An advanced neuromorphic accelerator on FPGA for next-G spectrum sensing

Muhammad Farhan Azmine

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

Dr. Yang Yi, Chair
Dr. Dong Ha
Dr. Creed F. Jones

April 10, 2024
Blacksburg, Virginia

Keywords: Field Programmable Gate Array,Hardware accelerator, On-Chip learning,
Artificial intelligence, Recurrent Neural Network,
Spike-Time-Dependent-Plasticity,Cognitive-Radio-Network, Spectrum Sensing

An advanced neuromorphic accelerator on FPGA for next-G spectrum sensing

Muhammad Farhan Azmine

(ABSTRACT)

Spectrum sensing is one of the most significant tasks in next-Generation(G) Multiple-Input-Multiple-Output (MIMO) communication system which involves detecting and identifying occupancy or availability of frequency bands in radio spectrum. The conventional methods for spectrum sensing suffer from issues like hardware complexity and Signal-to-Noise-Ratio (SNR) wall that results in poor performance in high noise channel environment. Recurrent Neural Networks (RNN) like Liquid State Machines (LSM) are well-suited for designing energy efficient accelerators, needing fewer trainable parameters to capture temporal dependencies among the primary user frequency spectrums. This paper presents a novel LSM based FPGA accelerator equipped with comprehensive on-chip learning capabilities for spectrum sensing. Our accelerator employs Reward-based Spike Timing-Dependent Plasticity (R-STDP) to discern temporal correlations within the spectrum occupancy. While conventional R-STDP methods encounter convergence challenges during on-chip learning, our proposed solution mitigates this issue through an innovative hardware-friendly loss function. This mechanism ensures easier hardware implementation and demonstrates accelerated convergence with training period reduction of about 47% in classification of spectrum sensing. Additionally, we implement an energy efficient-asynchronous unsupervised Triplet-based STDP learning in the reservoir of the LSM accelerator to increase the accuracy of about 3.88% in high noise channel. After implementation, our proposed efficient LSM accelerator architecture achieved only 0.29% and 0.499% increase in Look-Up-Table (LUT)

and register resource utilization percentage as compared to the basic fixed reservoir LSM accelerator for spectrum sensing implemented on Virtex-707 FPGA.

An advanced neuromorphic accelerator on FPGA for next-G spectrum sensing

Muhammad Farhan Azmine

(GENERAL AUDIENCE ABSTRACT)

In modern communication systems, it's important to detect and use available radio frequencies effectively. However, current methods face challenges with complexity and noise interference. We've developed a new approach using advanced artificial intelligence (AI) based computing techniques to improve efficiency and accuracy in this process. Our method shows promising results, requiring only minimal additional resources in exchange of improved performance compared to older techniques.

Dedication

To my beloved parents, grandparents, my advisor, respective professors and the God Almighty

Acknowledgments

I wish to extend my sincere appreciation to my thesis advisor, Dr. Yang (Cindy) Yi, for her unwavering support and guidance throughout my research journey. I am also deeply grateful to my committee members, Dr. Dong Ha and Dr. Creed F. Jones for their invaluable contributions and constructive feedback for further improving my research. Furthermore, I would like to express my heartiest gratitude to my research colleagues Ruizhe Li and Gauri Sharma for their collaborative efforts and assistance towards this research. I am extremely thankful to the esteemed faculty members at Virginia Polytechnic Institute and State University for imparting their expertise and providing me with invaluable knowledge and experiences relevant to my research. Finally, I am indebted to my family members for their unwavering encouragement and support throughout this endeavor.

Contents

List of Figures ix

List of Tables xi

1 Introduction 1

1.1 Spectrum Sensing in Next-G communication 4

1.2 Cost Efficient Spike-timing based smart AI 6

1.3 Challenges of LSM design 9

1.3.1 Algorithmic Challenges 9

1.3.2 Hardware Challenges 11

2 Literature Review 12

3 Spiking neuron & Liquid State Machine 15

3.0.1 Biological Spiking Neuron 15

3.0.2 Action Potential 16

3.0.3 Computational Model 18

3.0.4 Liquid State Machine 22

3.0.5 Advantages for LSM network 23

4 Learning Algorithm in LSM neurons **26**

4.0.1 Triplet based unsupervised learning 26

5 Hardware friendly loss function **30**

6 Hardware Architecture **34**

6.0.1 Overall LSM Hardware Design 34

6.0.2 Implementation of Digital Neuron 35

6.0.3 Triplet STDP Digital Hardware Design 37

6.0.4 Supervised learning engine (SLE) 41

7 Experiment & Results **44**

7.1 EXPERIMENTAL SETUP AND EQUIPMENTS 44

7.2 Training settings and Results 44

7.2.1 Comparative Analysis - Accuracy 46

7.2.2 Overall Design Analysis 46

8 Summary **50**

Bibliography **51**

List of Figures

1.1 Spectrum Sensing architecture with hardware accelerator 4

3.1 Spectrum Sensing architecture with hardware accelerator 17

3.2 Spectrum Sensing architecture with hardware accelerator 19

3.3 Liquid State Machine with spike based computing 22

4.1 TSTDP Timing Differences for pre-post-pre spiking and post-pre-post spiking respectively 29

5.1 Loss function landscape with local and global minima. The red arrow directs toward higher saddle point, resulting in a non-optimized performance 32

6.1 An exemplary LSM hardware architecture with 16 reservoir neurons and 2 readout neurons. One reservoir neuron receives one external input spike and up to 3 recurrent spikes from the spike record register. During training, each reservoir neuron generates one spike and sends them to the readout layer. Each readout neuron receives all 16 spikes from the reservoir layer. W denotes the plastic synaptic weight, and each BRAM stores the corresponding weight array depending on the task. 35

6.2 Hardware implementation of a digital neuron in both the reservoir and readout layers. 35

6.3 Hardware implementation of the learning engine in a neuron. 36

6.4 Hardware implementation of Triplet-based STDP 38

6.5 Implementation of exponential approximation in Triplet-STDP 38

6.6 Implementation of Supervised Learning Engine (SLE) 42

List of Tables

6.1 Parameters-TSTDP 41

7.1 Parameter-TSTDP 45

7.2 Parameter-SLE 45

7.3 Weight Parameter Settings 45

7.4 Accuracy comparison at different SNR levels and epochs. 47

7.5 Resource Utilization Comparison 47

7.6 Power report 48

7.7 Inference Speed Comparison Table 48

7.8 Comparison of iterations with and without loss 48

List of Abbreviations

3GPP 3rd Generation Partnership Project

AI Artificial Intelligence

ANN Artificial Neural Network

AWGN Additive White Gaussian Noise

CFD Cyclostationary Feature Detection

CNN Convolution Neural Network

CRN Cognitive Radio Network

DFT Discrete Fourier Transform

FFT Fast Fourier Transform

FPGA Field Programmable Gate Array

LAN Local Area Network

LIF Leaky Integrate and Fire

LSM Liquid State Machine

MFD Matched Filter Detector

MIMO Multiple Input Multiple Output

ML Machine Learning

NPU Neural Processing Unit

OFDM Orthogonal Frequency Division Multiplexing

PISO Parallel Input Serial Output

PU Primary User

QDFR Quantized Delay Feedback Reservoir

QPSK Quadrature Phase Shift Keying

RC Reservoir Computing

RNG Random Number Generator

RNN Recurrent Neural Network

RSTDP Reward modulated Spike Timing Dependent Plasticity

SLC Sweeping Local Carrier

SNN Spiking Neural Network

SNR Signal-to-Noise-Ratio

SS Spectrum Sensing

STDP Spike Timing Dependent Plasticity

SU Secondary User

TSML Teacher Student Machine Learning Model

Chapter 1

Introduction

Data usage in wireless communication has been proliferated in recent years especially in sectors like Internet of things (IoT), vehicular communication and smart city applications.[44] This surge can be attributed to the introduction of the 3rd Generation Partnership Project (3GPP) standards in wireless systems. The 3GPP standard been a significant catalyst in improving the Quality of Service (QoS) so that seamless connectivity can be facilitated across various platforms. These improvements come with increasing demand of greater reliability, throughput and less latency.[7] One of the biggest challenges in improving the performance in wireless communication is limited resources of frequency spectrum bands to host increasing number of wireless service and applications. To mitigate this issue, the emphasis on spectrum sharing among different users is ever growing and it is being seen as a possible solution. Spectrum sharing means to allocate spectrum bands dynamically and efficiently to multiple users at the same time so that they can utilize the available resources more effectively. This approach enables efficient utilization of the frequency spectrum bands, thereby helps in alleviating the growing challenge of spectrum limitation.

One of the groundbreaking technology which can help tackle the spectrum scarcity and enable efficient sharing among different users is named as Cognitive Radio Network **(CRN)**. A Cognitive Radio Network (CRN) system enables an unlicensed secondary user within a network to utilize the unused white spaces of a primary user's (PU) spectrum, when the PU is not actively utilizing them. This intelligent mechanism in the SU units that can

continuously monitor the availability of spectrum from the PU units and detect the idle spectrums is known as Spectrum Sensing (SS). . Spectrum sensing allows CRNs to detect and identify unused or underutilized spectrum bands in real-time, thus enabling opportunistic access to the spectrum of PU without causing any interference to the user.

There are different existing techniques for spectrum sensing in CRN such as Energy Detection **(ED)**, cyclostationary feature detection **(CFD)** etc. Energy detection is one of the least complex technique that measures the energy level from the received signal of a certain PU frequency band and compares it against a predefined threshold.[22] If the energy level is higher than the threshold then the spectrum band is considered currently occupied. Otherwise,it is considered currently available for use of the SU user so that the SU can opportunistically transmit it's data using the available band from the PU without interference. Although the method is low complex to implement in the hardware, it suffers from Signal-to-Noise ratio **(SNR)** wall problem which hinders it's detection capability when noise is introduced to the channel.[21, 35, 52] The other existing methods such as CFD [23] are able to perform accurately in high noise condition but they are more resource complex to implement in hardware. Moreover, they require the SU user to learn about more PU transmitter parameters except only energy data such as carrier frequency, modulation type, guard band etc. In practical scenarios, obtaining such detailed information about PU transmitters can be challenging and often unfeasible, thereby limiting the real-world performance of these methods.

The emergence of Artificial Intelligence (AI) has inaugurated new interest among researchers to harness it's power for different wireless application. [19] Different hardware friendly AI models such as reservoir computing architectures have already been tried for solving traditional wireless problems such as MIMO-OFDM-symbol detection [2, 16, 17, 27, 28, 57, 58] and channel estimation problems [18]. AI has also made it possible to utilize it for **blind spectrum sensing**, especially depending on energy data only. Blind spectrum

sensing means the capability of the SU to identify and detect the white space spectrum of the PU without having any other predefined knowledge about PU transmitter or signal characteristics. AI models can extract the inherent characteristics of the limited input feature data and thus if trained on sufficient dataset, can successfully detect the vacant spectrum of PU user spectrum bands without any other specific characteristics from the PU transmitter except the energy signal data. The AI based detectors can be an effective approach for spectrum sensing for the scenarios where thee detailed information about PU user is limited. Deep learning techniques such as convolution neural networks (CNN) have already been proved more effective in spectrum sensing than the existing model based detection techniques. [36, 41] The CNN integrated techniques also showed better performance in low SNR signal data. [36, 37, 41] However, DNN models required large number of data to be trained which might not be available if the AI detector was to be trained during real-time operation. As a result, the models may demonstrate under-fitting characteristics and would not be effective for the target application. Moreover, DNN models are high computational demanding to be implemented for on-chip learning in the edge network devices. The spectrum sensing task is a time-series problem where temporal relation exists between frequency spectrum bands in past and present. Recurrent Neural Network (RNN) can be effective for time-series problems like this since it can capture the temporal dependency in the data through it's recurrent feedback connections. However, all types of RNN networks are not suitable for implementing on-chip learning in the low complex hardware edge devices. Learning methods that are non-local such as back-propagation require continuous data transfer at each iteration among the layers which brings up the memory wall bottleneck issue of Vonn-Neuman architecture where the limited memory bandwidth datapath reduces the operation time efficiency.

