be bottlenecked, leading to slower computation and preventing the SNN from achieving its full potential. To support the high-speed demands of SNN applications, it is essential that the spike encoding operates at a sufficiently high frequency. This work, therefore, assesses and compares the maximum achievable frequency of these algorithms to ensure the system performs optimally.

2. Resource consumption: The resource consumption of the spike encoder significantly affects the maximum possible size of spiking neural networks (SNNs) that can be implemented on an FPGA. In this study, the primary resource considered is logic element, which are critical components for FPGA operation. Efficient utilization of these resources is essential to maximize the capacity and performance of SNNs on the hardware platform.
3. Accuracy: To assess the accuracy of the encoder, we compare the original signal with the reconstructed signal using several key metrics: Root Mean Square Error (RMSE), Signal-to-Noise Ratio (SNR), and average Absolute Error (aAE). These metrics quantify the differences between the two signals, providing a clear measure of the encoder’s performance. The primary goal is to minimize these differences, ensuring that the reconstructed signal closely matches the original, thus improving the overall fidelity of the encoding process.

$$RMSE = \sqrt{\frac{\sum_{t=1}^N (S_{ori}(t) - S_{rec}(t))^2}{N}} \quad (5.1)$$

$$SNR = 20 \cdot \log \frac{Power_{S_{ori}}}{Power_{S_{ori}-S_{rec}}} [dB] \quad (5.2)$$

$$aAE = average(|S_{ori}(t) - S_{rec}(t)|) \quad (5.3)$$

where S_{ori} is the original signal and S_{rec} is the reconstructed signal. The power of a signal is determined by calculating the average value of the signal's squared amplitude. $|\cdot|$ represents the absolute value, and $average(\cdot)$ represent the mean value. Minimizing the difference between the original and reconstructed signal serves as the main objective of this study.

To assess signal reconstruction accuracy, the original input signal is first encoded into spikes and then those generated spike trains are used through software to reconstruct the original signal from it. After that, the original input signal is compared with the reconstructed signal using their respective decoding algorithms in software. In the case of the signal reconstruction in rate encoding, the method involves counting spikes over a defined sampling interval. For the TTFS encoding, reconstruction is achieved by identifying the timing of the initial spike and translating it into a corresponding signal value based on an exponential decay function over a designated time window. For burst and multiplexing encoders, the reconstruction procedure utilizes the ISI calculation as specified in Equation 3.2, where T_{max} and T_{min} represent the two governing time constants.

Four distinct test signals, each containing 1,000 samples, are utilized as illustrated in Figure 5.1. These signals are sourced from various SNNs representing different functionalities [78, 79, 80, 81], allowing for a comprehensive evaluation that extends beyond a single functional type. Each sample of a given test signal is sequentially processed by the encoders. In rate encoding, 16 rate encoder modules operate in parallel, producing a 16-bit wide spike train over 16 clock cycles. For TTFS encoding, the encoding process for a single sample requires 40 clock cycles. Burst encoding completes the encoding of one sample in 17 clock cycles, while multiplexing encoding requires 34 clock cycles per sample.

Encoder	RMSE	SNR	aAE	Testing Signal
Existing rate	0.348	9.82	0.2818	SS
	0.3273	10.74	0.2693	NS
	0.334	10.46	0.2808	CRF
	0.3574	9.66	0.2808	SW
Proposed rate	0.0845	34.424	0.0567	SS
	0.1162	28.7426	0.093	NS
	0.1074	30.202	0.083	CRF
	0.1367	26.3585	0.0797	SW

Table 5.1: Accuracy comparison for rate encoder

Encoder	RMSE	SNR	aAE	Testing Signal
Existing TTFS	0.4841	4.096	0.382	SS
	0.501	3.341	0.464	NS
	0.495	3.652	0.429	CRF
	0.496	3.973	0.389	SW
Proposed TTFS	0.217	18.024	0.166	SS
	0.201	19.199	0.164	NS
	0.192	20.097	0.162	CRF
	0.222	17.922	0.162	SW

