

# DIGITAL IMPLEMENTATION OF BIO-INSPIRED SPIKING NEURAL NETWORK FOR ECG CLASSIFICATION

CHEN DZE RYNN

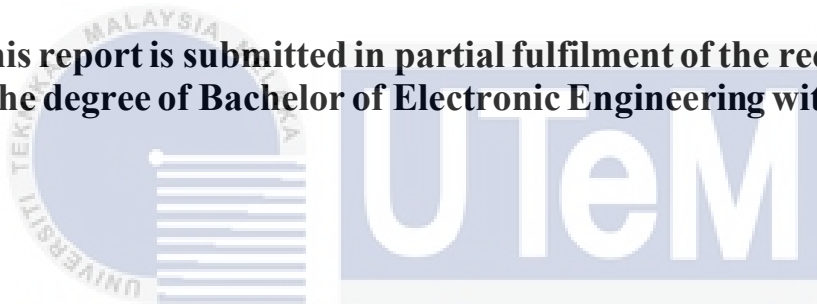


UNIVERSITI TEKNIKAL MALAYSIA MELAKA

**DIGITAL IMPLEMENTATION OF BIO-INSPIRED SPIKING  
NEURAL NETWORK FOR ECG CLASSIFICATION**

**CHEN DZE RYNN**

**This report is submitted in partial fulfilment of the requirements  
for the degree of Bachelor of Electronic Engineering with Honours**



**Faculty of Electronic and Computer Engineering  
Universiti Teknikal Malaysia Melaka**

**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2022**

## DECLARATION

I declare that this report entitled “Digital Implementation of Bio-Inspired Spiking Neural Network for ECG Classification” is the result of my own work except for quotes as cited in the references.



Signature : .....

Author : Chen Dze Rynn  
.....

Date : 11 January 2022  
.....

## APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Bachelor of Electronic Engineering with Honours.



اونيورسيتي تيكنيكل مليسيا ملاك

Signature : .....

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Supervisor Name : Assoc. Prof. Dr. Wong Yan Chiew

Date : 11 January 2022

## DEDICATION

For my late grandmother, whose strength continues to inspire me.



## ABSTRACT

Conventional techniques of off-chip processing for wearable devices cause higher hardware resource usage and power consumption. Hence, edge computing methods such as neuromorphic computing are considered the most promising modern technology to replace conventional processing. It is beneficial to employ neuromorphic processing in ECG classification, enabling engineers to overcome the constraints of hardware utilization. Thus, this work aims to investigate common building blocks in a spiking neural network (SNN), analyse the spike-based plasticity mechanism and implement ECG classification on a neuromorphic circuit. The MIT-BIH Arrhythmia database (MITDB) is used in this work, which is obtained from the Physionet website. The data is preprocessed in MATLAB, then used to train and test an SNN designed for field programmable gate arrays (FPGA), employing spike-based plasticity and Izhikevich neurons. The behaviour of spike timing dependent plasticity (STDP) in a neuromorphic circuit is also visualized in this work. The SNN classifies ECG data into two categories: normal and abnormal. The proposed digital design utilizes 1.058% of hardware resources on a Zedboard. Application-wise, this work provides a foundation for development of neuromorphic computing in wearable medical devices that perform continuous monitoring of ECG.

## ABSTRAK

*Teknik konvensional pemrosesan luar cip untuk peranti boleh pakai menyebabkan penggunaan sumber perkakasan yang lebih tinggi. Oleh itu, kaedah pengkomputeran neuromorfik dianggap sebagai teknologi moden yang paling berpotensi untuk menggantikan pemrosesan konvensional. Adalah berfaedah untuk menggunakan pemrosesan neuromorfik dalam klasifikasi ECG, membolehkan jurutera mengatasi kekangan penggunaan perkakasan. Oleh itu, projek ini bertujuan menyiasat teknik-teknik dalam rangkaian neural spiking (SNN), menganalisis mekanisme keplastikan berasaskan spike dan melaksanakan klasifikasi ECG pada litar neuromorfik. MIT-BIH Arrhythmia Database (MITDB) digunakan yang diperoleh daripada laman web Physionet. Data tersebut dipraproses dalam MATLAB, kemudian digunakan untuk melatih dan menguji SNN yang direka bentuk untuk field programmable gate array (FPGA), menggunakan keplastikan berasaskan spike dan neuron Izhikevich. Keberfungsian spike timing dependent plasticity (STDP) dalam litar neuromorfik juga divisualisasikan dalam kerja ini. SNN mengklasifikasikan data ECG kepada dua kategori: normal dan tidak normal. Reka bentuk digital yang dicadangkan menggunakan 1.058% sumber perkakasan pada Zedboard. Dari segi aplikasi, kerja ini boleh digunakan dalam peranti boleh pakai untuk pemantauan berterusan ECG.*

## ACKNOWLEDGEMENTS

I would like to extend my sincere gratitude to my supervisor, Associate Professor Dr. Wong Yan Chiew. I am so honored to have been taken under her wing, receiving her invaluable input and patient guidance throughout the completion of this research project. I have learned so much from her and cultivated an interest in hardware implementation along the way, which I hope to continue to pursue in the years after graduation.

I am deeply grateful as well to my parents, Mr Chen Thean Hock and Ms Chee Pooi Shan, for their unwavering support in the past year during which I have been working on this paper. All my life, they have believed in me more than I believed in myself. I would also like to offer special thanks to my extended family members. Without the support system they have provided, I would not be the person I am today.

Last but not least, I cannot forget to acknowledge my university mates as well as childhood friends. While we have all had our own struggles during the COVID-19 pandemic, they have never let me down when I needed a listening ear or a shoulder to cry on.



# TABLE OF CONTENTS

<b>Declaration</b>	
<b>Approval</b>	
<b>Dedication</b>	
<b>Abstract</b>	<b>i</b>
<b>Abstrak</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Table of Contents</b>	<b>iv</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Symbols and Abbreviations</b>	<b>xiv</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Overview	1
1.2 Background	1
1.3 Problem Statement	3
1.4 Objectives	4
1.5 Scope	4

1.6	Thesis Outline	5
1.7	Summary	5
<b>CHAPTER 2 BACKGROUND STUDY</b>		<b>7</b>
2.1	Overview	7
2.2	Electrocardiogram	8
2.2.1	Noise in ECG Signals	9
2.2.2	Filtering of ECG Signals	10
2.2.2.1	Morphological Filtering	11
2.2.2.2	Spike Encoding	12
2.2.2.3	Discrete Wavelet Transform	13
2.3	Deep Learning and Artificial Neural Networks	14
2.4	The Spiking Neural Network	16
2.4.1	Convolutional Neural Network versus Spiking Neural Network	16
2.4.2	SNN Models	20
2.4.2.1	Hodgkin-Huxley Model	20
2.4.2.2	Izhikevich Model	22
2.4.2.3	Leaky Integrate-and-Fire Model	24
2.5	Neuromorphic Computing	25
2.5.1	Intel Loihi	27
2.5.2	IBM TrueNorth	29

2.5.3	SpiNNaker	30
2.6	Design Implementation	31
2.6.1	Target Devices	32
2.6.2	FPGA Implementation in Previous Work	33
2.7	Summary	34
<b>CHAPTER 3 METHODOLOGY</b>		<b>35</b>
3.1	Overview	35
3.2	Data Collection	35
3.3	Data Preprocessing	37
3.3.1	Discrete Wavelet Transform	38
3.3.2	Normalization	43
3.3.3	Binarization	44
3.4	Spiking Neural Network	45
3.4.1	Neuron Model	45
3.4.2	Training	47
3.4.2.1	Spike Timing-Dependent Plasticity in the SNN	48
3.4.3	Testing	51
3.5	Design Synthesis and Implementation	51
3.5.1	ZedBoard Zynq-7000 ARM/FPGA SoC Development Board	51
3.6	Summary	52

<b>CHAPTER 4 RESULTS AND DISCUSSION</b>	<b>53</b>
4.1 Overview	53
4.2 Building Blocks of a Neuromorphic Design	53
4.2.1 Preprocessing Block	54
4.2.2 Training Block	59
4.2.3 Testing Block	60
4.3 Analysis of Spike-based Plasticity	62
4.4 Implementation on a Neuromorphic Circuit	63
4.4.1 Synthesis	64
4.4.2 Hardware Resource Utilization	67
4.4.3 Power Report	68
4.5 Performance Comparison with Previous Research	70
4.6 Summary	71
<b>CHAPTER 5 CONCLUSION AND FUTURE WORKS</b>	<b>72</b>
5.1 Overview	72
5.2 Conclusion	72
5.3 Future Works	74
5.3.1 Adaptive Peak Detection	74
5.3.2 Unified Power Format	75
5.3.3 Hardware Deployment	76

5.4	Lifelong Learning	77
5.5	Summary	77
	<b>REFERENCES</b>	<b>78</b>



## LIST OF FIGURES

Figure 2.1: General shape of an ECG waveform.	8
Figure 2.2: Visual representation of an ECG signal corrupted by power line interference [18].	10
Figure 2.3: A typical deep ANN network.	15
Figure 2.4: Neurons in the human brain whose operation is replicated in ANNs.	15
Figure 2.5: Different filters in the CNN extract different features in the image being processed.	17
Figure 2.6: An illustration of synaptic plasticity.	19
Figure 2.7: Varying complexity of different neuron models. [54]	20
Figure 2.8: Circuit representation of the HH model.	21
Figure 2.9: Different types of neurons resulting from changes in the parameters.	23
Figure 2.10: Schematic representation of the LIF model.	24
Figure 2.11: Graphical representation of Moore's law.	25
Figure 2.12: Relating various areas of study within neuromorphic computing.	26
Figure 2.13: Classifications of neuromorphic hardware.	27
Figure 2.14: Intel Loihi mesh operation enabling it to extend to other chips in four planar directions.	28
Figure 2.15: Major architectural entities of the Loihi's computational route.	29
Figure 2.16: Architecture of the TrueNorth neuromorphic chip.	30