Liquid State Machine (LSM) is a specific type of RNN where the dynamics of a liquid medium serve as a reservoir,utilizing it's non-trainable fixed recurrent connections to transform time

varying input into spatio temporal patterns in order to accurately classify them through a low learning complexity based readout layer for spectrum detection task. In LSM, there are two special features for hardware friendliness: LSM exhibits two key features that make it hardware-friendly: Firstly, it operates on spike-based sparse computing, akin to the human brain. Secondly, LSM employs a local learning method, whereby gradient or weight data does not need to be transferred to the previous layer. This localized learning approach enhances efficiency and scalability, making LSM a compelling solution for real-world applications. However, it also comes up with it's own deficiency. Since LSM uses local learning, with only simple supervised learning mechanism it is hard to get high efficiency in detecting spectrums from high noise-channel data. As LSM has local learning systems and low-complex architecture, in this thesis, an advanced LSM accelerator with hardware friendly architecture and learning mechanism was implemented for spectrum sensing task.

1.1 Spectrum Sensing in Next-G communication

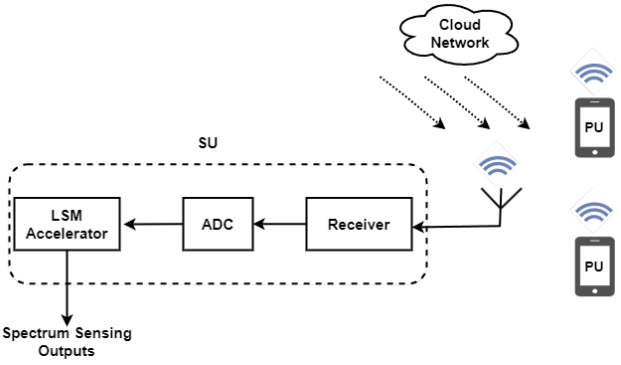

Figure 1.1: Spectrum Sensing architecture with hardware accelerator

For high speed and reliable data transmission in advanced-RF communication technology, Orthogonal frequency division multiplexing-Multiple Input Multiple Output (OFDM-MIMO) has become the best possible solution so far, especially for network architecture

such as local area network (LAN) and 5G mobile communication.[14, 50] In this method, the transmitter sends the modulated data symbols through several frequency sub-carriers which have small frequency difference among themselves. This method facilitates high speed and accurate data transmission with efficient frequency band utilization. Several MIMO receivers are aligned on the receiving side according to the OFDM frequency sub-carriers. At the transmitter side, data symbols are modulated using Fast fourier transform (FFT) and the received signals are demodulated using Inverse fourier transform to extract the transmitted symbols in the receiver side. Such implementation facilitates the data transmission process through bolstering signal gain and reduction of selective fading.[14]

In this research, the spectrum occupancy is detected from the collected data of energy from received signal of MIMO-OFDM receiver. The spectrum occupancy detection can be observed as a binary null hypothesis testing where H_0 is the null hypothesis and H_1 is the alternative hypothesis.[29]

$$\begin{cases} H_0 : \text{No data symbol exist in PU spectrum} \\ H_1 : \text{Data symbol exists in PU spectrum} \end{cases}$$

During the real-time spectrum sensing operation, the SU considers N_{sample} number of samples to decide if there is a PU in the network. Since there are multiple antenna of transmitters and receivers in the OFDM-MIMO system, the base band signal received at the i^{th} receiver node for n data symbol can be represented by $r_i[n]$ as followed :

$$r_i[n] = h_i[n] * s[n] + w[n]; \quad \text{for } n = 1, \ldots, N_{sample}; \tag{1.1}$$

Here, $s[n]$ is the base-band signal of the transmitted data symbol n and $h_i[n]$ is the channel impulse response at receiver antenna i. $[*]$ is convolution operator. $w[n]$ is the additive

white gaussian noise (AWGN). Now if the modulation is done on p subcarrier frequency by Quadrature Phase Shift Keying (QPSK), the discrete fourier transform (DFT) of received baseband signal $r_i[n]$ can be presented by $R_i^p[k]$ as follows :

$$R_i^p[k] = H_i[k].S^p[k] + W_i^p[k]; where\ k\ is\ discrete\ frequency\ index \tag{1.2}$$

$H_i[k]$,$S^p[k]$ and $W_i^p[k]$ are the DFT from the time domain of channel impulse response $h_i[n]$, baseband signal $s[n]$ and AWGN noise signal $w[n]$ from equation 1.1 respectively. [.] is the multiplication operator. Now, for the null hypothesis test, the spectrum sensor in the ideal condition would choose alternative hypothesis H_1 if there is baseband frequency spectrum $S^p[k]$ in the received signal frequency domain which means the spectrum is occupied by the PU user. Otherwise, it would choose H_0 as null hypothesis if $S^p[k]$ was unpresent in the received signal which means the spectrum is current not being utilized by the PU.

The null hypothesis can be summarized as followed :

$$R_i^p[k] = \begin{cases} W_i^p[k] & \text{if } H_0 \\ H_i[k].S^p[k] + W_i^p[k] & \text{if } H_1 \quad \text{for } i = 1, \ldots, total\ number\ of\ receiver\ antennae; \end{cases}$$

1.2 Cost Efficient Spike-timing based smart AI

Traditional ML-based NPUs (Neural Processing Unit) rely on high data transfer rates as they need to continuously perform dense matrix-vector multiplication for their weight training and apply the trained weights on incoming data during the inference. Such reliance on a heavy data transfer rate increases the negative impact of "Memory wall," where even the most computationally efficient logic units face bottle-neck issues due to frequent data

movement between the processing unit and memory unit; increasing pressure on the limited memory bandwidth of the data transfer bus and thus slowing down the heavy memory access dependent tasks.[35] Implementing such computational units for the traditional ML-based accelerators, which can perform multiple matrix-vector multiplication and process the input data parallel-wise, demands significant resource and power utilization. Even the best hardware-efficient traditional accelerators, such as Echo State Networks, where only the output layer has trainable parameters, require sufficient area and power utilization so that high speed-real-time throughput can be achieved.[52, 54] An LSM neuromorphic processor, inspired by the human brain, which functions based on spiketemporal dynamics, can circumvent these factors since it extracts and processes information using the temporal encoding of the data. Since LSM operates on event-driven principles and spike-time processing, it can handle sparse and temporal data without continuous and frequent memory access, which reduces pressure on the memory bandwidth and resolves the memory access bottleneck issue. Additionally, LSMs can also leverage the parallelism of neural networks by distributing the spike trains and processing them parallelly among their spiking neurons both in the reservoir and output layer. The human brain-inspired event-driven and sparse processing nature of LSM also handles weight updates by utilizing the temporal information of the data, which is unique against traditional ANN.

The brain-inspired learning methods of LSM, which can be implemented in the hardware, generally require less resource utilization in terms of logic unit complexity and also have been found to be inherently energy efficient.[23] Moreover, the time-dependent synaptic plasticity-based nature of the LSM is more favorable to implement online real-time learning in the neural engine so that it can fine-tune its parameters intelligently to adapt to changing environments. Since the spectrum detection task is a time-series classification-based task, the temporal information processing capacity of the spiking LSM accelerator is likely to enhance

the performance of the neural engine to perform with expected accuracy as SNN networks have been found to be efficient for time-series problems [10, 11]. However, developing a spike-based temporal information processing capable accelerator comes with its own challenges. All the learning methods available for spike timing-based synaptic weight update are not equally capable of performing such complex tasks with fast convergence and maximum accuracy. Despite already existing research contributions in LSM architechture design with different learning methods,[36, 41] they are not sufficient for expectancy of our targetted applications, especially for higher noise low spatial multiplexing based signal data.

Our research aims to explore and find the best spike-based learning algorithm and develop an optimized novel hardware architecture that can implement the algorithm for real-time operation. Our novel neuromorphic processing engine also aims to incorporate the following advanced features into the design :

- **On-board Real-time Adaptive Training with Transfer Learning :** Our proposed neural engine can accomplish fast convergence with limited real-time data and calibrate itself with changing environments. To achieve this goal, we plan to implement a robust spike-timing based online learning in our design.

- **Intelligent learning engine design with asynchronous architecture** This feature will enable the design to optimize memory resources by leveraging the sparse data processing of the spike trains inside the LSM. Moreover, the specific algorithms implemented in the design will ensure that data transfer can be limited to smaller memory hierarchies in order to reduce pressure on memory bandwidth. An advanced iterative sparsification process for the parameters will also be incorporated to achieve memory efficiency and to avoid over-fitting of the model.

- **Real-time feature extraction with sophisticated on-chip encoders and delay**

networks: To enhance the performance of a neural network, incorporating an efficient delay-based architecture is crucial as it facilitates improved feature extraction from time-series data, enabling the network to learn system dynamics more efficiently for real-time symbol detection. Moreover, the design will also include encoder-decoder circuits to convert real-time input into spike train formats ensuring real-time operation of the NPU.

1.3 Challenges of LSM design

Implementing such a robust LSM comes with new challenges. Both from software and hardware perspective. We detail the challenges in two subsection below :

1.3.1 Algorithmic Challenges

The LSM is inspired from the spiking neural network (SNN) which a specific type of neural network based on spike timing based computing such like biological human brain. This bio-inpired network has some inherent challenges for performing accurately like the ANN networks. The local learning rules of SNN are not as efficient as the best learning rules in ANN like backpropagation. Since backpropagation is not hardware efficient to implement on-chip learning, the solution is to find better local learning rules or improve the existing one's so that the LSM can capture the inherent spectrum signal characteristics to decide whether the spectrum is occupied or not in real time. The hardware friendly local learning rules of LSM such as STDP learning are not always accurate in their performance in simple mechanisms. Moreover, the low learning complexity architecture of LSM although comes with pros of hardware friendly implementation of on-chip learning, it also comes with the

disadvantages of inferior performance. So, in this research we worked on improving the implemented local learning STDP methods and remove it's limitations by using some efficient algorithmic approach. We also, introduced unsupervised learning mechanism in the fixed parameter reservoir in the LSM so that the input energy data can be mapped into higher dimension by efficiently tuned neurons of the reservoir. This way, the reservoir can provide the readout layer with better extracted features from the input data which the later one can efficiently use for supervised RSTDP training of the read out weight parameters of the LSM which can finally help classify the spectrums efficiently as expected.

Moreover, we solved the overfitting learning issue of the LSM network by introducing an adaptive threshold based approach in the hardware design which can be implemented digitally by counting the incoming number of spikes.

One more issue with the reward based STDP (RSTDP) supervised learning was reaching the convergence to the highest accuracy of the model with less number of training epochs. Since the RSTDP model does not have any feedback from the predicted output during the training mode like the regular ANN networks such as loss function, it can hardly use any optimization techniques to reach the highest performing point during the training period. As a result, it can take the network significantly higher number of epochs to get to the highest accuracy performance and also retain the optimized parameters. In our research, we introduce a hardware friendly loss function to add stability to the learning of the network which also helps it to achieve better performance for highly complex dataset in reduced training period. The reduction of training period significantly contributes to the energy efficiency of the design as the hardware has to be operated less epochs for performing with expected accuracy.

1.3.2 Hardware Challenges

Developing a spike-based temporal information processing capable accelerator comes with its own challenges. The efficient and continuous processing of the input energy data in form of spikes as well as tracking the spike timing data in an efficient process brings some hard challenges for the design. Reduced latency design becomes very much important when the design aims to be as much as asynchronous architecture as possible, which in respone, results in area efficient hardware when implemented of application specific integrated circuit (ASIC) chip. Moreover, making the hardware architecture optimized to implement the determined algorithmic approaches are also challenging, especially which impacts the supervised and unsupervised learning of the readout and reservoir layer respectively.