Table 5.2: Accuracy comparison for TTFS encoder

Encoder	RMSE	SNR	aAE	Testing Signal
Existing ISI	0.0858	34.154	0.071	SS
	0.0737	36.639	0.0636	NS
	0.0828	34.7257	0.0712	CRF
	0.0779	36.131	0.0625	SW
Proposed Multiplexing	0.0764	36.1533	0.064	SS
	0.0737	36.639	0.0636	NS
	0.0753	36.3729	0.0663	CRF
	0.0779	36.131	0.0625	SW

Table 5.3: Accuracy comparison for multiplexing encoder

Encoder	LUT	FF	Power(mW)	Fmax(MHz)
Existing rate	208	608	117	372.717
Proposed rate	344	627	118	356.761
Existing TTFS	312	19	106	174.825
Proposed TTFS	70	72	107	244.08
Existing ISI	52	77	106	190.694
Proposed multiplexing	64	127	106	197.355

Table 5.4: Resource utilization, Power consumption and Fmax for all the encoders

In this study, we define optimal parameters as those that maximize the signal-to-noise ratio (SNR) of the reconstructed signal in relation to the original signal. To ensure consistency between hardware and software implementations, we standardize the bit width to 16 for all arithmetic and logic operation in digital circuit, striking a balance between performance and resource cost in hardware.

The performance evaluation for all the encoding algorithms is shown in Tables 5.1, 5.2, and 5.3. The proposed architecture consistently outperforms the existing methods, evidenced by notably lower RMSE and aAE values across all testing signals, suggesting improved perfor-

mance and error reduction. Additionally, the SNR values for the proposed architecture are significantly higher, highlighting enhanced signal quality and robustness against noise. This improvement across multiple metrics underscores the efficacy of the proposed architecture in delivering superior performance for signal encoding under diverse testing conditions. Moreover, we also compare the resource and power consumption, and maximum operating speed in terms of FPGA implementation.

However, a balance between performance and hardware area is often necessary in digital design. Enhanced performance typically entails increased logic utilization or reduced speed, particularly in rate and multiplexing encoders. In Table 5.4, The proposed rate encoder shows a modest increase in LUT usage and a slight reduction in maximum operating speed. The proposed multiplexing encoder has a 3.49% improvement in running speed at the cost of utilizing more LUTs and FFs compared with the existing ISI encoder. Nevertheless, a notable exception is observed with the proposed TTFS encoder, which achieves a substantial improvement by reducing LUTs and FFs usage by approximately 57% and increasing speed by around 39.6%. This is accomplished by replacing the 40x16 ROMs with an optimized 16-bit exponential approximation module, representing a significant advancement in hardware efficiency. The power consumption remains comparable among all the existing and proposed design. Higher logic utilization leads to increased power consumption in the design.

Chapter 6

Discussion

Spiking Neural Networks(SNNs) represent information with brief events—spikes—rather than continuous values. Neurons integrate incoming signals over time and fire only when a threshold is reached. Because computation occurs only on activity, this event-driven style avoids needless updates and maps naturally to hardware that sleeps between events.

The attraction is largely practical. In conventional networks, energy is dominated by shuttling data to and from memory. Spiking systems cut this cost by keeping computation close to the stored state and remaining idle when nothing changes. This makes them well-suited to edge sensors and always-on applications with tight power budgets.

Effective spiking neurons must translate inputs into a membrane potential and decide when to fire. Timing matters as much as amplitude: information is encoded not just in how strong a signal is, but in when spikes occur. Time, therefore, becomes a first-class part of the representation.

This raises a central design question: how should real-valued measurements—images, audio, inertial readings—be converted into spikes before they reach the network? The conversion must retain task-relevant structure, keep spike counts low to save energy, and remain simple enough for efficient hardware implementation.

Spike encoding provides this interface from the physical world to spiking computation. Common strategies include rate coding (average spike frequency reflects intensity), Time-to-First-

Spike (TTFS) coding (stronger inputs spike earlier), Burst coding (emit spikes based on interval), and multiplexing coding (combine multiple encoding methods for higher information carrying ability and robustness). Each option trades precision, latency, energy, and circuit complexity differently.