Figure 2.17: Architecture of a SpiNNaker node.	31
Figure 2.18: Various categories of design implementation	32
Figure 2.19: Design process for a) an ASIC b) an FPGA	33
Figure 3.1: Abnormal rhythm annotation (left) and normal rhythm annotation (right).	37
Figure 3.2: Block diagram of the proposed ECG preprocessing process.	38
Figure 3.3: DWT process.	39
Figure 3.4: Filter bank trees for: (a) wavelet decomposition (b) wavelet reconstruction	39
Figure 3.5: Down sampling of ECG signals using Daubechies wavelet. [86]	41
Figure 3.6: Inputs of the DWT process.	42
Figure 3.7: Comparison of the results of DWT with different vanishing moments.	43
Figure 3.8: ECG signals before normalization (top row) and ECG signals after normalization (bottom row)	44
Figure 3.9: A portion of the analog ECG data shown in the MATLAB workspace after going through DWT.	44
Figure 3.10: Different threshold values are needed after the first round and second round of DWT respectively.	45
Figure 3.11: High-level view of the adapted digital IZH neuron module.	46
Figure 3.12: Internal block diagram of the module.	46
Figure 3.13: Overall operation of adapted IZH neuron module.	47
Figure 3.14: Digital implementation of the “v” equation (above) and the “u” equation (below) in the equations shown in Chapter 2.4.2.2.	47
Figure 3.15: STDP module modelled in the proposed SNN.	49
Figure 3.16: Hierarchy of the design sources in Vivado, where each neuron has its own RAM.	49
Figure 3.17: Internal block diagram of the STDP module.	50

Figure 3.18: Block diagram of the Zedboard.	52
Figure 4.1: Results of each preprocessing step performed in MATLAB for record 105 (normal).	54
Figure 4.2: Results of each preprocessing step performed in MATLAB for record 201 (abnormal).	55
Figure 4.3: The ECG signal for record 111 before any preprocessing.	56
Figure 4.4: The ECG signal for record 111 after being filtered by the db6 wavelet.	56
Figure 4.5: The binarized ECG signal for record 111 after being filtered by the db6 wavelet.	57
Figure 4.6: The ECG signal for record 111 after being filtered by the db6 and db2 wavelets.	57
Figure 4.7: Peak detection in Record 115 with a threshold of 0.7.	58
Figure 4.8: Peak detection in Record 111 with a threshold of 0.7.	58
Figure 4.9: Pseudocode for insertion of multiple data.	60
Figure 4.10: Pseudocode for increment of array pointer.	60
Figure 4.11: Insertion of training data automated by the counters moving through the normal and abnormal data arrays.	60
Figure 4.12: Neuron 37 spikes to classify record number 103 correctly as normal.	61
Figure 4.13: Neuron 38 spikes to classify record number 217 correctly as abnormal.	61
Figure 4.14: The incorrect output neuron randomly spikes while the correct output neuron is spiking.	61
Figure 4.15: STDP when the pre-synaptic neuron fires before the post-synaptic neuron does.	62
Figure 4.16: STDP when the pre-synaptic neuron fires after the post-synaptic neuron does.	63
Figure 4.17: RTL schematic of the synthesized design.	64



Figure 4.18: Close-up of gates in RTL schematic.	65
Figure 4.19: Technology schematic of the synthesized design.	66
Figure 4.20: Close-up of the Zedboard-specific elements shown in the technology schematic.	66
Figure 4.21: Graphical representation of hardware resource utilization percentages.	67
Figure 4.22: The stages of power consumption in an FPGA.	68
Figure 4.23: Post-synthesis power report.	69
Figure 4.24: Post-implementation power report.	69
Figure 4.25: Default switching activity settings in power estimation.	70
Figure 5.1: A visual representation of how a chip is divided into different power domains for power-gating.	76



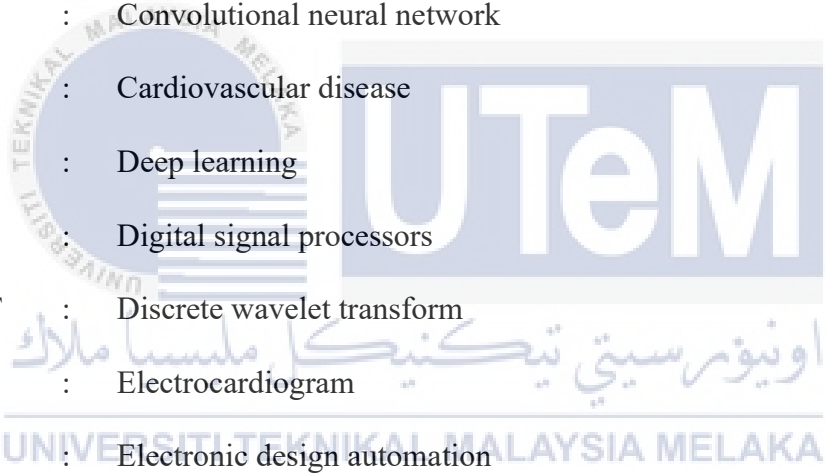
## LIST OF TABLES

Table 2.1: Past research using SNNs for ECG classification, without hardware implementation.	19
Table 2.2: Benchmarking of previous research implementing ECG classification on FPGA	34
Table 3.1: Meaning of beat annotations for the MIT-BIH Arrhythmia Database.	37
Table 4.1: Arrangement of training data in arrays.	59
Table 4.2: Hardware resource utilization report.	67
Table 4.3: Comparison of proposed design with previous research.	71

اونيورسيتي تيكنيكل مليسيا ملاك

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

## LIST OF SYMBOLS AND ABBREVIATIONS



ANN	:	Artificial neural network
ASIC	:	Application-specific integrated circuit
BSA	:	Ben's Spiker Algorithm
CNN	:	Convolutional neural network
CVD	:	Cardiovascular disease
DL	:	Deep learning
DSP	:	Digital signal processors
DWT	:	Discrete wavelet transform
ECG	:	Electrocardiogram
EDA	:	Electronic design automation
EMG	:	Electromyography
FPGA	:	Field-programmable gate array
HDL	:	Hardware description language
HH	:	Hodgkin-Huxley
HSA	:	Hough Spiker Algorithm
IDE	:	Integrated design environment
IO	:	Bonded input/output
IZH	:	Izhikevich

LIF	:	Leaky Integrate-and-Fire
mHSA	:	Modified Hough Spiker Algorithm
MI	:	Myocardial infarction
MITDB	:	MIT-BIH Arrhythmia Database
MNIST	:	Modified National Institute of Standards and Technology
NoC	:	Network-on-chip
PL	:	Programmable logic
PS	:	Processor subsystem
RTL	:	Register transfer level
SAIF	:	Simulation activity file
SNN	:	Spiking neural network
SNR	:	Signal-to-noise ratio
SoC	:	System-on-chip
STDP	:	Spike-timing dependent plasticity
t-LTD	:	Timing-dependent long-term depression
t-LTP	:	Timing-dependent long-term potentiation
UPF	:	Unified Power Format
WT	:	Wavelet transform

# CHAPTER 1

## INTRODUCTION



### 1.1 Overview

This chapter explains the key points of this work to provide some basic information about why this work was initiated. The topics covered include background in Section 1.2, problem statement in Section 1.3, objectives in Section 1.4, scope in Section 1.5 and thesis outline in Section 1.6.

### 1.2 Background

Cardiovascular diseases (CVDs) are a category of disorders in the human body involving the heart and blood vessels. According to the World Health Organization, CVDs are the number 1 cause of deaths globally, with around 17.9 million people losing their lives to CVDs in 2016. Since the COVID-19 pandemic in 2020, it has been proven that people with underlying CVDs are disproportionately attacked by COVID-

19 [1]. Cardiac arrhythmias refer to the impairment of the electrical impulses coordinating human heartbeats and are used to identify the presence of CVDs. Due to the nature of arrhythmias that can reflect electrical activities in the heart, they can be detected by analyzing electrocardiogram (ECG) signals taken from the body [2]. To do this, the conventional way was for medical professionals to manually inspect results of an ECG test. However, arrhythmias occur intermittently, and thus are difficult to detect based solely on ECG tests. Therefore, continuous monitoring of ECGs is crucial in early detection of potential cardiovascular problems.

With the invention of wearable personal devices such as smart watches, it is now possible to have continuous real-time ECG monitoring with the mechanism of artificial neural networks (ANNs) for ECG classification. Currently, convolutional neural networks (CNNs) are the most common form of neural networks used for recognition and classification tasks. Previous research has proven the effectiveness of CNN for ECG classification [3]. However, the discrete approximation method frequently replaces the precise convolution process itself in CNN [4] to calculate the dot product of overlapping areas between different layers. Due to this, the CNN consumes relatively high computational power [5] and methods to reduce power consumption for CNN such as the fast Fourier transform will drastically increase the cost. It was found in Ref. [6] that the higher the accuracy of the CNN, the higher its power consumption.

Thus, in recent years, researchers have been looking into alternative ANNs to CNNs to find a fair power-accuracy trade-off [7] in classification applications. Spiking neural networks (SNNs) have become a strong contender in these researches, such as in [8] which found that the power consumed by the algorithm that was

converted from a trained convolutional neural network into a spiking neural network was only 0.74% of that of the convolutional neural network. Employing bio-inspired neural networks in processors is an emerging area of research, which is neuromorphic computing. Neuromorphic computing essentially models neurobiology in its operation, replicating structures of the human brain to enable the system to learn on its own [9].

### 1.3 Problem Statement

The current norm for processing techniques used in wearable devices is employing Von Neumann architectures. Throughout the years, continuous research and progress has improved the sensitivity and resolution of wearable devices. However, these devices are still limited due to high volume of data transmission. In status quo, most wearable devices employ the mechanism of collecting data, then transmitting it to external servers which perform off-chip processing [10]. There are several problems with this.