In this research, the hardware design approach was taken from previously implemented LSM in hardware by Liu et ai [34] for speech signal recognition task. The design was simplified as the sparsiciation process was efficiently implemented using the adaptive threshold counter based approach. Which reduced the necessity of implementing any Calcium concentration based STDP approach for avoiding the overfitting system. This approach also removed the sparsification training mode from the design, which results in reduced training period for the LSM. Moreover, the unsupervised STDP learning in the reservoir of the proposed LSM in superior to the mentioned LSM as it uses exponential approximation based real-time hardware which can efficiently tune the reservoir weights with high precision calculation during training of the reservoir with unsupervised STDP. The implemented LSM in our research also uses learning engine which was designed with intelligent encoder based architecture, which made the design latency reduced. All these hardware improvements resulted in successfully implementing an on-chip learning hardware for spectrum sensing with high accuracy than any other proposed model so far for the same task.

Chapter 2

Literature Review

In traditional spectrum sensing, energy detector is one of the simplest model based approach which requires minimum data features: only energy signal data from received signal. It does not require any transmitter specific parameter for detecting the occupancy of frequency band from PU user. Although, this method has been found very hardware resource friend and low power hungry approach [43], it's performance drastically reduces when the channel environment is introduced to noise i.e signal-to-noise ratio (SNR) wall problem[35, 52]. The advancement in AI technology can help in this regard by learning to extract the inherent characteristics of the received signal from it's limited feature input energy data and thus classify the related spectrum vacancy or used status. But ANNs such as deep neural network are very power hungry and hard to implement in low-cost hardware devices such as FPGA since the backpropagation learning methods face severe memory contention problem due to nature of their operations. [6]

There has been a recent push in the research for finding machine learning architecture which can solve the memory wall issue for AI specific applications. AI is also being used for developing energy efficient hardware in this regard[4]. Moreover, there have been recent researches on exploring in-memory computing for hardware efficient ANN networks such as reservoir computing at device level in analog domain using different resistive ram based technology[38, 39, 40, 45, 46]. Infact, even specific analog domain researches such as in the field of spiking neural network (SNN) like Zheng et ai[60] have been explored. Researches

are also going on in hybrid domain where both digital and analog systems are combined to mitigate the memory wall issue for reservoir computing such as Osaze et ai [47, 61]. But analog devices are not scalable for industry purposes. That is why exploration has been going on for developing hardware efficient reservoir computing architecture in fully digital domain such as FPGA. [9, 30, 31, 32] Spiking neural networks (SNN) such as LSM which is also a specific type of reservoir computing (RC) are at a favourable position to be implemented on the hardware due to their energy efficient sparse spike based computing system as well as being inherently capable of extracting the features effectively from time sereis based data [12, 53, 59]. This is why, they are being extensively explored for application in both analog and digital hardware domain. For example, the Neurogrid multi-chip system, as described in [3], combines analog electronic circuits to emulate neural elements while utilizing digital spikes to transmit axonal arbors. Additionally, [51] introduced an analog Spiking Neural Network (SNN) to enhance the functionality of traditional cardiac synchronization therapy devices. Analog SNN devices can produce low-power accelerator for different applications by facilitating both from the sparse computing mechanism of LSM/SNN and by exploiting the inherent features of CMOS devices. However, such devices are not able to efficiently perform in predicting the results for different complex applications such as spectrum sensing in real-time with high noise channel environment. From the digital domain, different initiatives have been taken to implement SNN based hardware. Among them, IBM's TrueNorth chip [1] and Intels Loihi [6] were one of the most famous digital SNN chips. Both of the designs have own drawbacks such as the TrueNorth does not posses any on-chip learning capability and performs only inference of different trained SNN networks in real-time operation. For the Loihi chip, there has not been any signficant real world application such as spectrum sensing task explored and produced satisfactory results using on-chip training so far. As discussed before, although the SNN/LSM neurons represent the biological neuron structures more than the artificial previous gen ANNs, the so far modeled local learning mechanisms of

SNN/LSM neurons face difficulty in performance when matched against the ANN training mechanisms. In our work, we mitigate different limitations of LSM from both algorithmic and hardware approaches and make the dedicated LSM hardware perform with satisfacrtory on-chip training and prediction performance.

Chapter 3

Spiking neuron & Liquid State Machine

Spiking neural network (SNN) has similar network topology like the regular ANN networks. But the core difference between them is in the neuron unit structure and their functioning characteristics. Spiking neuron have more resemblance to actual behaviour of biological neuron which mainly performs on spikes. The brain consists of million number of signal processing units named as neurons. By employing complex mechanisms, the brain neuron units carry out difficult tasks such as information processing, pattern recognition etc. To understand the SNN architecture and one of it's specific topology which is liquid state machine or LSM, the spiking neuron is explained in a detailed approach in this chapter :

3.0.1 Biological Spiking Neuron

Since the spiking neuron has similar characteristics and functioning behaviour as a biological neuron, this section will discuss some abstract about the spiking biological neuron structure. A biological neuron is mainly consisting of 3 sections : Dendrite, Soma and Axon. The dendrites are the receptors that a neuron uses to receive the processed spike signals from the other neurons. Soma is the neruon cell body itself which functions as a non-linear processing system. The incoming spike signals from dendrites are processed through this soma where

the membrane potential is changed through the action of these spikes. When the membrane potential crosses a certain threshold limit, a new spike is generated from the cell itself and propagated through axons to the subsequent neuron connections. Thus the brain topology architecture consists of mainly two types of neuron : presynaptic neuron which sends spikes and post synaptic neuron which receives spikes and calculates the membrane potential. These two roles can be played by the same neuron inside the network. Once a spike is generated, the neuron potential is set to resting potential (V_{rest}).

The propagation of this spike signals through synapse depends on different ions such as chlorine, sodium, potassium and calcium. The calcium ion concentration plays a significant role in deciding whether the spike should be propagated to the next neuron or not. The calcium ions interact with proteins like synaptotagmin, which in result release the neurotransmitters resulting the spike propagation from presynaptic to postsynaptic neuron. In this research, we model the behaviour of calcium concentration with adaptive threshold activity in the digital neuron architecture.

3.0.2 Action Potential

The generation of an action potential, often referred to as a spike, by a neuron is characterized by a distinctive voltage signal. Typically, this signal exhibits an amplitude of approximately 100mV and lasts for a duration of 1-2ms. Within the axonal membrane, an array of ion channels serves as repeaters, ensuring the faithful transmission of the signal along the length of the axon. While all spikes share a uniform shape, information is encoded within the timing and frequency of their occurrence. Notably, neurons cannot sustain continuous spike generation; there exists a minimum time interval between successive spikes known as the absolute refractory period, emphasizing the discrete nature of spike events. Consequently, a

sequence of spikes generated by a neuron forms what is known as a spike train.

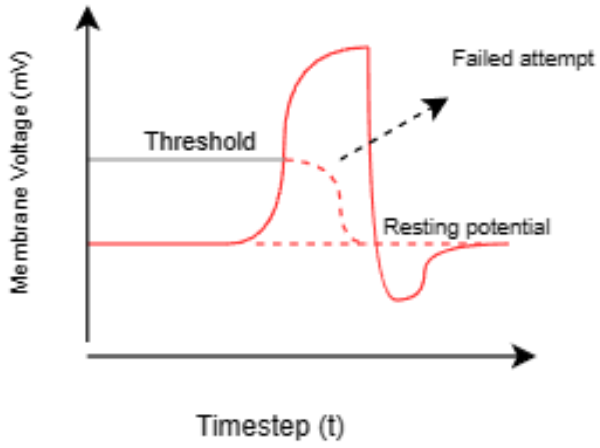

Figure 3.1: Spectrum Sensing architecture with hardware accelerator

Understanding the shape of an action potential involves observing the dynamic changes in the properties of the neuronal membrane. This process unfolds in distinct stages:

1. Rising Phase: Depolarization of the nerve membrane occurs, leading to the start of increase in membrane potential relative to the surrounded external environment.

2. Overshoot Phase: During this phase, the neuronal membrane potential surpasses the outer membrane potential, reaching its peak value before transitioning to the falling phase.

3. Falling Phase: Re-polarization takes place during this phase, bringing the membrane potential back to a level lower than the resting potential.

4. Undershoot Phase: Gradual restoration of the membrane potential to its resting state, completing the action potential cycle.

3.0.3 Computational Model

There are different types of computational model which represent the behaviour of the actual biological neuron function. From 3.0.2 it can be understood that the change of action potential results in changing the membrane potential. One of the most accurate mathematical model was derived from differential equation and follows the behaviour of the actual biological neuron closely is the Hodgkin and Huxley [15] model. The researchers performed multiple experiments on the axon of a large size squid and derived the mentioned model. But such differential models are not hardware friendly as they are tough for hardware calculation. For real-time device implementation, there are simpler abstraction which follows the behaviour of neuron and can be modeled with some resistors, capacitors or other circuit components. One of the most famous models that are implemented for neural network both in software and hardware is named as leaky integrate and fire (LIF) neuron. The LIF neuron is widely used in spiking neural network based machine learning networks also. In this research, the spiking neuron is represented by this LIF neuron model.

Although from neural network perspective, topological formation of ANN and SNN networks are similar, there are fundamental difference in the behaviour method between the neuron unit of both types of networks. The SNN neurons perform on addition of the updated weights based on the input spike positions. The summed weights are taken as membrane potential value and compared against a threshold potential instead of getting the neuron output through a ReLU or Sigmoid function like the ANN neurons. Once the membrane potential exceeds the threshold value, a spike is generated and the membrane potential is pulled down to V_{rest}. The generated spike will be delivered to the input of other spiking neurons as a forward or recurrent connection, depending on the network connection structure.

In the actual brain, the generated spikes do not propagate to all the next layer neurons

connected to the current neuron through it's axon. There are dynamical synaptic connection between axon and dendrite receptors of two neurons that also play a role whether the spike would be delivered to a neuron or not. This synapse connection strength was modeled into a numerical metric by Lapicque [25] who experimented on a nerve fiber collected from a frog's leg by providing stimulating it with a current source.

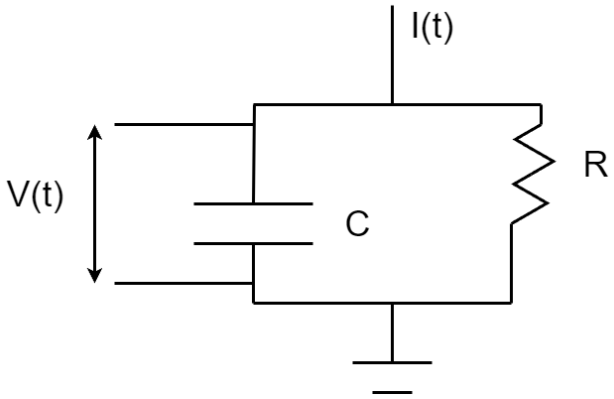

Figure 3.2: Spectrum Sensing architecture with hardware accelerator

He observed the variation of time that the frog leg takes to twitch depends on the change of amplitude and duration of providing the stimulus current to it. After experimentation, Lapicque proposed that the neuron spike function behaves similar to a low pass RC filter where the resistance is modeled based on the gating of ions channels to modulate the charge carrier diffusion through the membrane towards the cell and the capacitance is modeled from the insulation of bilayer lipid of surrounding membrane of the neuron cell. The spiking neuron model dynamics thus can be represented by a low pass filter RC circuit equation of schematic in figure 3.2 by the following :

$$\frac{dV_{mem}}{dt} = -\frac{V_{mem}}{\tau_{mem}} + \frac{I(t)}{C} \tag{3.1}$$

In the stated equation, V_{mem} is the current state of the action potential of the membrane.