Looking ahead, progress will hinge on: (1) encoding schemes that improve efficiency and robustness while remaining biologically inspired; (2) tighter integration of learning methods with encoders to boost end-to-end accuracy; (3) hardware-friendly designs for ASICs and FPGAs that meet real-time constraints; (4) unified encoders for multi-modal inputs such as vision, audio, and dynamic-vision sensor data; (5) adaptive encoders that adjust to environment and task; (6) greater transparency for debugging and trust; and (7) scalable methods that bring these gains to large models without sacrificing capability.

Chapter 7

Conclusions

For neural encoding, the challenge of converting real-world signals to spiking signals is a critical aspect of SNN applications. With the growing trend of implementing SNN on FPGA platforms, a new issue arises: selecting an efficient spike encoding algorithm and designing an optimized architecture for SNN deployment on FPGA. An effective spike encoding algorithm should ideally offer high signal reconstruction accuracy, fast computation, low resource consumption, and strong noise resilience. This study assesses and refines existing hardware rate encoding and TTFS encoding by introducing two innovative architectures to meet specified performance requirements in hardware SNN. Additionally, a novel multiplexing temporal encoder based on the ISI and PWM-based phase encoder is proposed for the first time, to the best of our knowledge, which shows the highest information-carrying capability and anti-noise ability compare with alternative spiking encoders. Furthermore, we present an efficient architecture for multiplexing temporal encoding implemented on FPGA, designed for seamless integration with neuromorphic hardware architectures, offering significant potential for future research applications. A diverse set of practical signals is used to test discussed encoders, providing insights into selecting the most suitable spike encoding algorithm and architecture. The results indicate that the multiplexing temporal encoding outperforms other state-of-the-art rate and temporal encoding algorithms in performance and noise robustness across the majority of test samples. It demonstrates a maximum signal reconstruction accuracy improvement of 9.95x across all testing signals compared with other

encoding algorithms. Additionally, the proposed rate encoding architectures outperform existing models, showing up to a 3.5x accuracy improvement on all test signals. Furthermore, the proposed TTFS architecture not only demonstrates up to a 5.5x accuracy enhancement over current methods but also reduces logic resource utilization by 57.1%. Thus, the results offer a comprehensive analysis of encoder selection and implementation for hardware neuromorphic accelerators.

Building on these findings, future work should establish common datasets and evaluation protocols to enable fair, reproducible comparisons across platforms and encoder families. Pairing the proposed encoders with on-chip learning (e.g., reward-modulated STDP) may improve adaptation to nonstationary inputs, while co-design with front-end quantization and fixed-point formats can better preserve informative signal structure. Extending the multiplexing encoder to multi-channel, multi-rate inputs and adding lightweight calibration for device variability would strengthen real-world readiness across biomedical, audio, and wireless applications. Finally, translating the FPGA design to ASIC and releasing open, reusable IP cores and reference implementations would create a shared foundation for the community and accelerate adoption in neuromorphic systems.

Bibliography

- [1] C. Neves, I. Gonzalez, J. Leander, and R. Karoumi, “A new approach to damage detection in bridges using machine learning,” 01 2018, pp. 73–84.
- [2] 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, p. 638474, 2021.
- [3] T. Yucek and H. Arslan, “A survey of spectrum sensing algorithms for cognitive radio applications,” *IEEE Communications Surveys Tutorials*, vol. 11, no. 1, pp. 116–130, 2009.
- [4] M. Wellens, A. de Baynast, and P. Mahonen, “Exploiting historical spectrum occupancy information for adaptive spectrum sensing,” in *2008 IEEE Wireless Communications and Networking Conference*, 2008, pp. 717–722.
- [5] K. Hamedani, L. Liu, S. Liu, H. He, and Y. Yi, “Deep spiking delayed feedback reservoirs and its application in spectrum sensing of mimo-ofdm dynamic spectrum sharing,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 1292–1299.
- [6] A. Rauniyar and S. Y. Shin, “Cooperative spectrum sensing based on adaptive activation of energy and preamble detector for cognitive radio networks,” *APSIPA Transactions on Signal and Information Processing*, vol. 7, p. e2, 2018.
- [7] A. Mehrabian and A. Zaimbashi, “Robust and blind eigenvalue-based multiantenna