Firstly, conventional techniques of using remote servers and signal processing requires intensive computation and processing, causing higher hardware resource usage. Due to this, edge computing and other new computing methods or have become popular areas of research. Particularly, neuromorphic circuits are considered the most promising modern technology beyond Von Neumann processing [11]. SNNs used in neuromorphic computing do not employ the fetch-and-execute cycle of Von Neumann architectures but rather process data in the form of event-driven spikes, with an emerging learning mechanism which is spike-based plasticity. In applications such as ECG classification, it is beneficial to employ processing techniques that are able to overcome the constraints of hardware utilization, allowing further development of this

technology in the future. However, a jarring downside of neuromorphic circuits is that they are not general-purpose, but rather need to be designed specifically, or customized for their applications. The customization of neuromorphic circuits for the purpose of ECG classification on FPGAs is an area that lacks research, thus this work aims to fill that gap.

#### **1.4 Objectives**

- a) To investigate common building blocks and techniques used for a neuromorphic circuit based on an SNN.
- b) To analyze spike-based plasticity mechanism in neuromorphic circuit.
- c) To implement ECG classification on a neuromorphic circuit.

#### **1.5 Scope**

This work aims to propose a digital design for implementation of neuromorphic computing on an FPGA. An SNN algorithm is developed for the purpose of ECG classification which employs spike-based plasticity to train neurons. Using electronic design automation (EDA) tools which are Vivado as well as MATLAB, raw ECG data is obtained from the MIT-BIH Arrhythmia database (MITDB), which is obtained via Physionet. The database contains 48 ECG records taken on two leads, which are the MLII lead and V5 lead. Only the data from the MLII lead is used, which accurately depicts the overall condition of the ECG record. In this work, the duration of records taken is 10 seconds long containing 3500 samples. This may present limitations due to the short length of the records, however it ensures that the preprocessing step does not over compress the data. This scope may be widened in future works as the design is very much portable and can be used on a much bigger scale. The raw ECG data



undergoes preprocessing in MATLAB and is compressed into 35-bit binary stream so that it can be fed into the SNN.

The SNN algorithm is developed based on adaptations of methods used in past research on developing SNNs. With 35 normal records and 13 abnormal records for the 10-second-long MITDB records, the algorithm is trained to identify normal and abnormal heartbeat rhythms by using 10 records for training and 3 records for testing in each category. The algorithm is developed in a hardware description language (HDL) which is VHDL, so that it can be synthesized and optimized for the target field programmable gate array (FPGA) which is Zedboard. No actual hardware is used in this research. The design is represented in the form of a neuromorphic circuit, which is then analyzed in terms of hardware resource utilization.

## 1.6 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 will cover the literature review of past research related to ECG classification, SNNs and neuromorphic hardware. Chapter 3 will highlight the methodology applied to achieve the objectives mentioned in Chapter 1. Chapter 4 will lay out the results and findings of the research performed, and the algorithm developed. Finally, Chapter 5 will conclude the findings of this research and elaborate on potential future work.

## 1.7 Summary

This work was initiated to address the research gap in designing neuromorphic circuit customized for ECG classification. The problem lies in the unsustainability of off-chip processing used in the Von Neumann architecture in wearable devices when it comes to hardware resource usage. This paper aims to investigate common building blocks and techniques used for a neuromorphic circuit based on SNN, analyze spike-

based plasticity mechanism in neuromorphic circuit, and implement ECG classification on a neuromorphic circuit. The neuromorphic design created is synthesized for implementation on a Zedboard and analyzed in terms of hardware resource utilization.



## CHAPTER 2

### BACKGROUND STUDY

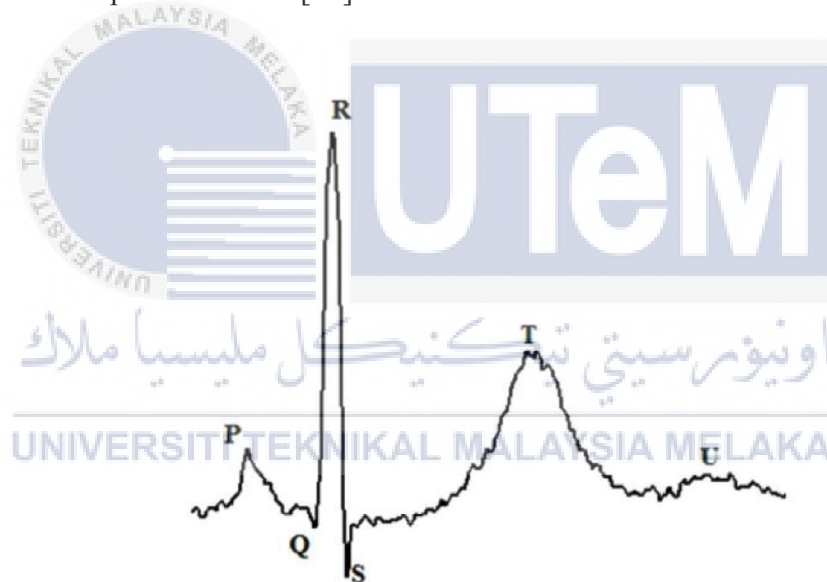


#### 2.1 Overview

This chapter details the footprints that have been left in the area of research by previous authors, highlighting the main elements that will be adopted in this work. Section 2.2 describes the ECG as well as its need to be preprocessed and the preprocessing methods. Section 2.3 and Section 2.4 outline deep learning and SNN respectively, with the latter describing neuron models that have been used in previous works. Section 2.5 discusses neuromorphic computing in the electronics industry, while Section 2.6 compares previous papers that have performed ECG classification on FPGAs.

## 2.2 Electrocardiogram

ECGs allow medical professionals to observe the state of human heart, and thus are a common diagnostic method used in the medical field to identify heart diseases in patients. ECGs are measured by attaching electrodes to the surface of the patient's skin. Underneath the skin, a path called a 'lead' is formed between the conducting electrodes. The ECG is basically a graphical representation of the potential difference [12] detected between the electrodes. ECGs are periodic waveforms consisting of a P wave, a QRS complex, a T wave, and a U wave as shown in Figure 2.1. The shape of these signals enables a medical professional to obtain important information about the condition of a patient's heart [13].



**Figure 2.1: General shape of an ECG waveform.**

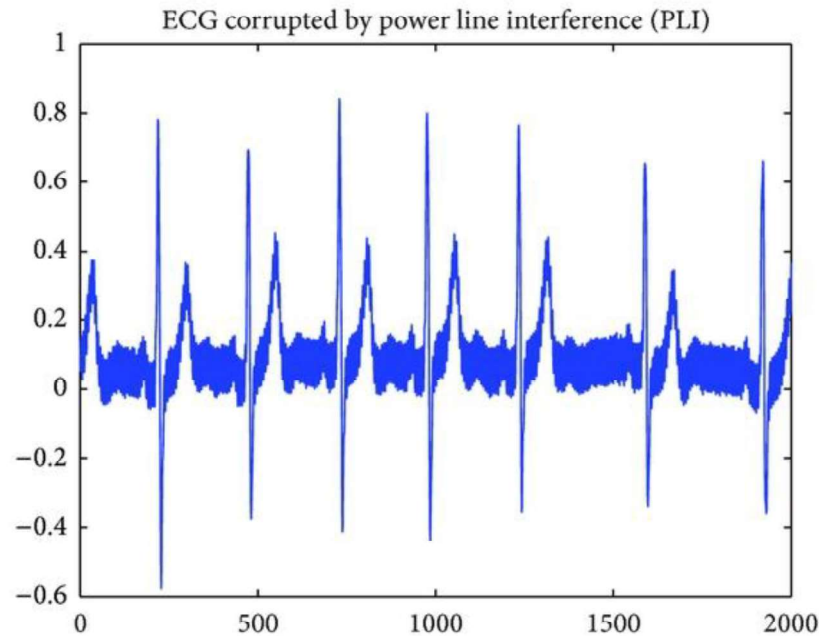
In recent years, researchers have begun to pursue developing computer-aided systems to automatically diagnose ECG [14], [15] where the goal is to automatically detect cardiac arrhythmia which leads to myocardial infarction (MI), also known as a heart attack, caused by the inability of the heart muscles to contract normally.

With the intended application of the SNN for classifying ECG data, several methods are explored in order to enable the SNN to read the ECG data. This includes preprocessing steps to obtain a clean ECG signal, as well as spike encoding which is a method specifically needed for SNNs.

### 2.2.1 Noise in ECG Signals

As with any signal, ECG signals are prone to the presence of noise, which are additional unwanted information present in a recorded signal. Several noise sources are common [16] in ECG signals, such as power line interference, electromyography (EMG) interference, and baseline drift noise.

Power line interference is characterized by sinusoidal signals [17] between 50 to 60Hz, depending on the power line frequency in different countries. These cause a low-amplitude interference in ECG signals, and thus affect the visual representation of ECG data. Due to this, power line interferences tend to interfere with automatic segmentation processes that take place for ECG signal processing. An example of power line interference is observed in Figure 2.2, where it is difficult to identify the regions of P-waves and T-waves.



**Figure 2.2: Visual representation of an ECG signal corrupted by power line interference [18]**

Baseline wander in ECG signals is a low-frequency noise artifact. It can be caused by respiration, body movement, or electrically charged electrodes and have a varying electrical isoline [19]. The removal of baseline wander is usually one of the first steps in preprocessing ECG signals. It is important not only for automatic ECG classification algorithms, but also in manual visual diagnosis. This is because drastic fluctuations in the ST segment, be it elevation or depression, are telling factors in an ECG signal when dealing with acute coronary syndrome caused by ischemia or myocardial infarction [20].