τ_{mem} is the time constant of the decay rate of the membrane potential. $I(t)$ is the injected current from the external source. Since in the actual brain topology, the injected current enters from multiple presynaptic neuron, the current source can be modeled for a certain post-synaptic neuron as a function of the incoming injected current of all the presynaptic neurons and the remaining strength connection between them. If $I_{pre}(t)$ is the incoming input current from a certain presynaptic neuron, w_{pre} is the weight connection strength between the current neuron and the respective presynaptic neuron and t_j is the time representation on the j^{th} time instant with associated synaptic transmission delay d_{pre}, the total stimulation current from one presynaptic neuron over all time steps T can be represented as follows :

$$I_{pre}(t) = \sum_{j=1}^{T} w_{pre} s(t - t_j - d_{pre}) \tag{3.2}$$

Here, $s(.)$ is the spiking activation function which represents binary $1, 0$ depending on whether there is a spike for the corresponding time step or not.

Now, for all the M number of presynaptic neuron the total presynaptic current injected can be represented as

$$I_M(t) = \sum_{i=1}^{M} \sum_{j=1}^{T} w_i s(t - t_j - d_i) \tag{3.3}$$

If $I_M(t)$ is used for the total input current for membrane potential dynamics equation in 3.1 then it can be written as :

$$\frac{dV_{mem}}{dt} = -\frac{V_{mem}}{\tau_{mem}} + \sum_{i=1}^{M} \sum_{j=1}^{T} w_i s(t - t_j - d_i) \tag{3.4}$$

For the case of the supervised R-STDP learning in the output layer, there is am external

signal that has to be introduced to train the specific desired neuron. In our research, we named it as classification teacher (CT) signal. The CT signal is provided only to the desired output neuron for each sample which is supposed to predict the target label of the current sample during the training. If $I_{ct}(i)$ is the CT signal for desired output neuron i, then the equation can be written as :

$$\frac{dV_{mem}}{dt} = -\frac{V_{mem}}{\tau_{mem}} + \sum_{i=1}^{M}\sum_{j=1}^{T} w_i s(t - t_j - d_i) * I_{ct}(i) \tag{3.5}$$

Where $I_{ct}(i) \in \{1\}$ for the desired output neuron i and $I_{ct}(i) \in \{0\}$ for the undesired output neurons for a certain sample spike train during training.

Now, if the equation of 3.5 can be converted into a digitised equation with difference instead of differential equation, then the hardware friendly version of membrane volage calculation can be referred as :

$$V_{mem}^{n} = V_{mem}^{n-1} - \frac{V_{mem}^{n-1}}{\tau_{mem}} + \sum_{i}\sum_{j} W_i S(T_n, T_{i,j} + D_i) * I_{ct}^{n-1} \tag{3.6}$$

Here n subscript is the discrete time step of the digitised equation instead of continuous value of differential equation. The equation refers that the input spike keep being accumulated into register V_{mem} unless a certain threshold is reached. When the threshold is reached, a new spike output is generated from the neruon itself which propagates to the forward layer meanwhile the neurons V_{mem} is set to $V_rest = 0$. If there is no spike input on certain timestep n then the equation for that timestep becomes a decaying equation :

$$V_{mem}^{n} = V_{mem}^{n-1} - \frac{V_{mem}^{n-1}}{\tau_{mem}} \tag{3.7}$$

The decay continues to consequent timesteps if there are no incoming spike input in them

also until it reaches $V_{reset} = 0$ where it stop and stay at the resting potential before new spike input comes to the neuron to increase the membrane potential again.

Thus, how the digitised spiking neuron in our thesis works.

3.0.4 Liquid State Machine

A liquid state machine is a specific type of reservoir computing (RC), which operates based on spike-temporal correlation. An RC is a distinctive type of RNN (Recurrent Neural Network) consisting of an input layer, reservoir layer and readout(output) layer. The reservoir is a hidden layer with fixed weight parameter based neuron synapses used to generate a higher dimensional representation states from the input data. The neurons inside the reservoir also have sparse recurrent connections among themselves which help to extract better spatio-temporal correlation from the derived input features of the generated states. The state representation from an LSM reservoir can be put throught the following equation :

$$\begin{aligned} \mathbf{S}[t] &= (1-b)\mathbf{S}[t-1] \\ &\quad + b \cdot f(W_{in}^T \mathbf{X}[t] + W_{res}^T \mathbf{S}[t-1]), \end{aligned} \tag{3.8}$$

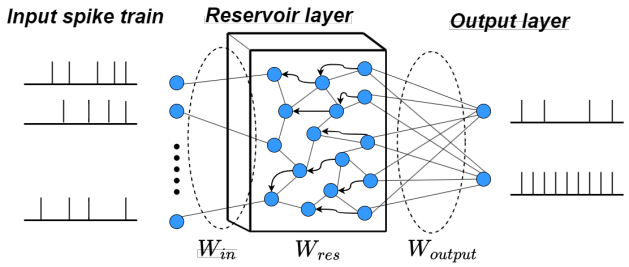

Figure 3.3: Liquid State Machine with spike based computing

Here, t is the discrete time step and $\mathbf{X}[n] \in \{0, 1\}$ and $\mathbf{S}[n] \in \{0, 1\}$ are the input and

generated states in format of binary spike train from the reservoir layer of the LSM. W_{in} represents the connection between input layer and reservoir layer which are non-trainable fixed and sparse connections. W_{res} is the reservoir parameters among the recurrent synapses of the liquid reservoir of the LSM which is also sparse. The W_{res} can be non-trainable fixed or be trainable with unsupervised learning mechanism. The constant parameter b can be tuned by controlling the weighted average of the LSM reservoir. The output spike train $Y[t]$ can be predicted from the generates states through the readout layer by the equation :

$$\mathbf{Y[t]} = (\mathbf{W_{readout}})^{\mathbf{T}} \cdot (\mathbf{S[t]}), \tag{3.9}$$

Where $W_{readout}$ is the readout layer trainable weights which can be trained through a supervised learning algorithm. The $\mathbf{Y}[t] \in \{0, 1\}$ also is predicted in the form of spike train. The shape of $Y[t]$ spike array can be affected by the choice of output encoding and learning mechanism.

3.0.5 Advantages for LSM network

There are several advantages of designing an LSM.

1. **Local learning :** In the traditional dense layer ANN, back-propagation is the most effective method used for any type of classficiaton or regression task. This back-propagation is although very effective, it has it's own drawbacks. The vanishing or exploding gradient problem can be a great hindrance for deeper network layers which are needed for extracting better spatio-temporal correlation from the input signal data. There are different solution methods implemented in deep neural networks (DNN) algorithms that can solve this issue such as the skipping layer in ResNet [56]. But the back-propagation faces a fundamental issue which limits it to be implemented on

hardware for on-chip learning. That is, non-local learning mechanism. The up stream gradients have to be back-propagated during training on each iteration as they have to be multiplied with local gradients inside each neuron. These mechanisms put pressure on the memory wall issue if implemented for hardware on-chip training, where the bottleneck of the data transfer between memory and computing logic element becomes a severe issue. The LSM or in greater sense spiking neural networks, have it's own distinct supervised and unsupervised learning mechanisms, derived from brain-computing architecture, which are local in nature and the tuning of the weight parameters are done inside the neuron only. No transfer of data between neurons needed[6].

2. **Avoidance of Overfitting:** Unlike some traditional neural network architectures, LSMs often demonstrate robustness against overfitting, a common problem in machine learning where the model memorizes noise or irrelevant features in the training data. By employing sparse distributed representations and incorporating temporal dynamics, LSMs can generalize well to unseen data, enhancing their performance in various tasks.

3. **Hardware-Friendly Design:** The spiking nature of LSMs lends itself to efficient hardware implementation, making them suitable for deployment in resource-constrained environments such as embedded systems or neuromorphic hardware. The spike based mechanism that is used by human brain, can be used to perform complex pattern recognition task with very limited amount of power (1220 W) which is extremely energy efficient as compared to the DNN networks for same tasks. [8] Moreover, the binarzied format and sparse spike data computation based training mechanism is low-resource complexity as well as energy friendly for hardware design. Thus, the event-driven processing paradigm of LSMs helps to minimize computational overhead and power consumption, making them an attractive choice for real-time applications and low-power devices. Additionally, the simple synaptic update rules and parallelizable

nature of LSM computations further contribute to their hardware-friendly design.

Chapter 4

Learning Algorithm in LSM neurons

4.0.1 Triplet based unsupervised learning

The synapse connection between neurons of the brain adapt through a specific biological plasticity mechanism. This mechanism helps the neurons of our brain to learn to do different tasks such as image classification, speech recognition, gesture recognition etc by processing the input information. In the LSM, the reservoir synapses are random and sparsely connected along with recurrent connection in the network, which in return helps it to extract better features from the input by utilizing the present and past input spike based temporal relation generated by reservoir from the input spike train. The reservoir and readout layer of the LSM are connected through some other sets of weight parameters which are trained using a supervised local learning algorithm that is also dopamine based biologically inspired. In regular LSM, the reservoir weights remain fixed as it follows the structure of reservoir computing where the fixed weights connection in the reservoir layer are used to generate echo through recurrent synapses and extract features from the input using the fixed weights and recurrent states of the reservoir. But, it has been found that introducing Unsupervised local learning to the reservoir can greatly help to converge better weights in the reservoir than randomly initiated weight parameters.[41] One of the biologically inspired unsupervised learning mechanism, that only uses the input spike train's and generated output spike's timing difference to tune the weights is named as spike timing dependent plasticity (STDP).

Utilizing STDP to train the LSM reservoir can help to tune the weights more efficiently to extract better features from input spike train as well as converge to sparse connection in some of the aspects. The sparse connection inherited as a result of STDP training can help to reduce the hardware complexity and improve energy efficiency of the reservoir of the LSM. However, the implementation of STDP on-chip learning in the hardware comes with two tradeoff challenges: precision and complexity. If more spike temporal information need to be extracted directly from the input spike trains on-hardware training, then the hardware complexity increases exponentially. Again, using counter based designs to reduce the hardware complexity can increase the latency and impact the training speed in on-chip performance. To solve the presented challenges, we took a balanced approach that uses moderate encoder based designs to extract better feature from the input than regular duplet spike based STDP hardware. This approach is called triplet based spike timing dependent plasticity (TSTDP) training for LSM reservoir. The proposed approach helped increasing the accuracy of the overall LSM on the dataset effectively with moderate hardware complexity increase in the reservoir.

The biologically inspired STDP learning rule utilizes the spike timing difference of pre-synaptic and post-synaptic neuron to decide whether the weight should be strengthened or weakened. In biological neuron, the potentiation or depression calculated between pre-synaptic and post-synaptic neuron is not only impacted by the nearest spiking event of the presynaptic spike timing or post-synaptic spike timing. For hardware low resource complexity, usually the ”nearest-neighbour” approach is taken such that, if the pre-synaptic neuron spikes at certain time instant, then only the previous nearest spike timing of the post-synaptic neuron is searched for depression and vice-versa for potentiation. This simplified method cannot extract enough useful features from the input spike trains. Thus, resulting in poorer performance in prediction. To mitigate this issue, a better and efficient approach is taken

for weight adjustment by taking into consideration a third spiking event, named as triplet spike timing plasticity (TSTDP) which deploys the equations below :

$$\triangle w = \begin{cases} \triangle w^+ = e^{-\triangle t_1/\tau^+}(A_2^+ + A_3^+ e^{-\triangle t_2/\tau^y}) \\ \triangle w^- = -e^{\triangle t_1/\tau^-}(A_2^- + A_3^- e^{-\triangle t_3/\tau^x}) \end{cases} \tag{4.1}$$

Here Δw^+ and Δw^- mean the amount of weight change that needs to be added(subtracted) with the existing weight to strengthen(weaken) the weights for potentiation(depression) respectively. Potentiation occurs when the presynaptic neuron spikes earlier than the post-synaptic neuron and depression occurs when presynaptic neuron spikes later than post-synaptic neuron. A_2^+, A_3^+, A_2^-, A_3^- can be designated as strength constants, and τ^+, τ^-, τ_x, and τ_y act as timing constants which decide the width of the STDP curve.

The depth of the bit resolution of the timing window Δt decides the actual width of the STDP curve on the horizontal axis.