- spectrum sensing under iq imbalance,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5581–5591, 2018.
- [8] R. Tandra and A. Sahai, “Snr walls for signal detection,” *IEEE Journal of selected topics in Signal Processing*, vol. 2, no. 1, pp. 4–17, 2008.
- [9] A. Momeni, B. Rahmani, M. Malléjac, P. del Hougne, and R. Fleury, “Backpropagation-free training of deep physical neural networks,” *Science*, vol. 382, no. 6676, pp. 1297–1303, 2023. [Online]. Available: <https://www.science.org/doi/abs/10.1126/science.adi8474>
- [10] A. Ghani, “Neuro-inspired speech recognition based on reservoir computing,” in *Advances in Speech Recognition*, N. Shabtai, Ed. Rijeka: IntechOpen, 2010, ch. 2. [Online]. Available: <https://doi.org/10.5772/10186>
- [11] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, “Isolated word recognition with the liquid state machine: a case study,” *Information Processing Letters*, vol. 95, no. 6, pp. 521–528, 2005, applications of Spiking Neural Networks. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020019005001523>
- [12] R. Gaurav, T. C. Stewart, and Y. Yi, “Reservoir based spiking models for univariate time series classification,” *Frontiers in Computational Neuroscience*, vol. 17, 2023. [Online]. Available: <https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2023.1148284>
- [13] R. Gaurav, T. C. Stewart, and Y. C. Yi, “Spiking reservoir computing for temporal edge intelligence on loihi,” in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, 2022, pp. 526–530.
- [14] M. R. Sarkar, S. Salekin Chowdhury, J. S. Walling, and C. Yang Yi, “Phase signal

- classification with reservoir computing,” in *2025 IEEE 34th Microelectronics Design Test Symposium (MDTS)*, 2025, pp. 1–6.
- [15] F. Nowshin, Y. Zhang, L. Liu, and Y. Yi, “Recent advances in reservoir computing with a focus on electronic reservoirs,” in *2020 11th International Green and Sustainable Computing Workshops (IGSC)*, 2020, pp. 1–8.
- [16] F. Nowshin, S. Sethi, Z. Dong, and Y. Yi, “Enhancing driving behavior analysis in autonomous systems: A reservoir computing and temporal-aware machine learning approach,” in *2024 IEEE International Conference on Mobility, Operations, Services and Technologies (MOST)*, 2024, pp. 24–33.
- [17] F. Nowshin, L. Liu, and Y. Yi, “Energy efficient and adaptive analog ic design for delay-based reservoir computing,” in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020, pp. 592–595.
- [18] F. Nowshin, Z. Dong, and Y. Yi, “Memory-augmented autoencoder with reservoir computing for edge-based anomaly detection in autonomous systems,” *IEEE Internet Computing*, pp. 1–10, 2025.
- [19] C. Lin, Y. Liang, and Y. Yi, “Fpga-based reservoir computing with optimized reservoir node architecture,” in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 1–6.
- [20] C. Lin, M. F. Azmine, Y. Liang, and Y. Yi, “Leveraging neuro-inspired AI accelerator for high-speed computing in 6G networks,” *Frontiers in Computational Neuroscience*, vol. 18, Feb. 2024, publisher: Frontiers. [Online]. Available: <https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2024.1345644/full>

- [21] C. Lin, M. F. Azmine, and Y. Yi, “Invited paper: Accelerating next-g wireless communications with fpga-based ai accelerators,” in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–8.
- [22] F. Nowshin, Y. Huang, M. R. Sarkar, Q. Xia, and Y. Yi, “Merrc: A memristor-enabled reconfigurable low-power reservoir computing architecture at the edge,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 1, pp. 174–186, 2024.
- [23] M. R. Sarkar and C. Y. Yi, “An in-memory computing architecture utilizing energy-efficient vgsot mram device,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 7, pp. 3258–3262, 2024.
- [24] M. R. Sarkar, S. S. Chowdhury, J. S. Walling, and C. Y. Yi, “An in-memory power efficient computing architecture with emerging vgsot mram device,” in *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2024, pp. 1–5.
- [25] M. R. Sarkar, M. M. A. Bappy, M. M. Azmir, D. M. Rashid, and S. I. Hasan, “Vg-sot mram design and performance analysis,” in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 0715–0719.
- [26] F. Nowshin and Y. Yi, “Memristor-based deep spiking neural network with a computing-in-memory architecture,” in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 1–6.
- [27] F. Nowshin, H. An, and Y. Yi, “Towards energy-efficient spiking neural networks: A robust hybrid cmos-memristive accelerator,” *J. Emerg. Technol. Comput. Syst.*, vol. 20, no. 1, Jan. 2024. [Online]. Available: <https://doi.org/10.1145/3635165>
- [28] F. Nowshin and Y. Yi, “ReRAM-Based Neuromorphic Computing,” in *Frontiers of*