### 2.2.2 Filtering of ECG Signals

In using raw ECG data, it is important to put the data through some form of preprocessing to remove any impurities or discrepancies that could affect the usefulness of the data. Traditionally, preprocessing ECG signals can be done by applying high-pass or low-pass filters to them [21] depending on the type of noise that

is being targeted. For example, baseline wander is usually a very low frequency noise, in which a high-pass filter would be applied to the ECG signal to filter it out. However, recent works have explored more conventional ways of denoising ECG signals, mainly due to the fact that the high- and low-pass filters have very specific cut-off frequencies, which often result in the input signal being distorted above or below a certain frequency. These options will be explored further in this section. To make the ECG signal usable in an SNN, the preprocessing stage also requires a step to ensure the ECG signal is in a form that is readable by the network.

### 2.2.2.1 Morphological Filtering

Morphological filtering is a non-linear method of signal processing. In denoising ECG signals, non-linear methods are preferred [22] over linear methods because the ECG signal itself is non-linear. The main advantage of this method is that it is able to decently maintain the shape information of the input signal [23]. This is because the concept of mathematical morphology is set-theoretic, which means that it is able to quantitate the geometrical properties of a signal or image being analyzed. Structuring elements are a key unit in mathematical morphology. They are designed specifically for the signal of interest, depending on the shape characteristics to be extracted from the signal [24]. The structuring element is used to analyze how its shape fits or does not fit the shape of the signal, thus extracting the shape information.

There are two operators used in morphological filtering, which are erosion and dilation. ECG signals are one-dimensional. Thus, based on the erosion and dilation operators, the opening operator is used to suppress peaks, and the closing operation is to suppress pits. The morphological filtering is performed by combining both the opening and closing operations [23]. Specific peaks and pits of the ECG signal can

be removed or filtered out by essentially ‘moving’ the opening operator across the peaks and the closing operator across the pits of the ECG signal, giving the result of the highest peaks reached by the opening operator and lowest pits reached by the closing operator [22]. The width of the structural element needs to be wider than the noise waveform and narrower than the signal waveform. Thus, morphological filtering is commonly used for removing baseline wander from ECG signals, which are very low frequency waveforms.

### 2.2.2.2 Spike Encoding

Most real-world signals are analog in nature. Typically, ANNs use matrix-vector operations within their networks to process a set of input and produce an output. This means that they are able to operate directly on the numerical forms that the input data is in. However, SNNs are not able to do this. Thus, the input values need to be converted into the form of spikes or spike events [25] in the process called spike encoding. This process considerably compresses the size of the input data, since the train of spikes allows for faster processing in the SNN [26]. Spike encoding can be done based on instantaneous average firing rate, population rank, or temporal coding. The two main methods of spike encoding are temporal coding and rate coding. Rate coding converts the input into spike trains with frequency proportional to the values, while temporal coding converts it into precise spike times [27]. Temporal coding is a widely used mechanism in deep learning algorithms, especially those employing spike-timing dependent plasticity (STDP) [28].

In general, existing temporal coding methods can be categorized into two types, which are temporal contrast and stimulus estimation. The theory applied in temporal contrast is that every change in signal value is compared with a specific threshold



value, and the encoded spike is generated accordingly [29]. The temporal contrast algorithm is inspired by the human retina. As for stimulus estimation, the focus is generating a spike train that best approximates the original input when it is reconstructed. Common implementation methods for this reversal or deconvolution process are the Hough Spiker Algorithm (HSA) [30], the modified HSA (mHSA) and Ben's Spiker Algorithm (BSA) [31]. In Ref. [31], the function for the signal-to-noise ratio (SNR) against threshold of the BSA is smoother than in the case of HSA, which shows that the optimal threshold is stable and does not fluctuate with a change in test signals.

### 2.2.2.3 Discrete Wavelet Transform

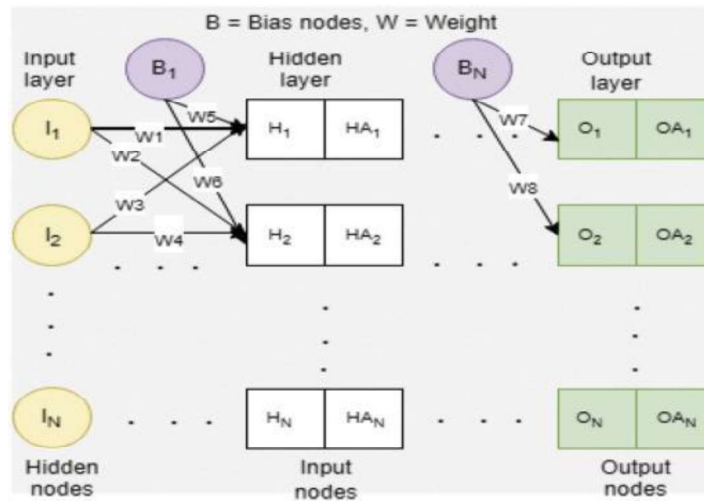
The theory of using wavelets encapsulates various independently developed techniques for ECG preprocessing. Recently, many wavelet transform (WT)-based methods have been proposed for signal processing, especially for non-stationary signals like ECG because it is more efficient in preserving important information in the signal. With a continuous input signal such as ECG, it is common practice to discretize it [32] before processing because it is computationally impossible to analyze a continuous signal using all wavelet coefficients. Recent studies [33], [34] have shown the effectiveness of the DWT. Unlike the Fourier transform, the DWT domain is able to describe both frequency and time representation. Another way to interpret the process of performing WT is as a tool for signal decomposition. In Ref. [35] and [36], the DWT is used to decompose the noisy ECG signal, before removing the noisy elements and reconstructing it.

There are several wavelet families that are available in the DWT method, namely the Daubechies family, Coiflet family, Haar family, Biorthogonal family and Symmlet

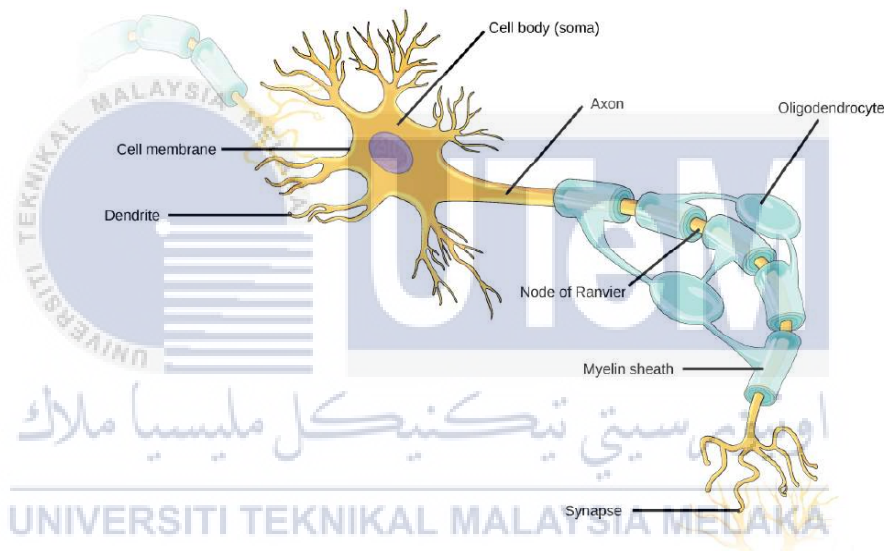
family. Several papers have studied all five wavelet families in relation to the optimum wavelet to be used for ECG processing. According to Ref. [37], the nominal wavelet for MIT-BIH database is the Daubechies wavelet family. Ref. [38] found that Daubechies wavelet to be able to accurately filter ECG signals with its ability to retrieve lost data, reducing distortion of the compressed ECG data. Based on its simplicity and lack of distortion, the DWT method is employed for preprocessing of ECG data in this work.

### 2.3 Deep Learning and Artificial Neural Networks

Deep learning (DL) is an umbrella term covering a research field that studies the process of extracting knowledge or data, and forecasting or making predictions by recognizing patterns and trends in a set of training data [13]. With this identification of trends and patterns in DL, the significant steps in conventional techniques for machine learning such as feature extraction, feature selection, and classification are somewhat applied, but there is not as clear a distinction in the steps [39], as the DL model is able to 'self-learn' based on the training data being fed to it. ANNs, as the name suggests, are complex networks that attempt to replicate the mechanism used by the human brain to transmit and process information from one neuron to another, demonstrated by the architecture in Figure 2.3. In the brain, this is done through a lot of connected processors called neurons as shown in Figure 2.4.



**Figure 2.3: A typical deep ANN network**



**Figure 2.4: Neurons in the human brain whose operation is replicated in ANNs.**

The features of neurons that are emulated are mainly the cell body that drives cellular activity, the dendrites that receive impulses and the axon which sends impulses to the next neuron. Biologically, the vital characteristic of neurons that makes them utilizable are their ability to conduct electrical impulses and receive and send information through these signals [40]. In NNs, input neurons receive data from the surrounding environment and activate the consecutive neurons connected to it through

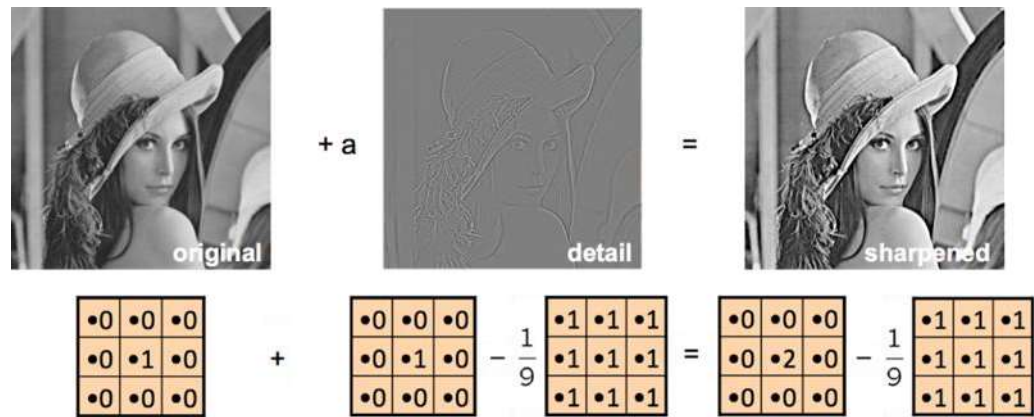
weighted connections. Recently, more modern NN models are being innovated and researched [41], especially machine and deep learning methods such as CNN.