Moreover, the timing window Δt can be defined as below:

- Δt_1: Spike timing difference between presynaptic neuron spike and post-synaptic neuron spike.
- Δt_2: Spike timing difference between two consecutive spikes generated from the same post-synaptic neuron.
- Δt_3: spike timing difference between two consecutive spikes generated from the same pre-synaptic neuron.

The proposed learning engine utilizes priority encoder based architecture to calculate the timing differences among the spikes. When the MSB of the presynaptic shift register is 1,

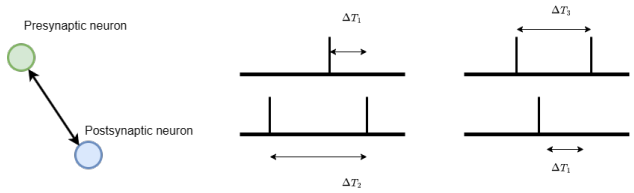

Figure 4.1: TSTDP Timing Differences for pre-post-pre spiking and post-pre-post spiking respectively

then the priority encoder searches for the nearest spike in the post-synaptic shift register. From that index, it continues searching for another nearest leading 1 index in the presynaptic shift register so that the pre-post-pre spike timing differences can be calculated for depression cases. The post-pre-post spike timing differences are calculated for potenitation cases when needed. All of these calculations can be done in one signle clock cycles. The maximum timing window between two spike events of presynaptic and post-synaptic shift registers are restricted as 4 indexes. Such method is faster to calculate the timing difference as the design is asynchronous instead of using a sequential counter which takes 4 clock-cycles to determine only one section of the timing difference.

Implementation of TSTDP design in the learning engines of the reservoir neuron from the LSM were proven to be perform with enough efficiency as par the offline training, which effectively reduces the time to implement the network on hardware as online training can be directly performed on-chip. The TSTDP design not only provides asynchronous architecture with reduced latency period, but also helps converging weights more accurately and sparsely among the reservoir neurons compared to traditional duplet based STDP architecture. In terms of design contribution, the implemented method also uses less number of Look-up-tables as the complex exponential curve values are directly calculation using an exponential approximator. Thus, the TSTDP based learning engine helps the overall LSM to extract the temporal feature from the input spike trains efficiently and perform better classification.

Chapter 5

Hardware friendly loss function

Inspired by the influence of dopamine on biological learning processes, our research employs Reward-based Spike-Timing-Dependent Plasticity (R-STDP) [42] as a supervised learning technique for weight training in the readout layer of the Liquid State Machine (LSM). R-STDP operates as a form of local learning mechanism, wherein the desired output neuron within the readout layer is either rewarded or penalized during training based on the timing of firing between presynaptic and postsynaptic neurons. When the presynaptic neuron fires prior to (subsequently to) the postsynaptic neuron, synaptic potentiation (depression) occurs, reinforcing (weakening) the synaptic connections between these neurons. Throughout the training process, only the target output neuron corresponding to the spike-train representation of the input sample is selected via a teaching signal utilizing one-hot encoding. The R-STDP learning equation utilized in our LSM design for readout layer training is as follows Reward-based STDP equation:

$$\text{Weight}_{ij} = \text{Weight}_{ij} + \eta \cdot \Delta w_{ij}^{\text{potentiation/depression}} \tag{5.1}$$

Where:

$$\Delta w_{ij}^{potentiation} = A_{\text{pos}} \cdot \exp(-\frac{\Delta t_{potentiation}}{\tau_{\text{pos}}}) \tag{5.2}$$

$$\Delta w_{ij}^{depression} = -A_{\text{neg}} \cdot \exp(-\frac{\Delta t_{depression}}{\tau_{\text{neg}}}) \tag{5.3}$$

Weight_{ij} is the synaptic weight between neuron i and neuron j,

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

$\Delta w_{ij}^{\text{depression}}$ is the reward signal for depression,

Δt is the pre-post spike firing time difference

η=1 is the learning rate,, $\tau_{\text{pos/neg}}$ are time constants

A_+& $A_-$are also equation constants

The rewards for both potentiation and depression may vary and can be adjusted to optimize performance for specific applications. Similar adjustments can be made for other constant values as well. While this reward or punishment mechanism aids the network in learning to a satisfactory level of accuracy, there remains a fundamental limitation to this learning approach. In conventional machine learning techniques, a loss function is typically employed to assess the disparity between predicted outputs and actual target labels, guiding parameter updates to minimize this loss function.

The loss function serves as a guiding signal for adjusting the weight parameters during the optimization process. Unlike other supervised machine learning methods, reward-modulated learning lacks feedback from predicted results, leading to weight parameters moving in different directions within the loss landscape without guidance. This often results in delayed convergence and occasionally less accurate performance, depending on the application's complexity. While basic R-STDP, which doesn't utilize a loss function, can simplify hardware

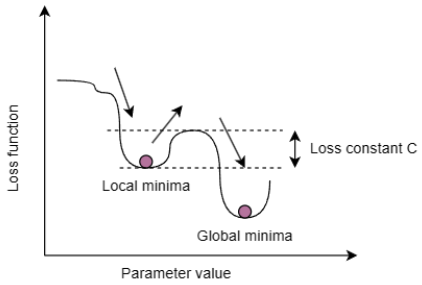

Figure 5.1: Loss function landscape with local and global minima. The red arrow directs toward higher saddle point, resulting in a non-optimized performance

architecture, it may compromise the on-chip learning accelerator's performance. To address this issue, we propose a novel loss function for LSM networks that guides parameter updates during real-time on-chip training and is hardware-friendly for implementation.

Let $S(t) \in \mathbb{R}^{\mathrm{D}N}$ denote the spike event emitted by the desired output neuron DN at each time step t during R-STDP training. If $\hat{y} \in \mathbb{R}^{\mathrm{D}_N}$ represents the total predicted spike count of the desired neuron for an input sample, the spike events $S[t]$ over T time steps can be summed as:

$$\hat{y} = \sum_{t=0}^{T} S[t]. \tag{5.4}$$

For each input sample, the predicted spike count $\hat{y}$ and the ideal spike count y for the desired target neuron are compared, and the total loss function value for all input samples in one training iteration is given by:

$$L_{\mathrm{loss}} = \sum_{i=1}^{\mathrm{I}} (y_i - \hat{y}_i) \tag{5.5}$$

This loss function guides whether the current iteration's updated weights should be retained or reverted to those inherited from the previous iteration. Thus, the model receives guidance

on parameter optimization towards the global minima, facilitating better accuracy in fewer iterations. Below is a simplified algorithm illustrating the implemented loss function in the output STDP.

Algorithm 1 Adaptive Learning with Weight Reversion

Initialize the weight matrix W
Initialize all other necessary parameters and hyperparameters
Define a loss threshold constant C
for each iteration i up to the maximum number of iterations **do**
    Copy the current weight matrix to W_{temp}
    Set the loss for this iteration, $loss_i$, to 0
    for each sample j in the dataset **do**
        Update the weight matrix: $W \leftarrow W + \text{reward}$
        Update the loss for this iteration: $loss_i \leftarrow loss_i + (y_j - \hat{y}_j)$
    end for
    if $loss_i < loss_{i-1} - C$ **then**
        Revert to the previous weight matrix: $W \leftarrow W_{\text{temp}}$
    end if
end for=0

Therefore, by evaluating the total loss function value of the current iteration compared to the previous one, the LSM's readout layer can enhance its precision globally and attain quicker convergence.

Chapter 6

Hardware Architecture

6.0.1 Overall LSM Hardware Design

In the proposed hardware architecture of the LSM, the preprocessing and training processors are implemented using an encoder layer, reservoir layer, and readout layer, as depicted in Figure 6.1. The encoder layer converts raw data, such as numerical values, into different spike encoding schemes like rate encoding, time-to-first-spike (TTFS), and inter-spike interval (ISI) encoding.

The reservoir neurons consist of learning units (LUs), which operate in parallel to process incoming spike trains. These spike trains are distributed to all LUs via a fixed yet random connection. The resulting output spikes from the LUs are temporarily stored in a 16-bit register known as the spike record. Simultaneously, some spikes from the spike record register are sent to other LUs through another fixed and random connection to extract temporal features from the input data. Additionally, the spikes stored in the spike record are transmitted to the readout layer to represent the reservoir response.

As illustrated in Figure 6.1, the readout layer comprises readout units (RUs) that can be configured for various tasks based on demand. Each RU receives the reservoir response from the reservoir layer and updates corresponding synaptic weights in parallel during training. For supervised learning, a classification teacher (CT) signal is utilized to manage weight updates within each RU, ensuring that weights are adjusted according to the task at hand.

Furthermore, the synaptic weights associated with each RU are stored in dedicated block RAM (BRAM).

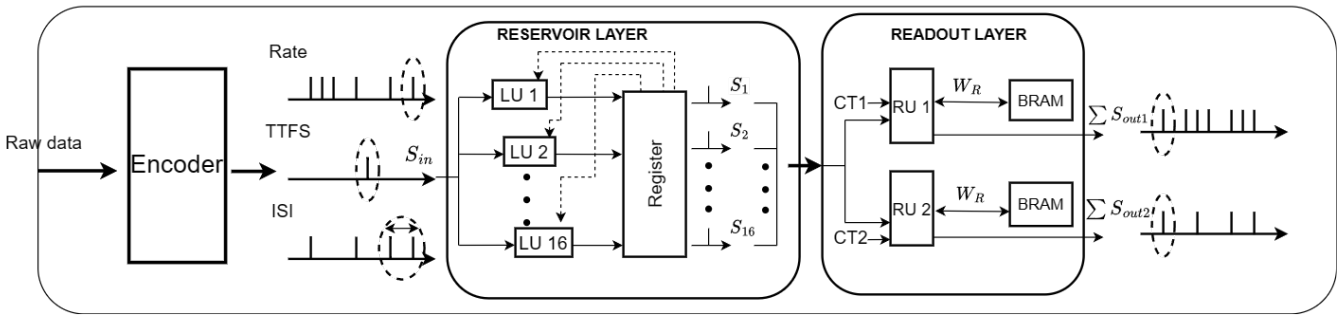

Figure 6.1: An exemplary LSM hardware architecture with 16 reservoir neurons and 2 readout neurons. One reservoir neuron receives one external input spike and up to 3 recurrent spikes from the spike record register. During training, each reservoir neuron generates one spike and sends them to the readout layer. Each readout neuron receives all 16 spikes from the reservoir layer. W denotes the plastic synaptic weight, and each BRAM stores the corresponding weight array depending on the task.

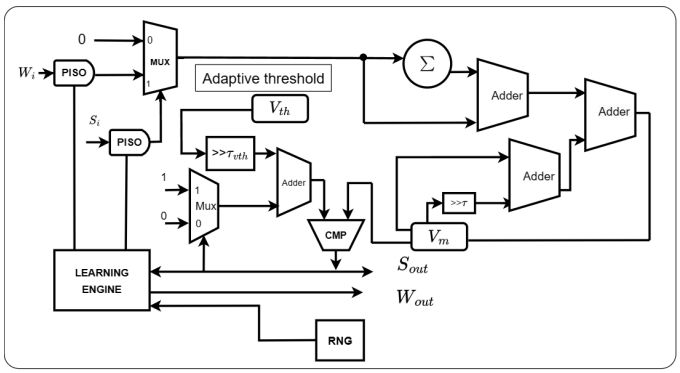

Figure 6.2: Hardware implementation of a digital neuron in both the reservoir and readout layers.

6.0.2 Implementation of Digital Neuron

In our proposed framework, digital neurons, situated in both the reservoir layer and readout layer, are constructed based on the commonly used Leaky Integrate-and-Fire (LIF) model.