- Quality Electronic Design (QED): AI, IoT and Hardware Security*, A. Iranmanesh, Ed. Cham: Springer International Publishing, 2023, pp. 43–65. [Online]. Available: https://doi.org/10.1007/978-3-031-16344-9_2
- [29] M. R. Sarkar, S. S. Chowdhury, G. S. Kumar, and C. Y. Yi, “Sarhd: A 6t-sram based side-channel attack resilient brain-inspired hyperdimensional computing architecture,” in *2025 IEEE 34th Microelectronics Design Test Symposium (MDTS)*, 2025, pp. 1–6.
- [30] R. Gaurav, B. Tripp, and A. Narayan, “Spiking approximations of the maxpooling operation in deep snns,” in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–8.
- [31] R. Gaurav, D. A. Do, T. T. Doan, and Y. Yi, “Dalton - deep local learning in snns via local weights and surrogate-derivative transfer,” *IEEE Transactions on Emerging Topics in Computing*, vol. 13, no. 3, pp. 578–590, 2025.
- [32] R. Gaurav, T. C. Stewart, and Y. Yi, “Legendre-SNN on Loihi-2: Evaluation and Insights,” Oct. 2024. [Online]. Available: <https://openreview.net/forum?id=wUUvWjdE0K>
- [33] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, “Spiking Neural Networks and Their Applications: A Review,” *Brain Sciences*, vol. 12, no. 7, p. 863, Jul. 2022, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: <https://www.mdpi.com/2076-3425/12/7/863>
- [34] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *Bulletin of Mathematical Biology*, vol. 52, no. 1, pp. 25–71, Jan. 1990. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0092824005800047>

- [35] W. Gerstner, “SPIKING NEURON MODELS Single Neurons , Populations , Plasticity,” 2002. [Online]. Available: <https://www.semanticscholar.org/paper/SPIKING-NEURON-MODELS-Single-Neurons-, -Populations-Gerstner/849f849f647bc8d49c025dc408f4e927e12bceec>
- [36] E. Izhikevich, “Simple model of spiking neurons,” *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.
- [37] M. Zhang, Z. Gu, and G. Pan, “A survey of neuromorphic computing based on spiking neural networks,” *Chinese Journal of Electronics*, vol. 27, no. 4, pp. 667–674, 2018.
- [38] H. Zheng and Y. Yi, “Spiking domain feature extraction with temporal dynamic learning,” in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, 2023, pp. 1–5.
- [39] R. Li, M. F. Azmine, G. Sharma, and Y. Yi, “Efficient digital architecture of spiking encoders for neuromorphic accelerators,” in *2025 26th International Symposium on Quality Electronic Design (ISQED)*, 2025, pp. 1–8.
- [40] R. Li, K. Bai, Y. Yi, R. Li, K. Bai, and Y. Yi, “Design and Optimization of Digital Neural Encoding for Neuromorphic Computing Systems,” in *Recent Advances in Neuromorphic Computing*. IntechOpen, May 2025. [Online]. Available: <https://www.intechopen.com/chapters/1191566>
- [41] H. Zheng, N. Mohammadi, K. Bai, and Y. Yi, “Low-power analog and mixed-signal ic design of multiplexing neural encoder in neuromorphic computing,” in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 154–159.
- [42] H. Zheng and Y. Yi, “Enhancing snn training performance: A mixed-signal triplet