## **2.4 The Spiking Neural Network**

In modelling an SNN, multiple factors contribute to the distinction of the network itself from other SNNs that have been developed over the years. In designing an SNN that can efficiently accomplish the task of classifying ECG data, each of these factors are thoroughly analyzed to choose the best combination of features to incorporate into the design.

### **2.4.1 Convolutional Neural Network versus Spiking Neural Network**

CNNs are one of the most popular deep neural learning architectures [42]. CNNs apply the mathematical convolution operation and consist of many consecutive layers or ‘building blocks’, connected in a feedforward manner. The main layers include the convolutional, normalization and pooling layers used for feature extracting, and the fully-connected layer for classification of data [43]. Every single one of these layers can be utilized for different ‘filters’ on the input, thus enabling different desired features to be extracted as shown in Figure 2.5.



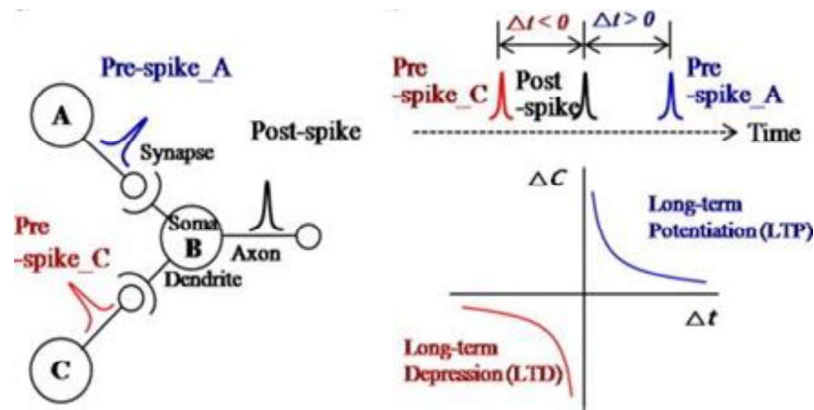
**Figure 2.5: Different filters in the CNN extract different features in the image being processed.**

The application of convolution in this NN technique allows it to recognize images that are not spatially dependent [42]. To reduce connection size when connecting all the neurons, neurons in the inner layers only receive data from the corresponding part of its previous layer, giving it the name ‘convolution’. With this mechanism, CNN is able to detect patterns, for example, faces, in any position in a given image.

According to Ref. [44], the main disadvantage of existing NNs up to CNNs is that they do not seem to be quite similar enough to biological neural networks in that biological neural networks represent information by a series of instantaneous spikes, where the value carried by the information is proportional to the frequency of the spikes. With the success of emulating features of biological neural networks, naturally researchers started moving into the emulation of the spiking behavior. In applications such as the one proposed in this paper, which is for wearable devices, the key is for an algorithm to be power-efficient due to the size and portability of the device. In Ref. [45], CNNs were found to consume up to 96.82% more power than an SNN in supervised learning applications.

SNNs are considered the third generation of ANNs [46], after McCulloch-Pitts neurons and continuous activation. SNNs were originally modeled as a means of biological signal processing in the human brain, where data and information are passed around through neurons via spikes. However, all the spikes in spiking neural networks are the same. Thus, the spikes themselves do not carry any information but it is the parameters surrounding it, which are the number and timing of their occurrences that contain information in them. There are various SNN models such as the leaky integrate-and-fire (LIF), Hodgkin-Huxley (HH) and Izhikevich (IZH) models. Information processing in SNN depends on the timing of the spikes [47], which will influence the weights of the synapse connection in transmitting the information to other neurons.

SNNs employ STDP to operate and carry out pattern recognition and computation. The transfer and storing of signals are determined by the strength of the connection of the synapses, also known as the synaptic plasticity. Modifications of this synaptic connectivity based on activity is the core of the human brain's learning process [48] as well as memory. A lot of the research done on synaptic plasticity was pioneered by Donald Hebb and proven by more recent research [49] that repetitive simulation of a presynaptic cell immediately before spikes occur in a postsynaptic cell will cause synaptic strengthening called timing-dependent long-term potentiation (t-LTP). Vice versa, experiments also hypothesize that repetitive simulation of a presynaptic cell immediately after a spike occurs in a postsynaptic cell will result in timing-dependent long-term depression (t-LTD). Together, these synaptic phenomena are collectively recognized as spike-timing-dependent plasticity [49] as shown in Figure 2.6.



**Figure 2.6: An illustration of synaptic plasticity.**

Due to the nature of the spikes which are only generated when a threshold of potential is exceeded, the SNN provides a low-power operation [50] compared to other ANN techniques, making it suitable for this application. In fact, the use of SNNs for electrocardiogram classification has been explored in multiple research papers as of late, as listed in Table 2.1.

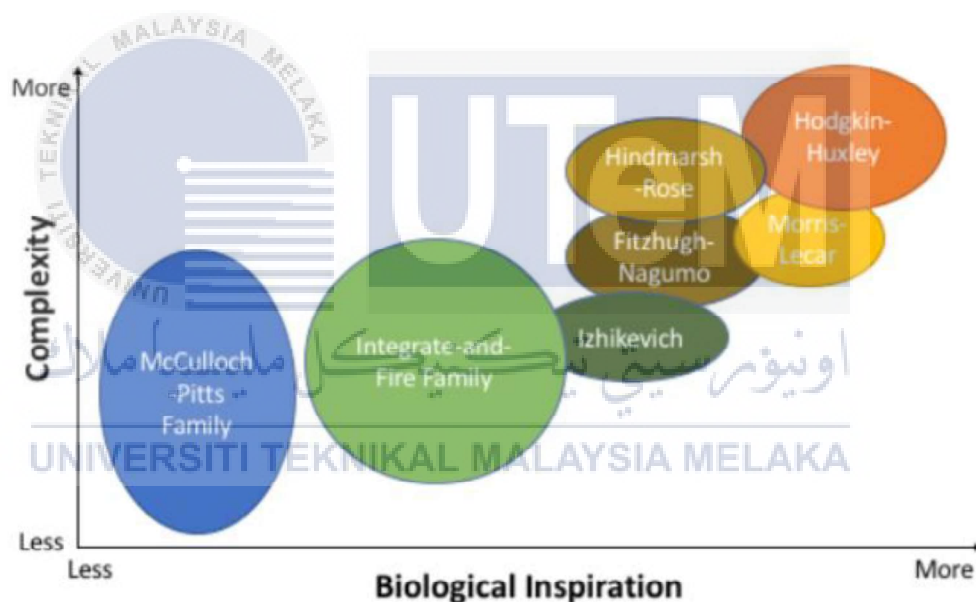
**Table 2.1: Past research using SNNs for ECG classification, without hardware implementation.**

Paper	ANN type	SNN model	Preprocessing	Training database	Abnormal categories
Yan <i>et al.</i> [8] (2021)	SNN, CNN	LIF	Spike rate encoding	MIT-BIH Arrhythmia Database	4
Rana <i>et al.</i> [13] (2021)	SNN	LIF	Probabilistic encoding	PTB Diagnostic ECG Database	1
Bauer <i>et al.</i> [51] (2019)	RNN	-	Sigma-delta encoding	MIT-BIH Arrhythmia Database	5
Amirshahi <i>et al.</i> [52] (2019)	SNN	LIF	Gaussian layer	MIT-BIH Arrhythmia Database	1
Kolağasioglu [53] (2018)	SNN	LSM	Gaussian layer	MIT-BIH Arrhythmia Database	16



## 2.4.2 SNN Models

Fundamentally, information processing in SNN depends on the timing and synchronization of the spikes. However, there are several neural models to be considered in modelling the neuronal behavior in an SNN. These models vary in complexity depending on how closely they are modelled to resemble the human brain activity. Three of the most often used models are highlighted which are the Hodgkin-Huxley, Izhikevich, Wilson, Fitzhugh-Nagumo, and Leaky Integrate-and-Fire models. In this section, each neural model will be reviewed theoretically as well as in terms of their mathematical behavior and computational cost.



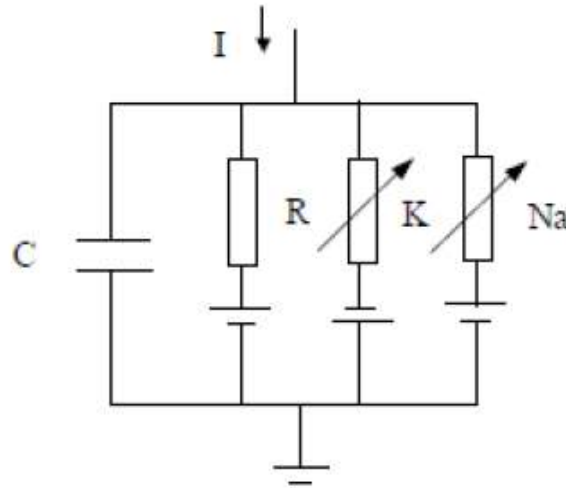
**Figure 2.7: Varying complexity of different neuron models. [54]**

### 2.4.2.1 Hodgkin-Huxley Model

The HH model was the first biologically relevant mathematical neuron model to be introduced in 1952 by Hodgkin and Huxley [55] while investigating the ionic excitation on the axon of a squid. Among all neural models that have been developed since then, the HH model is still considered to be one of the most accurate representations of electrophysiological neuronal activity [56]. Most of the other neural



models were developed as simplified versions or extensions of the HH model. The schematic representation of the HH model is shown in Figure 2.8.