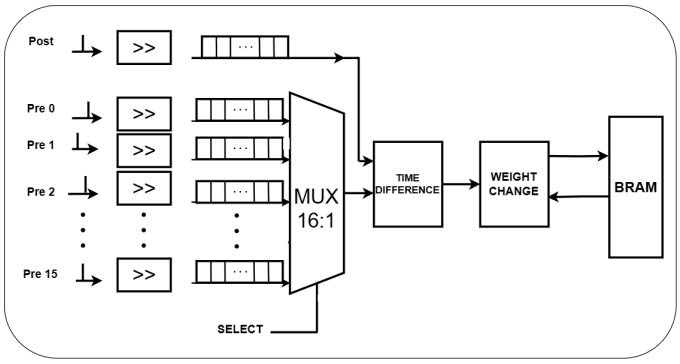

Figure 6.3: Hardware implementation of the learning engine in a neuron.

The dynamics of these neurons can be described by the following equation:

$$V_{mem}(t) = V_{mem}(t-1) - \frac{V_{mem}(t-1)}{\tau_{mem}} + \sum W_i * S_i \tag{6.1}$$

where $V_{mem}(t)$ represents the membrane potential of the LIF model at time step t, τ_{mem}denotes the membrane time constant, W signifies the synaptic weight stored in each neuron, and S_i represents the spiking events from the i_{th} presynaptic neurons. If the i_{th} presynaptic neuron does not fire at time $t-1$. S_i is set to 0, otherwise it is 1.

Fig. 6.2 illustrates the structure of a digital neuron within a Liquid State Machine (LSM) based on equation 6.1. This neuron receives input spikes S_i from presynaptic neurons in both the reservoir and readout layers. The variable W_i represents either the fixed synaptic weight in the reservoir layer or the plastic synaptic weight in the readout layer. During the training process, if the membrane potential V_{mem} exceeds a threshold V_{th}, the neuron emits a spike, and V_{mem} is reset to V_{rest}. Additionally, a small increment is applied to the threshold following each spike, which then decays exponentially over time. This adaptive threshold mechanism is designed to sustain spiking activity throughout the training phase, as depicted in equation 6.2.

$$V_{th}(t) = V_{th}(t-1) - \frac{V_{th}(t-1)}{\tau_{th}} + C_{th} \tag{6.2}$$

where $V_{th}(t)$ represents the adaptive threshold at time step t. τ_{th} denotes the threshold decaying constant, and C_{th} is the threshold increment constant. When a neuron emits a spike, C_{th} is set to 1; otherwise, it remains 0. This mechanism enables the adjustment of the threshold, maintaining a high frequency of synaptic weight updates initially and gradually decreasing it over time, thereby promoting efficient learning.

Fig. 6.3 illustrates the learning mechanism within a digital neuron. The learning engine consists of a shift register, time-difference calculator, weight change module, and weight memory module, such as BRAM. Different learning approaches lead to variations in the time-difference calculator and adjustments in the weight change module. Our proposed method incorporates two distinct learning rules within the learning engine. Specifically, supervised learning mechanisms like Reward-based STDP and unsupervised mechanisms like Triplet-based STDP are implemented in the readout and reservoir units, respectively. After updating the synaptic weights using the learning engine, the signal W_{out} is written back to BRAM. A random number generator (RNG) determines the probability of updating the synaptic weights. For each neuron, the signal W_i is associated with a BRAM output, and the input spike S_{in} is sequentially computed using a parallel-input-serial-output module (PISO).

6.0.3 Triplet STDP Digital Hardware Design

The Triplet Spike-Timing-Dependent Plasticity (TSTDP) module plays a crucial role in the learning process within the reservoir digital neuron. This module is responsible for updating the weights within the synaptic input processor, which utilizes these weights to compute

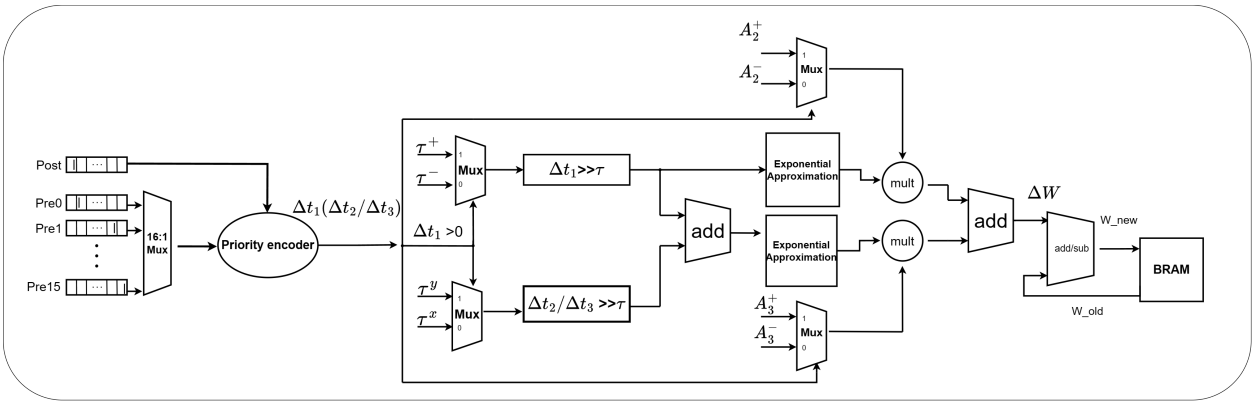

Figure 6.4: Hardware implementation of Triplet-based STDP

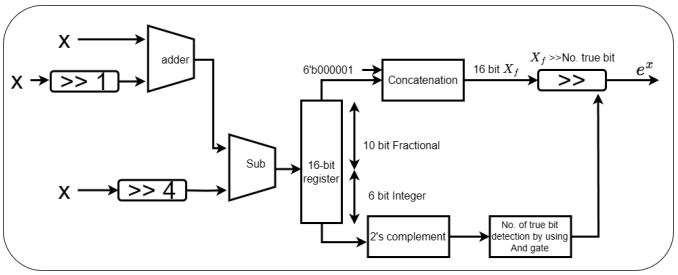

Figure 6.5: Implementation of exponential approximation in Triplet-STDP

the dynamics of the synaptic response of the neuron. Consequently, this computation aids in determining the membrane potential of the neuron based on the input spike pattern. The spike generation module of the Leaky Integrate-and-Fire (LIF) model monitors whether the membrane potential surpasses the voltage threshold and generates a spike accordingly. Upon each spike occurrence and its associated timing event, the weights between inputs and the target neuron are updated. The Reservoir Unit undergoes this training process until the distribution of synaptic weights stabilizes. Subsequently, the Readout Unit initiates a dedicated training phase for classification tasks. While the Readout Unit undergoes training, the Reservoir Unit remains active, providing spike inputs while maintaining its synaptic weights unchanged.

The top-level architecture of the unsupervised Spike-Timing-Dependent Plasticity (STDP) module consists of three sub-modules: Input Spike Selector, Timing Difference Calculator,

and Weight Change Calculator. These modules collaborate to form the top-level TSTDP learning module, which is utilized within the digital neuron model of the Reservoir Unit in our Liquid State Machine (LSM) processor. The functionality and implementation details of the TSTDP modules are elaborated below:

1) Input Spike Selector:

The input spike selector functions as the module needed for serializing the parallel tasks of updating multiple weight synapse between the current neuron and the previous presynaptic connections inside the learning engine of the current neuron for achieving the goal of hardware resource saving. For the reservoir neuron, there is one shift-register which records the spike for the neurons own generated LIF comparator. There are 3 presynaptic shift registers where one of the shift registers record the external spike input of the original encoded input spike train and two other shift registers take spikes from the reservoir generated recurrent states. These recurrent connections are randomly selected and remain fixed over the whole design during both training and inference. The shift registers always take the inputs in the MSB and keep shifting them towards the LSM by shifting every time a spike output is generated from the LIF comparator of the current neuron and all of them are 16 bit wide. A mux is used to select one of the presynaptic shift registers from which the whole 16 bit data content is sent to the output of the mux. The mux selective bit changes at every clock cycle whereas the shifting clock of the shift register changes only a the LIF comparator output is generated. Thus, it helps to extract the temporal connections of all the input

2) Timing Difference Calculator:

This part of the module is used for timing difference calculation between current neuron's own LIF generated post-synaptic and incoming pre-synaptic spike. If any of the MSB of

Algorithm 2 Timing Difference for TSTDP

Detect PRE spike at the 16th bit.
Check for a POST spike within the next 4 bits.
if POST SPIKE DETECTED: **then**
- $\Delta t_3 = 0$.
- Check for another PRE spike within 4 bits after the POST spike.
- Calculate Δt_2.

end if
Repeat a similar process for a post-pre-post synaptic event:
Detect a POST spike at the 16th bit.
Check for a PRE spike within the next 4 bits.
if PRE SPIKE DETECTED: **then**
- $\Delta t_2 = 0$
- Check for another POST spike within 4 bits after the PRE spike.
- Calculate Δt_3.

end if
Utilize the calculated time differences (Δt_1, Δt_2, Δt_3) for further processing or analysis in the context of synaptic events. =0

either the post-synaptic shift register or the selected pre-synaptic shift register by the mux is 1, then the leading 1 after that is searched in the other corresponding shift register. If no leading spike 1 is found in nearest 4-bit depth, then the whole timing difference is taken as 0. Otherwise, if a spike is found, then the timing difference is taken for calculating Δt_1 and then leading one in the primer shift register is searched again. If that is also found within 8-bit depth from the MSB position in the primer shift register, then that spike's position index is calculated using another priority encoder to calculate the second timing difference of Δt_2. Thus, both timing differences are added together to calculate the pre-post-pre or post-pre-post timing difference needed for the TSTDP calculation. An overall algorithm for the timing difference calculation is provided below:

3) Weight Change Calculator: Based on whether the presynaptic or postsynaptic spike is detected first, the time difference calculator module determines the sign of Δt_1. Subsequently, with Δt_1, Δt_2, or Δt_3, we employ an exponential approximate model, similar to the one pro-

Table 6.1: Parameters-TSTDP

Parameter	Value
τ_+	4
τ_-	5
A_2+	12
A_3+	7
A_2-	9
A_3-	10

posed in [13], to compute weight changes. To maintain precision, calculations are performed with 5 decimal bits using fixed-point representation. The exponential approximator block calculates $2^{1.786X}$ and outputs a 16-bit number with 14 fractional bits, 1 integer bit, and 1 sign bit. Triplet STDP learning is implemented, and the final weight is stored in the LSM Processor BRAM memory, where weights are saved sequentially for each digital neuron. The equations describing this module's behavior are provided below:

$$\begin{aligned} \Delta w^+ &= A_2^+ 2^{-1.4375 \frac{\Delta t_1}{\tau^+}} + A_3^+ 2^{-1.4375 \left(\frac{\Delta t_2}{\tau_y} + \frac{\Delta t_1}{\tau_+}\right)} \\ \Delta w^- &= -A_2^- 2^{-1.4375 \frac{\Delta t_1}{\tau^-}} + A_3^- 2^{-1.4375 \left(\frac{\Delta t_3}{\tau_x} - \frac{\Delta t_1}{\tau_-}\right)} \end{aligned} \tag{6.3}$$

Using shifters and adders, the output of the approximator block is summed to compute weight changes, which are then updated in the BRAM data array memory as illustrated in Figure 6.4. The parameters of the triplet STDP employed in our architectural model are detailed in Table 6.1.

6.0.4 Supervised learning engine (SLE)

The output neurons RU have a supervised learning engine (SLE), which uses the classification teacher signal and STDP learning probability curve, to permit the weight update process. There are K shift registers in the SLE of one output neuron to receive the spikes generated

by K number of reservoir neurons at each timestep. There is also one post-synaptic shift register to record the generated spike event from the output neuron itself.