- reconfigurable stdp circuit with multiplexing encoding,” in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.
- [43] H. Zheng and Y. C. Yi, “Spiking Neural Encoding and Hardware Implementations for Neuromorphic Computing,” in *Neuromorphic Computing*, Y. C. Yi and H. An, Eds. London: IntechOpen, 2023, section: 7. [Online]. Available: <https://doi.org/10.5772/intechopen.113050>
- [44] H. Zheng and Y. Yi, “Spiking neural encoding schemes and stdp training algorithms for edge computing,” in *Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing*, ser. SEC '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 365–371. [Online]. Available: <https://doi.org/10.1145/3583740.3626816>
- [45] H. Zheng, K. J. Bai, and Y. Yi, “Enabling a new methodology of neural coding: Multiplexing temporal encoding in neuromorphic computing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 3, pp. 331–342, 2023.
- [46] S. S. Chowdhury, D. Sharma, A. Kosta, and K. Roy, “Neuromorphic computing for robotic vision: algorithms to hardware advances,” *Communications Engineering*, vol. 4, no. 1, p. 152, Aug. 2025, publisher: Nature Publishing Group. [Online]. Available: <https://www.nature.com/articles/s44172-025-00492-5>
- [47] “Boosting AI with neuromorphic computing,” *Nature Computational Science*, vol. 5, no. 1, pp. 1–2, Jan. 2025, publisher: Nature Publishing Group. [Online]. Available: <https://www.nature.com/articles/s43588-025-00770-4>
- [48] Y. Liu, Z. Qin, Y. Zhu, and G. Y. Li, “SpikACom: A Neuromorphic Computing Framework for Green Communications,” Feb. 2025, arXiv:2502.17168 [eess]. [Online]. Available: <http://arxiv.org/abs/2502.17168>

- [49] H. Zheng, J. Xu, L. Liu, and Y. Yi, “Snnot: Spiking neural network with on-chip training for mimo-ofdm symbol detection,” *IEEE Transactions on Green Communications and Networking*, vol. 8, no. 4, pp. 1809–1823, 2024.
- [50] M. F. Azmine, R. Li, G. Sharma, and Y. Yi, “Spikespec: An on-chip learning neuromorphic accelerator for spectrum sensing with triplet-boosting and hardware friendly loss function,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 9, pp. 3423–3436, 2025.
- [51] M. F. Azmine, “An advanced neuromorphic accelerator on FPGA for next-G spectrum sensing,” Ph.D. dissertation, Virginia Tech, Apr. 2024. [Online]. Available: <https://hdl.handle.net/10919/119039>
- [52] G. Sharma, “Optimizing Reservoir Computing Architecture for Dynamic Spectrum Sensing Applications,” Ph.D. dissertation, Virginia Tech, Apr. 2024. [Online]. Available: <https://hdl.handle.net/10919/118671>
- [53] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, “Recent advances in physical reservoir computing: A review,” *Neural Networks*, vol. 115, pp. 100–123, Jul. 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608019300784>
- [54] R. Pascanu and H. Jaeger, “A neurodynamical model for working memory,” *Neural Networks*, vol. 24, no. 2, pp. 199–207, Mar. 2011. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608010001899>
- [55] B. J. Grzyb, E. Chinellato, G. M. Wojcik, and W. A. Kaminski, “Which model to use for the liquid state machine?” in *2009 International Joint Conference on Neural Networks*, 2009, pp. 1018–1024.

- [56] G. M. Wojcik and W. A. Kaminski, "Liquid state machine and its separation ability as function of electrical parameters of cell," *Neurocomputing*, vol. 70, no. 13, pp. 2593–2597, Aug. 2007. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231207000331>
- [57] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.
- [58] A. Rauniyar and S. Y. Shin, "Cooperative spectrum sensing based on adaptive activation of energy and preamble detector for cognitive radio networks," *APSIPA Transactions on Signal and Information Processing*, vol. 7, p. e2, 2018.
- [59] A. Mehrabian, M. Sabbaghian, and H. Yanikomeroglu, "Spectrum sensing for symmetric α -stable noise model with convolutional neural networks," *IEEE Transactions on Communications*, vol. 69, no. 8, pp. 5121–5135, 2021.
- [60] Q. Peng, A. Gilman, N. Vasconcelos, P. C. Cosman, and L. B. Milstein, "Robust deep sensing through transfer learning in cognitive radio," *IEEE Wireless Communications Letters*, vol. 9, no. 1, pp. 38–41, 2019.
- [61] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated stdp," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6178–6190, 2018.
- [62] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, "Neuro-inspired speech recognition with recurrent spiking neurons," in *Artificial Neural Networks-ICANN 2008: 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part I 18*. Springer, 2008, pp. 513–522.