**Figure 2.8: Circuit representation of the HH model.**

The HH model describes the semi-permeable membrane of the neuron as a capacitor which stores charge, due to the difference in ion concentration within the neuron and in the extracellular fluid outside it. The idea is that the membrane conserves charge during the transport of ions across the membrane. Thus, Eq. 2.1 is derived, equating the total applied current to a neuron, to the summation of the capacitive current in the membrane and current from presynaptic neurons.

$$I(t) = I_{cap}(t) + \sum_k I_k(t) \quad \text{Eq. 2.1}$$

Applying the general equation of capacitance in Eq. 2.2, the variable  $u$  symbolizes the total current across the membrane. Thus, the overall equation of Hodgkin and Huxley's findings is simplified in Eq. 2.3 [47] where  $E_{Na}$ ,  $E_K$  and  $E_L$  are reversal potentials, and  $g_{Na}m^3h$ ,  $g_Kn^4$  and  $g_L$  are channel conductance.

$$C \frac{du}{dt} = - \sum_k I_k(t) + I(t) \quad \text{Eq. 2.2}$$

$$\sum_k I_k(t) = g_{Na}m^3h(u - E_{Na}) + g_Kn^4h(u - E_K) + g_L(u - E_L) \quad \text{Eq. 2.3}$$

Hardware implementation of the HH model are rare [57], due to its high computational cost. While recent studies [58] have attempted to dispute this, the hardware implementations they have proposed are based on simplified versions of the HH model, and only model the behavior of a single neuron [57], not an entire SNN.

#### 2.4.2.2 Izhikevich Model

With the main drawback of the HH model being that it is computationally expensive and thus cannot be practically implemented on a larger scale, the IZH model [59] was developed for precisely this purpose. Combining the biological veracity of neurons in the HH model and the computational advantages of LIF neurons, the author, Eugene Izhikevich aimed to enable the simulation of tens of thousands of IZH neurons using a desktop PC. Theoretically, the behavior of an IZH neuron adopts an ‘all-or-nothing’ behavior.

The synaptic currents from presynaptic neurons increase the neuron’s membrane potential when they arrive. This then introduces two possible outcomes. The first is if the total presynaptic currents arriving at the membrane are not sufficient to cause the neuron to spike, then the membrane voltage is reset to its original value before any of the currents arrived. The second possibility is if the total presynaptic currents do in fact suffice to cause a spike, then the membrane voltage,  $v$  and the recovery variable,  $u$  are reset.

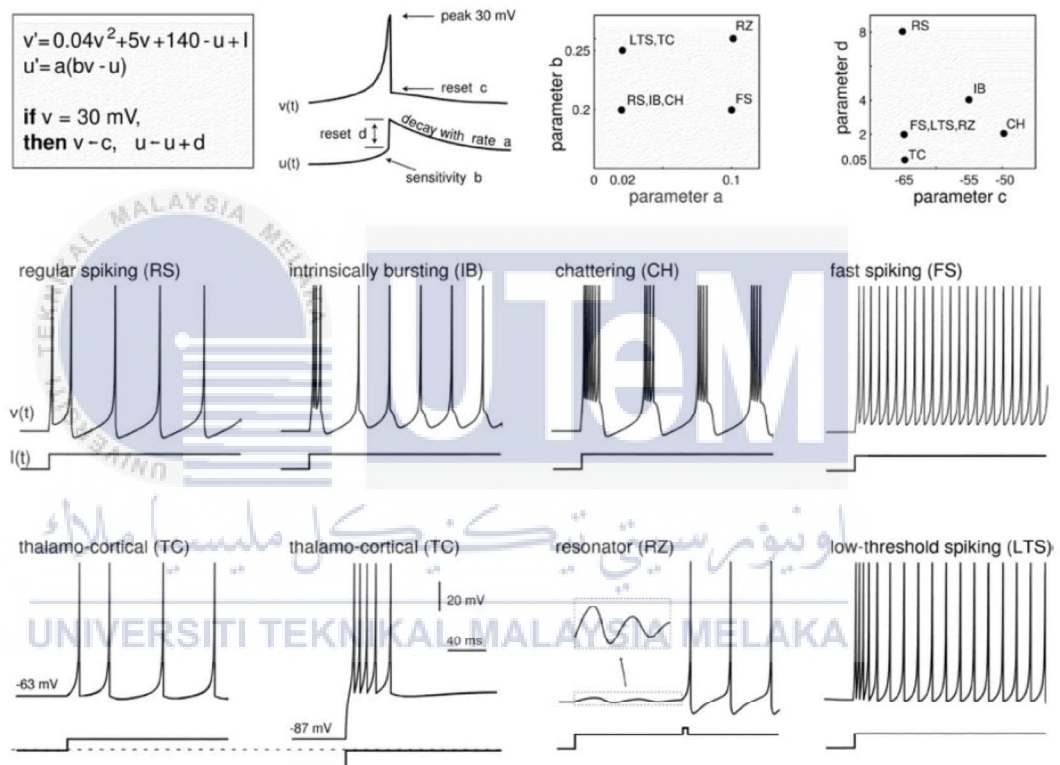
Mathematically, the IZH model is much simpler, consisting of Eqs. 2.4, 2.5 and 2.6 where  $v$  and  $u$  are dimensionless variables and  $a$ ,  $b$ ,  $c$  and  $d$  are dimensionless parameters. Parameter  $a$  (typical value = 0.02) is related to the recovery of the variable  $u$  in terms of time. Parameter  $b$  (typical value = 0.2) shows the sensitivity of variable  $u$  to the subthreshold changes in variable  $v$ . Parameter  $c$  (typical value = -65mV)

describes the post-spike reset value of  $v$ . Parameter  $d$  (typical value = 8) shows the post-spike reset value of  $u$ . The effects of these parameters and variables are illustrated in Figure 2.9 [47].

$$v' = 0.04v^2 + 5v + 140 - u + I \quad \text{Eq. 2.4}$$

$$u' = a(bv - u) \quad \text{Eq. 2.5}$$

$$\text{If } v \geq 30\text{mV, then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \quad \text{Eq. 2.6}$$



**Figure 2.9: Different types of neurons resulting from changes in the parameters.**

Recent studies [60],[61] show successful low-power implementations of the Izhikevich model on hardware, proving that the Izhikevich model is indeed practical for complex hardware implementation as intended by Izhikevich himself.

### 2.4.2.3 Leaky Integrate-and-Fire Model

In 1907, Louis Lapicque conducted a [62] based on the excitability of nerves by electrical impulses. Based on these findings, the LIF model was proposed in the 1960s [63]. The LIF is easily one of the simplest established neural models. The LIF model describes the behavior of the neuron in which the voltage on the neuron's membrane increases with time when current arrives at it. The voltage at the membrane keeps increasing with the arrival of current until the membrane voltage reaches a threshold value, at which the neuron spikes then resets to the original value. This is mathematically described in Eq. 2.7 where  $I(t)$  is the current arriving at the membrane,  $C_m$  is the membrane capacitance and  $V_m(t)$  is the membrane potential. Figure 2.10 shows the schematic representation of the LIF model, where a so-called leaky resistor (conductance,  $g_{\text{leak}}$ ) and a capacitor are connected in parallel and  $i_{\text{inject}}$  represents  $I(t)$  in Eq. 2.7.

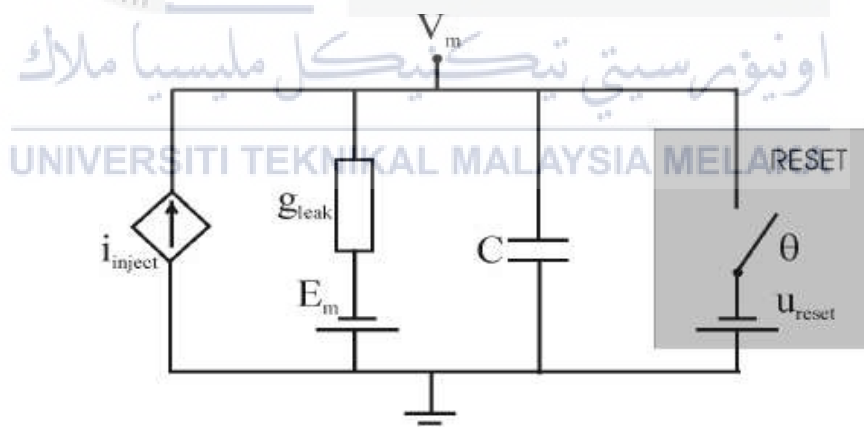


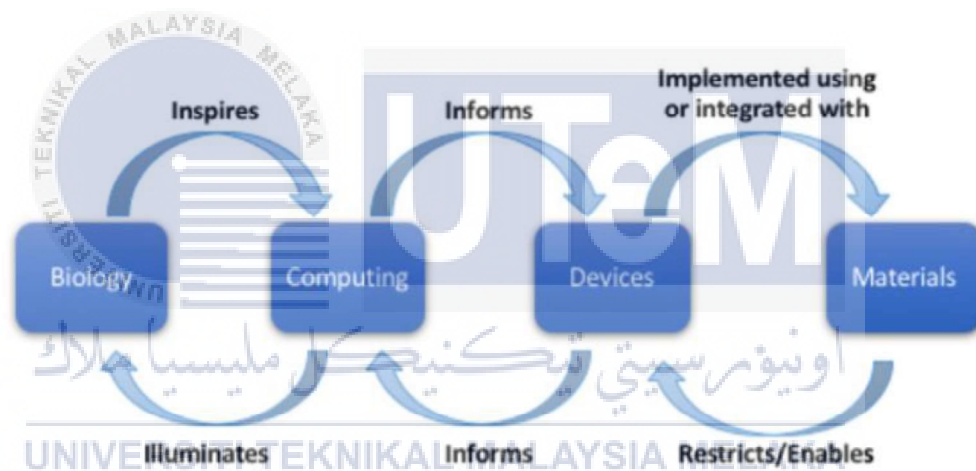
Figure 2.10: Schematic representation of the LIF model.