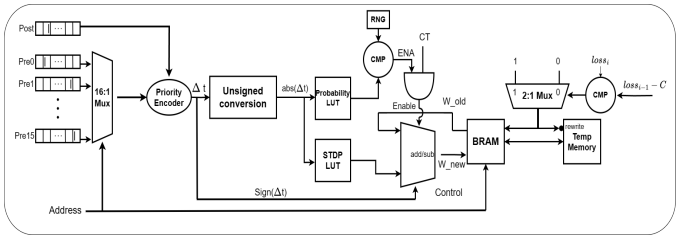

Figure 6.6: Implementation of Supervised Learning Engine (SLE)

Each shift registers are taken as of 16 bit wide. The shift registers right shifts once after every K clock cycle, during which time the PISO weights are accumulated according to the respective PISO spikes in the Vmem accumulator of the LIF part of the output neuron and generates one spike event (1/0) which is stored in the postsynaptic shift register which also shifts after every K clock cycle. The weight memory is a dual port memory ram which can read and write in the same memory address at two different clock edges in the same clock cycle. The select bit and the address of the weight memory increment by 1 at every clock cycle. Whenever there is a 1 at the MSB in any of the shift registers, the priority encoder determines the leading one position in the other shift register and thus generates the output of the spike firing time difference. The maximum window for timing difference calculation between spikes is taken as 12. The absolute value of the timing difference is used to generate a probability value through a look up table. The look up table values follow the activity based probability STDP curve. [20] Both the probability values and STDP curve values for determining the reward amount of the weight update are hardcoded into look-up-tables after optimizing them in a fixed point software simulation. If the timing difference is less, the probability value is high and vice-versa. The probability value is compared against a random number generator which generates ENA signal to formulate a hebbian learning based activity function. The sign bit of the original timing difference is used to

determine potentiation(depression) which finally decides whether the existing weight should be strengthened(weakened). The respective SLE of the desired output neuron is only trained when the classification teacher (CT) signal and the ENA signal are one which activates the Enable signal to permit the weight update for the desired neuron with represented reservoir state generated from the respective input sample of training dataset. The weight memory updated at the end of each iteration are stored in a temporary weight register which can be used for weight reversion if the loss function count of next iteration is extensively higher than the count of previous iteration. The loss function is counted at the output spike counter of the output LIF neuron and comparison-decision is taken through logic circuit to inform the SLE whether the current memory should copy it's weights to temporary register or vice-versa.

Chapter 7

Experiment & Results

7.1 EXPERIMENTAL SETUP AND EQUIPMENTS

The accelerator was synthesized using Vivado HLS and deployed on a Xilinx Virtex-707 FPGA. The dataset utilized for spectrum sensing is obtained from the RWTH Aachen University Static Spectrum Occupancy Measurement Campaign database [55]. This dataset comprises static spectrum occupancy measurements of Primary User (PU) activity across various frequency bands and time slots. It includes data collected under different channel conditions with varying noise levels, such as SNR= -10/-20/-30 dB. Each scenario involves different transmitter/receiver (TX-RX) configurations ranging from 2 TX- 2RX to 6 TX-6 RX. Increasing the number of antennas results in improved spatial multiplexing gain, enhancing the accuracy of spectrum band detection from the signal data.

7.2 Training settings and Results

LSM accelerators featuring diverse learning engines underwent training and evaluation using the aforementioned spectrum sensing dataset. All models employ the same R-STDP-based supervised learning engine (SLE) but differ primarily in reservoir learning methodology. The baseline LSM utilizes a reservoir with non-trainable fixed randomly initialized weights, while the other two methods incorporate unsupervised learning engines in the reservoir. The un-

Table 7.1: Parameter-TSTDP

Parameter	Value
τ_+	4
τ_-	5
A_2+	12
A_3+	7
A_2-	9
A_3-	10

Table 7.2: Parameter-SLE

Parameter-SLE	Value
τ_{pos}	2
τ_{neg}	2
A_{pos}	5
A_{neg}	3
τ_{th}	1000
C_{th}	1

Table 7.3: Weight Parameter Settings

Parameter Type	bit resolution
Weight resolution of Triplet STDP	16-bit (signed)
Weight resolution of SLE	10-bit (signed)
Weight initialization	Initialization technique from [26]
Weight bounding in readout layer	[-400,400]

supervised STDP LSM utilizes a lookup table-based learning engine with duplet spike, in contrast to our triplet spike-based learning engine in the reservoir, capable of performing exponential calculations on hardware. For model comparison, various metrics such as accuracy, resource utilization, power consumption, inference speed, and weight convergence were considered. The models underwent approximately 100 epochs of training with a loss constant of 50, and hyperparameters from Tables 7.1, 7.2, and 7.3 were kept constant throughout training and testing. Each model's reservoir unit comprised only 16 reservoir neurons to ensure low hardware complexity. To optimize hyperparameters for hardware implementation, a fixed-point software design emulating the hardware design was initially employed. The quantized parameters determined from this stage were then maintained constant during onboard training of weight parameters.

7.2.1 Comparative Analysis - Accuracy

For accuracy analysis, all variants of LSM accelerators were compared with state-of-the-art ML-based accelerator models [33] and traditional methods such as SLC [5, 24] commonly used for spectrum sensing. As the rate-based encoder yielded the best performance, it was exclusively implemented for generating Table 7.4 to facilitate accuracy comparison. Considering the IEEE 802.22 standard for WRAN (Wireless Regional Area Network), which stipulates spectrum sensing techniques capable of operating with a sensing receiver sensitivity of at least -116 dBm [49], evaluations were conducted only on two different SNR conditions (SNR=-10 or -20 dB) from the spectrum sensing dataset to benchmark compliance with this requirement. The experiments were conducted across various spatial multiplexing gains, ranging from (2 TX-RX) to (6 TX-RX). From Table 7.4, it can be inferred that all models, including traditional methods, exhibit accurate performance under low-noise conditions (SNR=-10 dB) for antenna configurations of (4 TX-RX and 6 TX-RX). For lower spatial multiplexing gains of 2 TX-RX, our proposed model outperforms the traditional SLC method, which performs worse than even a random classifier for this balanced dataset [14]. Additionally, our proposed model outperforms other methods/models in higher noise channel environments with SNR=-20 dB across all spatial multiplexing configurations.

7.2.2 Overall Design Analysis

To comprehend the overall efficiency of the suggested design model, various metrics including resource usage, inference time, power, and weight convergence are examined. Comparisons in resource usage were conducted across LSM accelerators employing different learning engines. As the complexity of the learning approach escalates, so does the resource consumption. The analysis presented in Table 7.5 indicates that, for 16 reservoir neurons, the increment

Table 7.4: Accuracy comparison at different SNR levels and epochs.

SNR	Model	2 TX-RX	4 TX-RX	6 TX-RX
-10 dB	SLC [48]	x	98.32%	99.46%
	FPDFR+TSML [48]	x	98.81%	99.75%
	QDFR+TSML[48]	x	98.53%	99.70%
	Baseline	64.78%	98.21%	98.56%
	Unsupervised STDP	68.39%	98.86%	99.71%
	Triplet-STDP-Loss	69.18%	99.18%	99.88%
-20 dB	SLC [48]	x	66.81%	95.26%
	FPDFR+TSML [48]	x	88.83%	96.75%
	QDFR+TSML [48]	x	88.71%	96.59%
	Baseline	60.34%	87.02%	93.14%
	Unsupervised STDP	64.53%	88.35%	95.79%
	Triplet-STDP-Loss	66.66%	90.86%	97.02%

in utilization is comparatively modest when juxtaposed with the performance enhancement achieved, with reported accuracy stemming from a -20 dB dataset for a 6 TX-RX configuration. Concerning the proposed TSTDP-loss model, FPGA utilization percentage increases by approximately 0.18% and 0.41% for LUT and FF, respectively, compared to the unsupervised STDP model. Unlike the unsupervised STDP model, which relies on pre-determined values optimized in software for lookup-table-based weight training in the reservoir, our proposed hardware implements real-time calculations in triplet STDP, guided by loss function for training, rendering the trade-off worthwhile for implementation.

Table 7.5: Resource Utilization Comparison

Resource	Baseline	Unsupervised	Triplet-STDP-loss
LUTs	3295	3615	4172
FFs	2978	3529	5997
Utilization (LUT%)	1.08%	1.19%	1.37%
Utilization (FF%)	.491%	.58%	.99%
Accuracy (-20 dB)	93.14%	95.79%	97.99%

Table 7.6: Power report

Model & Hardware Platform	**Energy/Sample (mj)**
QDFR+TSML [33]	.018
Baseline	.028
Unsupervised STDP	.17
Triplet-STDP-Loss	.996
QDFR+TSML in GPU[33]	1.632

Table 7.7: Inference Speed Comparison Table

Model	**Inference time**
Unsupervised STDP	1
Triplet-STDP-without loss	1.2x
Triplet-STDP-with loss	1.2084x

Dataset	**With-loss**	**Without-loss**	**Improvement (%)**	**Accuracy (%)**
-10 dB (2TX-RX)	60	88	46.67%	69.18%
-20 dB (2TX-RX)	56	147	155.3%	66.66%

Table 7.8: Comparison of iterations with and without loss

The comparison of energy consumption per sample for all models against state-of-the-art models is presented in Table 7.6. The results indicate that the quantized DFR model from [33] exhibits the lowest power consumption, whereas our designed accelerators demonstrate higher energy consumption per sample as complexity increases. The QDFR+TSML model employs a teacher-student model learning technique, training parameters offline and then transferring weights onboard. In contrast, our LSM accelerator models employ online learning, resulting in increased complexity and power consumption. The proposed Triplet-STDP-loss model exhibits relatively high power consumption due to the inclusion of a circuit for

exponential approximation, unlike the fixed/LUT-based design of the baseline and unsupervised STDP reservoirs. However, the proposed on-chip learning model remains more energy-efficient than training the QDFR+TSML model on a GPU, which consumes almost double the power of our dedicated FPGA design.

Regarding the implementation of the loss function detailed in Table 7.7, it only results in a 0.7% increase in latency when executed on the hardware. This increase is deemed satisfactory compared to the improvements in convergence efficiency and training period observed with the accelerator models for the proposed hardware design.

Chapter 8

Summary

This study presents a novel on-chip learning LSM accelerator tailored for blind spectrum sensing tasks, marking its inaugural application. A pioneering triplet-STDP encoder-based architecture is introduced in the reservoir, eliminating the necessity for sequential counters, thus conserving energy and reducing latency. Furthermore, the integration of TSTDP yields a 3.88% enhancement in accuracy compared to traditional LSM under -20 dB SNR conditions with a 6 TX-RX configuration. Implementing on-chip learning with high precision on hardware slightly increases LUTs and Registers by 0.29% and 0.499%, respectively, on the Virtex-707 FPGA board. Additionally, a novel hardware-friendly loss function for R-STDP output learning is introduced to expedite network convergence, reducing training periods by approximately 47% with a mere 0.7% increase in inference latency. Real-time spike generation for the input spectrum sensing dataset is achieved by incorporating a suitable rate-based encoder on the FPGA, facilitating efficient online data conversion and resulting in an 8.87% accuracy enhancement for the 6 TX-RX setup at -20 dB SNR. In summary, this endeavor yields the design of an on-chip learning ML processor adept at precise spectrum classification and efficient data conversion in high-noise environments.

Bibliography

[1] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

[2] Kang Jun Bai, Clare Thiem, Jack Lombardi, Yibin Liang, and Yang Yi. Design strategies and applications of reservoir computing: Recent trends and prospects [feature]. *IEEE Circuits and Systems Magazine*, 23(4):10–33, 2023. doi: 10.1109/MCAS.2023.3325496.

[3] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[4] Dake Chen, Xuan Zhou, Yinghua Hu, Yuke Zhang, Kaixin Yang, Andrew Rittenbach, Pierluigi Nuzzo, and Peter A Beerel. Unraveling latch locking using machine learning, boolean analysis, and ilp. In *2023 24th International Symposium on Quality Electronic Design (ISQED)*, pages 1–8. IEEE, 2023.

[5] Hao Chen, Lingjia Liu, John D Matyjas, and Michael J Medley. Optimal resource allocation for sensing-based spectrum sharing d2d networks. *Computers & Electrical Engineering*, 44:107–121, 2015.