- [63] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, “Isolated word recognition with the liquid state machine: a case study,” *Information Processing Letters*, vol. 95, no. 6, pp. 521–528, 2005.
- [64] H. Jang, O. Simeone, B. Gardner, and A. Grüning, “An introduction to spiking neural networks: Probabilistic models, learning rules, and applications [supplementary material],” 2019.
- [65] Y. Liu, S. S. Yenamachintala, and P. Li, “Energy-efficient fpga spiking neural accelerators with supervised and unsupervised spike-timing-dependent-plasticity,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 3, pp. 1–19, 2019.
- [66] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Benamoun, D. S. Jeong, and W. D. Lu, “Training spiking neural networks using lessons from deep learning,” *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023.
- [67] Y. Jin and P. Li, “Ap-stdp: A novel self-organizing mechanism for efficient reservoir computing,” in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 1158–1165.
- [68] M. A. Montemurro, M. J. Rasch, Y. Murayama, N. K. Logothetis, and S. Panzeri, “Phase-of-firing coding of natural visual stimuli in primary visual cortex,” *Current Biology*, vol. 18, no. 5, pp. 375–380, 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0960982208001681>
- [69] A. Cattani, G. T. Einevoll, and S. Panzeri, “Phase-of-firing code,” 2015. [Online]. Available: <https://arxiv.org/abs/1504.03954>
- [70] F. Walter, F. Röhrbein, and A. Knoll, “Computation by time,” *Neural Processing Letters*, vol. 44, 08 2016.

- [71] J. Kim, H. Kim, S. Huh, J. Lee, and K. Choi, “Deep neural networks with weighted spikes,” *Neurocomputing*, vol. 311, pp. 373–386, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231218306726>
- [72] A. Arriandiaga, E. Portillo, J. I. Espinosa-Ramos, and N. K. Kasabov, “Pulsewidth modulation-based algorithm for spike phase encoding and decoding of time-dependent analog data,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 3920–3931, 2020.
- [73] M. R. Rezaei, R. S. Fard, M. R. Popovic, S. A. Prescott, and M. Lankarany, “Synchrony-division neural multiplexing: An encoding model,” *Entropy*, vol. 25, 2021. [Online]. Available: <https://api.semanticscholar.org/CorpusID:240225435>
- [74] S. Panzeri, N. Brunel, N. K. Logothetis, and C. Kayser, “Sensory neural codes using multiplexed temporal scales,” *Trends in Neurosciences*, vol. 33, no. 3, pp. 111–120, 2010. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0166223609002008>
- [75] C. Kayser, M. Montemurro, N. Logothetis, and S. Panzeri, “Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns,” *Neuron*, vol. 61, pp. 597–608, 03 2009.
- [76] Z. Nadasdy, “Information encoding and reconstruction from the phase of action potentials,” *Frontiers in systems neuroscience*, vol. 3, p. 6, 02 2009.
- [77] S. Gomar and M. Ahmadi, “Digital realization of pstdp and tstdp learning,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–5.
- [78] C. D. Virgilio G., J. H. Sossa A., J. M. Antelis, and L. E. Falcón, “Spiking neural networks applied to the classification of motor tasks in eeg

- signals,” *Neural Networks*, vol. 122, pp. 130–143, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0893608019303193>
- [79] S. Zhao, Z. Li, R. Cui, Y. Kang, F. Sun, and R. Song, “Brain–machine interfacing-based teleoperation of multiple coordinated mobile robots,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5161–5170, 2017.
- [80] R. Miko, V. Steuber, and M. Schmuker, “Brain-inspired spiking neural network for gas-based navigation,” 04 2019.
- [81] B. Petro, N. Kasabov, and R. M. Kiss, “Selection and optimization of temporal spike encoding methods for spiking neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 358–370, 2020.