$$I(t) = C_m \frac{dV_m(t)}{dt}$$

Eq. 2.7



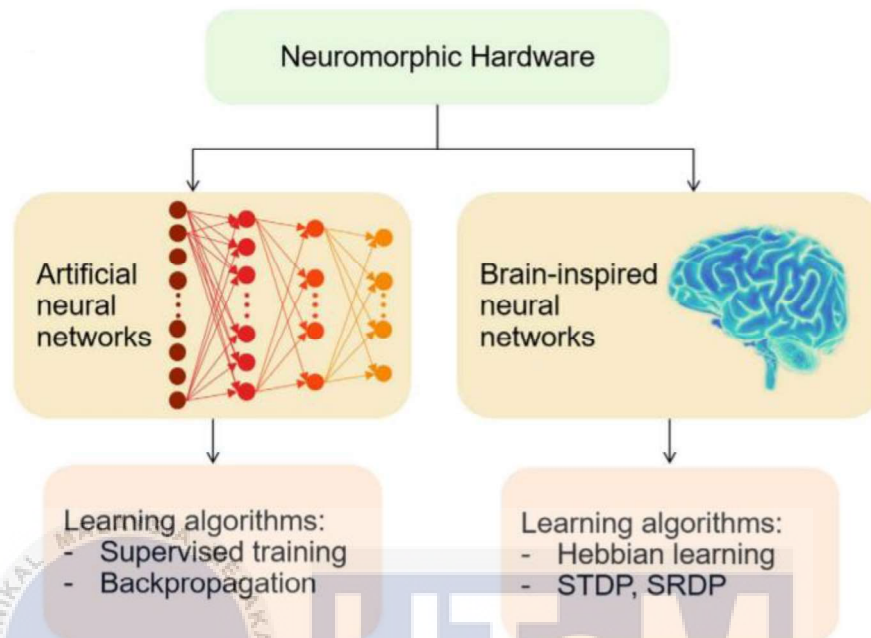
Neuromorphic computing is one of the more advanced techniques being explored. A defining feature of neuromorphic computing is the emulation of the human brain's behavior to complete tasks. It is typically employed in tasks involving cognitive abilities such as recognition and classification using algorithms to replicate neuronal behavior in the human brain. Neuromorphic computing sends information through networks of synapses, in line with McCulloch and Pitts' description of neurons being an arithmetic function of its synapses [66]. Various areas of study exist within the scope of neuromorphic computing, as illustrated in Figure 2.12 [54] closely tying it to biological studies.



**Figure 2.12: Relating various areas of study within neuromorphic computing.**

Neuromorphic hardware can be divided into two types as shown in Figure 2.13 [67], where ANNs and brain-inspired neural networks are placed in different categories. While ANNs are also inspired by human brain activity in terms of their working principle, the main difference between ANNs and brain-inspired networks is the training methodology, where the latter employs learning mechanisms that are also derived from neurobiological systems, such as STDP discussed in Section 2.3.1. On

the other hand, training methods for the ANN use algorithms not modeled after neurobiological systems.



**Figure 2.13: Classifications of neuromorphic hardware.**

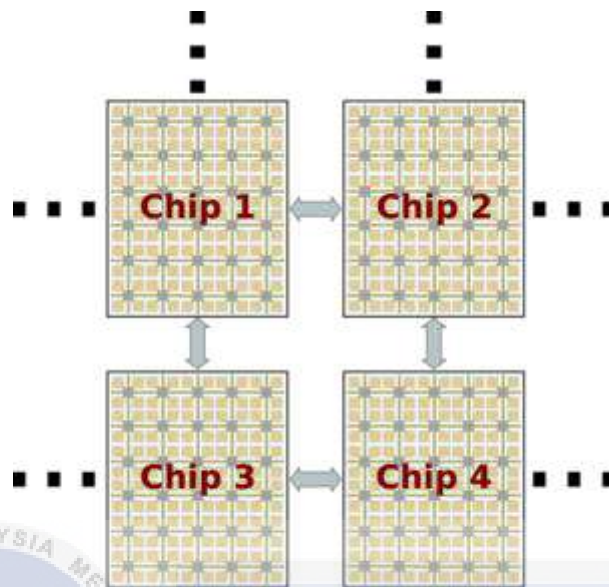
The rise of neuromorphic hardware in recent years is drastic, although the field of study has been around for over a decade now. Major semiconductor companies such as Intel and IBM have produced their own versions of neuromorphic chips, namely Intel's Loihi [68] and IBM's TrueNorth chip [69]. Smaller-scale neuromorphic chips exist as well and are widely used within their field of study, such as the SpiNNaker chip and the DYNAP-SE.

### 2.5.1 Intel Loihi

The Loihi chip is a  $60\text{mm}^2$  neuromorphic chip with a manycore mesh [68]. It contains 128 neuromorphic cores and three processor cores, and the mesh is able to be extended to other chips in all four directions as shown in Figure 2.14. The Loihi chip



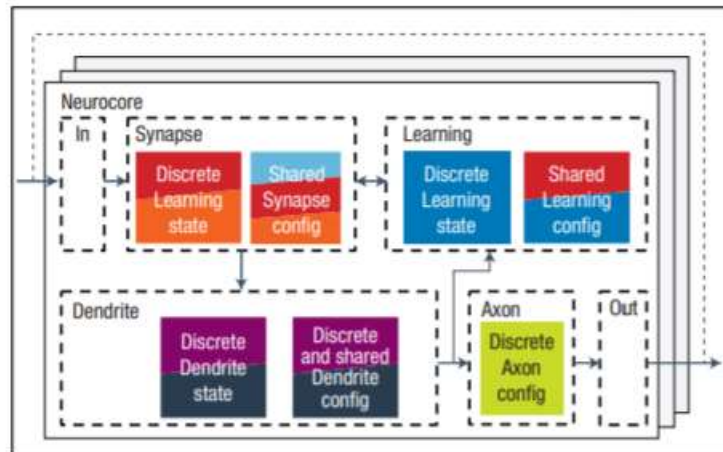
operates on SNN with a learning block that is programmable, and its logic is fully digital.



**Figure 2.14: Intel Loihi mesh operation enabling it to extend to other chips in four planar directions.**

The computational route of the Loihi is summed up in Figure 2.15 [70] where each dashed line outlines independent asynchronous blocks and the solid boxes are the states and configuration registers. Each neurocore is arranged in an asynchronous network-on-chip (NoC).

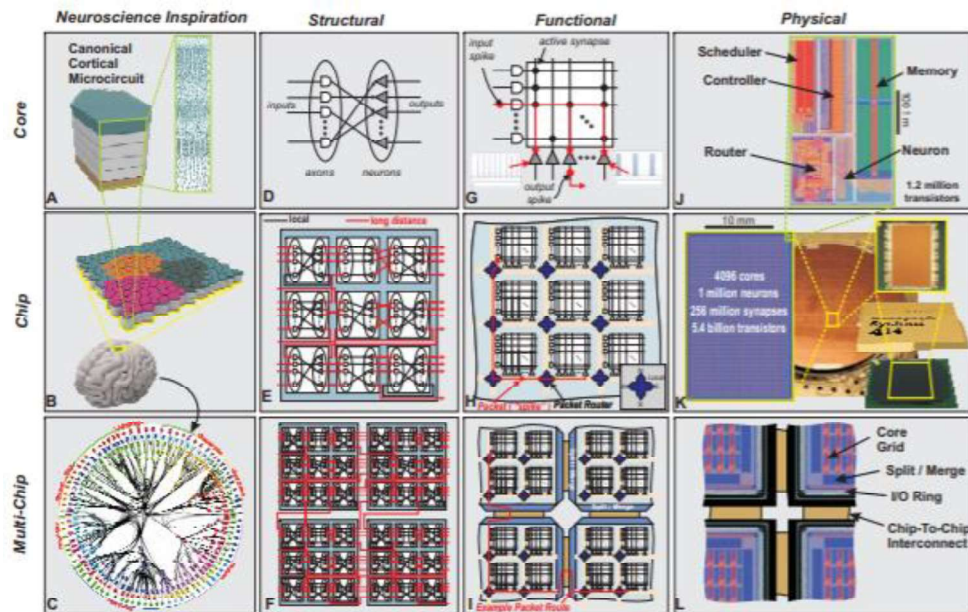




**Figure 2.15: Major architectural entities of the Loihi's computational route.**

### 2.5.2 IBM TrueNorth

The TrueNorth processor is an application-specific integrated circuit (ASIC) neuromorphic chip, released by IBM [71]. It contains 5.4 billion transistors with a million programmable spiking neurons, and 256 million synapses. The chip is energy-efficient due to its methodology of data and calculation division across the entire chip, so that the information does not have to travel long distances over the chip. In applications where power supply is scarce, the TrueNorth chip thrives as it processes and transfers data asynchronously, only as required. This feature is similar to how the human brain functions.