[6] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao,

Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1): 82–99, 2018.

[7] Aaron Yi Ding and Marijn Janssen. Opportunities for applications using 5g networks: Requirements, challenges, and outlook. In *Proceedings of the Seventh International Conference on Telecommunications and Remote Sensing*, pages 27–34, 2018.

[8] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 2023.

[9] Victor M Gan, Yibin Liang, Lianjun Li, Lingjia Liu, and Yang Yi. A cost-efficient digital esn architecture on fpga for ofdm symbol detection. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 17(4):1–15, 2021.

[10] Ramashish Gaurav, Terrence C Stewart, and Yang Cindy Yi. Spiking reservoir computing for temporal edge intelligence on loihi. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, pages 526–530. IEEE, 2022.

[11] Ramashish Gaurav, Terrence C Stewart, and Yang Yi. Reservoir based spiking models for univariate time series classification. *Frontiers in Computational Neuroscience*, 17: 1148284, 2023.

[12] Arfan Ghani, T Martin McGinnity, Liam P Maguire, and Jim 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*, pages 513–522. Springer, 2008.

[13] Shaghayegh Gomar and Majid Ahmadi. Digital realization of pstdp and tstdp learning. *IEEE*, 2018.

[14] Kian Hamedani, Lingjia Liu, Shiya Liu, Haibo He, and Yang 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*, volume 34, pages 1292–1299, 2020.

[15] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[16] Shashank Jere, Hussein Metwaly Saad, and Lingjia Liu. Error bound characterization for reservoir computing-based ofdm symbol detection. In *ICC 2022 - IEEE International Conference on Communications*, pages 1349–1354, 2022. doi: 10.1109/ICC45855.2022.9839095.

[17] Shashank Jere, Ramin Safavinejad, and Lingjia Liu. Theoretical foundation and design guideline for reservoir computing-based mimo-ofdm symbol detection. *IEEE Transactions on Communications*, 2023.

[18] Shashank Jere, Ramin Safavinejad, Lizhong Zheng, and Lingjia Liu. Channel equalization through reservoir computing: A theoretical perspective. *IEEE Wireless Communications Letters*, 2023.

[19] Shashank Jere, Karim Said, Lizhong Zheng, and Lingjia Liu. Towards explainable machine learning: The effectiveness of reservoir computing in wireless receive processing. In *MILCOM 2023-2023 IEEE Military Communications Conference (MILCOM)*, pages 667–672. IEEE, 2023.

[20] Yingyezhe Jin and Peng Li. Ap-stdp: A novel self-organizing mechanism for efficient reservoir computing. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1158–1165. IEEE, 2016.

[21] Hyun Gu Kang, Iickho Song, Seokho Yoon, and Yun Hee Kim. A class of spectrum-sensing schemes for cognitive radio under impulsive noise circumstances: Structure and performance in nonfading and fading environments. *IEEE Transactions on Vehicular Technology*, 59(9):4322–4339, 2010.

[22] Vishal Khatri and Gaurab Banerjee. A 0.25–3.25-ghz wideband cmos-rf spectrum sensor for narrowband energy detection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(9):2887–2898, 2016.

[23] Marko Kosunen, Vesa Turunen, Kari Kokkinen, and Jussi Ryynänen. Survey and analysis of cyclostationary signal detector implementations on fpga. *IEEE journal on emerging and selected topics in circuits and systems*, 3(4):541–551, 2013.

[24] Vijaykumar Kuppusamy and Rajarshi Mahapatra. Primary user detection in ofdm based mimo cognitive radio. In *2008 3rd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2008)*, pages 1–5. IEEE, 2008.

[25] L Lapicque. L'excitation electrique des nerfs. *J. Physiol. Pathol. Gen.*, 9:620–635, 1907.

[26] Chankyu Lee, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity. *IEEE Transactions on Cognitive and Developmental Systems*, 11(3):384–394, 2018.

[27] Lianjun Li, Lingjia Liu, Jianzhong Zhang, Jonathan D. Ashdown, and Yang Yi. Reservoir computing meets wi-fi in software radios: Neural network-based symbol detection using training sequences and pilots. In *2020 29th Wireless and Optical Communications Conference (WOCC)*, pages 1–6, 2020. doi: 10.1109/WOCC48579.2020.9114937.

[28] Yibin Liang, Lianjun Li, Yang Yi, and Lingjia Liu. Real-time machine learning for symbol detection in mimo-ofdm systems. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 2068–2077. IEEE, 2022.

[29] Ying-Chang Liang, Kwang-Cheng Chen, Geoffrey Ye Li, and Petri Mahonen. Cognitive radio networking and communications: An overview. *IEEE transactions on vehicular technology*, 60(7):3386–3407, 2011.

[30] Chunxiao Lin, Yibin Liang, and Yang Yi. Fpga-based reservoir computing with optimized reservoir node architecture. In *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pages 1–6. IEEE, 2022.

[31] Chunxiao Lin, Muhammad Farhan Azmine, and Yang Yi. Accelerating next-g wireless communications with fpga-based ai accelerators. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2023.

[32] Chunxiao Lin, Muhammad Farhan Azmine, Yibin Liang, and Yang Yi. Leveraging neuro-inspired ai accelerator for high-speed computing in 6g networks. *Frontiers in Computational Neuroscience*, 18:1345644, 2024.

[33] Shiya Liu, Lingjia Liu, and Yang Yi. Quantized reservoir computing for spectrum sensing with knowledge distillation. *IEEE Transactions on Cognitive and Developmental Systems*, 15(1):88–99, 2022.

[34] Yu Liu, Sai Sourabh Yenamachintala, and Peng Li. Energy-efficient fpga spiking neural

accelerators with supervised and unsupervised spike-timing-dependent-plasticity. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(3):1–19, 2019.

[35] Amir Mehrabian and Amir Zaimbashi. Robust and blind eigenvalue-based multiantenna spectrum sensing under iq imbalance. *IEEE Transactions on Wireless Communications*, 17(8):5581–5591, 2018.

[36] Amir Mehrabian, Maryam Sabbaghian, and Halim Yanikomeroglu. Spectrum sensing for symmetric α-stable noise model with convolutional neural networks. *IEEE Transactions on Communications*, 69(8):5121–5135, 2021.

[37] Amir Mehrabian, Maryam Sabbaghian, and Halim Yanikomeroglu. Cnn-based detector for spectrum sensing with general noise models. *IEEE Transactions on Wireless Communications*, 22(2):1235–1249, 2022.

[38] Fabiha Nowshin. *Spiking neural network with memristive based computing-in-memory circuits and architecture*. PhD thesis, Virginia Tech, 2021.

[39] Fabiha Nowshin and Yang Yi. Memristor-based deep spiking neural network with a computing-in-memory architecture. In *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pages 1–6. IEEE, 2022.

[40] Fabiha Nowshin, Yi Huang, Md. Rubel Sarkar, Qiangfei Xia, and Yang Yi. Merrc: A memristor-enabled reconfigurable low-power reservoir computing architecture at the edge. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(1):174–186, 2024. doi: 10.1109/TCSI.2023.3329337.

[41] Qihang Peng, Andrew Gilman, Nuno Vasconcelos, Pamela C Cosman, and Laurence B Milstein. Robust deep sensing through transfer learning in cognitive radio. *IEEE Wireless Communications Letters*, 9(1):38–41, 2019.

[42] Fernando M Quintana, Fernando Perez-Pena, and Pedro L Galindo. Bio-plausible digital implementation of a reward modulated stdp synapse. *Neural Computing and Applications*, 34(18):15649–15660, 2022.

[43] Ashish Rauniyar and Soo Young Shin. Cooperative spectrum sensing based on adaptive activation of energy and preamble detector for cognitive radio networks. *APSIPA Transactions on Signal and Information Processing*, 7:e2, 2018.

[44] Dinesh Kumar Sah, D Praveen Kumar, Chaya Shivalingagowda, and PVY Jayasree. 5g applications and architectures. *5G Enabled Secure Wireless Networks*, pages 45–68, 2019.

[45] Md Rubel Sarkar and Cindy Yang Yi. An in-memory computing architecture utilizing energy-efficient vgsot mram device. *IEEE Transactions on Circuits and Systems II: Express Briefs*, pages 1–1, 2024. doi: 10.1109/TCSII.2024.3359993.

[46] Md Rubel Sarkar, Md Maruf Abir Bappy, Md Minhajul Azmir, Dewan Mohammed Rashid, and Saad Ibn Hasan. Vg-sot mram design and performance analysis. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0715–0719, 2021. doi: 10.1109/IEMCON53756.2021.9623147.

[47] Osaze Shears, Kangjun Bai, Lingjia Liu, and Yang Yi. A hybrid fpga-asic delayed feedback reservoir system to enable spectrum sensing/sharing for low power iot devices iccad special session paper. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[48] Yang Yi Shiya Lui, Lingjia Liu. Quantized reservoir computing for spectrum sensing with knowledge distillation. *IEEE Transactions on Cognitive and Developmental Systems*, 2022.

[49] Carl R Stevenson, Gerald Chouinard, Zhongding Lei, Wendong Hu, Stephen J Shellhammer, and Winston Caldwell. Ieee 802.22: The first cognitive radio wireless regional area network standard. *IEEE communications magazine*, 47(1):130–138, 2009.

[50] Gordon L Stuber, John R Barry, Steven W Mclaughlin, Ye Li, Mary Ann Ingram, and Thomas G Pratt. Broadband mimo-ofdm wireless communications. *Proceedings of the IEEE*, 92(2):271–294, 2004.

[51] Qing Sun, François Schwartz, Jacques Michel, Yannick Herve, and Renzo Dal Molin. Implementation study of an analog spiking neural network for assisting cardiac delay prediction in a cardiac resynchronization therapy device. *IEEE transactions on neural networks*, 22(6):858–869, 2011.

[52] Rahul Tandra and Anant Sahai. Snr walls for signal detection. *IEEE Journal of selected topics in Signal Processing*, 2(1):4–17, 2008.

[53] David Verstraeten, Benjamin Schrauwen, Dirk Stroobandt, and Jan Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.

[54] Gayathri Vijay, Elyes Ben Ali Bdira, and Mohamed Ibnkahla. Cognition in wireless sensor networks: A perspective. *IEEE sensors journal*, 11(3):582–592, 2010.

[55] Matthias Wellens, Alexandre de Baynast, and Petri Mahonen. Exploiting historical spectrum occupancy information for adaptive spectrum sensing. In *2008 IEEE Wireless Communications and Networking Conference*, pages 717–722. IEEE, 2008.

[56] Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. *arXiv preprint arXiv:2002.05990*, 2020.

[57] Jiarui Xu, Zhou Zhou, Lianjun Li, Lizhong Zheng, and Lingjia Liu. Rc-struct: A structure-based neural network approach for mimo-ofdm detection. *IEEE Transactions on Wireless Communications*, 21(9):7181–7193, 2022. doi: 10.1109/TWC.2022.3155945.

[58] Jiarui Xu, Lianjun Li, Lizhong Zheng, and Lingjia Liu. Detect to learn: Structure learning with attention and decision feedback for mimo-ofdm receive processing. *IEEE Transactions on Communications*, 72(1):146–161, 2024. doi: 10.1109/TCOMM.2023.3292468.

[59] Yong Zhang, Peng Li, Yingyezhe Jin, and Yoonsuck Choe. A digital liquid state machine with biologically inspired learning and its application to speech recognition. *IEEE transactions on neural networks and learning systems*, 26(11):2635–2649, 2015.

[60] Honghao Zheng, Kang Jun Bai, and Yang Yi. Enabling a new methodology of neural coding: Multiplexing temporal encoding in neuromorphic computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 31(3):331–342, 2023.

[61] Xuan Zhou, Xiaolei Zhu, Bing Chen, Yi Zhao, and Chengjie Fu. An 8-bit rram based multiplier for hybrid memory computing. In *2019 IEEE International Workshop on Future Computing (IWOFC*, pages 1–3. IEEE, 2019.