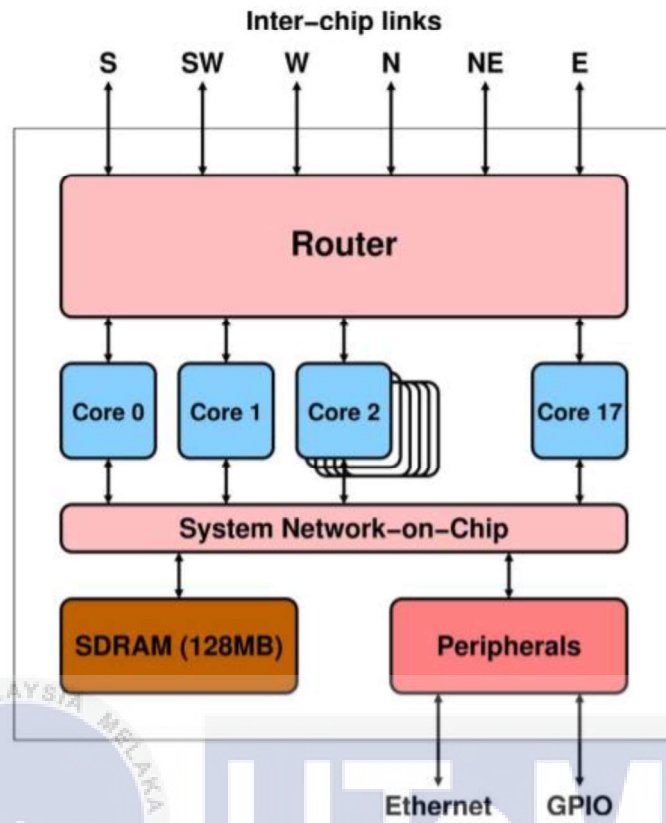


**Figure 2.16: Architecture of the TrueNorth neuromorphic chip.**

The architecture of the TrueNorth chip divides its panels into core, chip and multi-chip rows as well as neuroscience inspiration, physical, functional and structural columns as shown in Figure 2.16 [72]. Due to its performance, the TrueNorth is suitable to be used for research in neuroscience modeling. It has the potential and ability to develop neuromorphic simulators, allowing researchers to deeper study and understand the human brain.

### 2.5.3 SpiNNaker

SpiNNaker stands for Spiking Neural Network Architecture, which is a piece of neuromorphic hardware meant for simulating large volumes of neurons [73]. It is based on the acknowledgement of the fact that the form of communication used by the human brain has heavy computational requirements. Thus, the SpiNNaker architecture introduces a form of infrastructure for communication, the fundamental principle of which is carrying large volumes of very small packets of singular spike events [74]. The packets consist of a 32-bit AER identifier and 8 management bits.



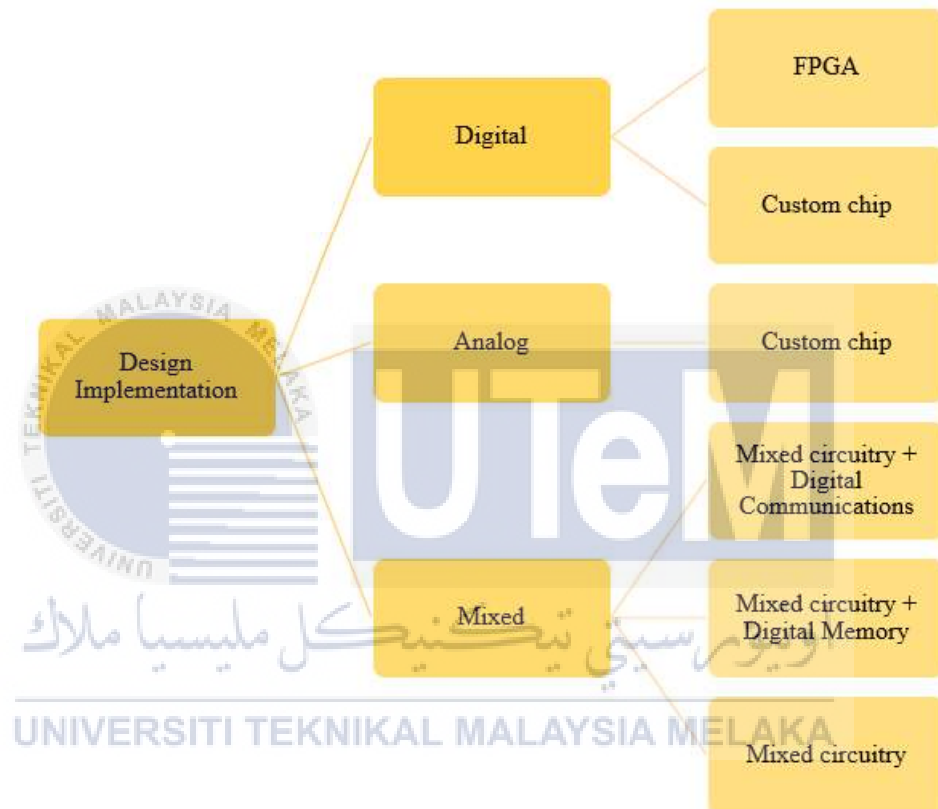
**Figure 2.17: Architecture of a SpiNNaker node.**

Overall, the SpiNNaker hardware is a homogenous 2D multiple data array consisting of processing nodes. The essential components of a singular node in the SpiNNaker architecture are illustrated in Figure 2.17. The implementation of each of these nodes is done on a  $19\text{mm}^2$  300 ball grid array package. The packet-routing mechanism in the chip is a key innovation based on conventional AER, where the packet router in every node checks the neural event packets and routes them depending on their source.

## 2.6 Design Implementation

Generally, neuromorphic algorithms can be implemented to investigate their functionality and analyze their performance [75]. On the integrated design environment (IDE), the SNN design can be synthesized and implemented. This allows

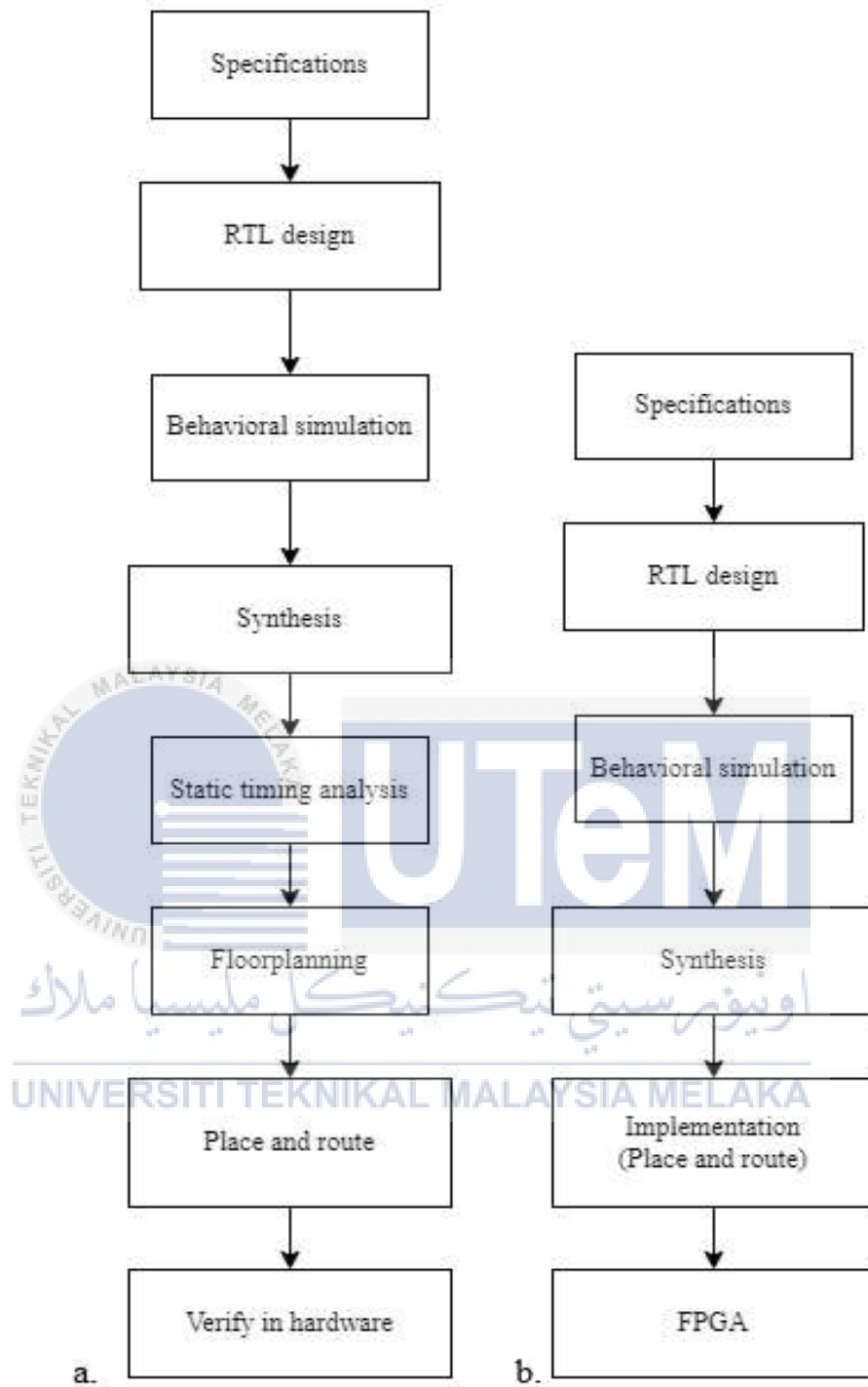
the designer to evaluate the design on a high level, as well as analyze the necessary parameters such as hardware utilization, layout density and power consumption before proceeding with hardware implementation on open-source FPGAs or custom fabricated chips. When it comes to implementation of a design, there are typically three categories as shown in Figure 2.18.



**Figure 2.18: Various categories of design implementation**

### 2.6.1 Target Devices

The design in this work is meant for implementation on the Zynq-7000 FPGA, so the design would be ready for hardware after it passes the analysis at the implementation stage, as the implementation is directly performed based on the constraints file for the board which is included in the design. However, the process would be different for implementation on an ASIC. The comparison for both processes are shown in Figures 2.19.



**Figure 2.19: Design process for a) an ASIC b) an FPGA**

### 2.6.2 FPGA Implementation in Previous Work

Table 2.2 shows previous research papers that have studied the implementation of ECG classification on an FPGA. The hardware resource utilization on the FPGA is

tabulated for each of these papers, consisting of the lookup tables, RAM, slice registers, digital signal processors (DSPs) and bonded input/output (IO). The power consumption for research that studied this parameter are also included. The results of the proposed design will be compared to these benchmarks, comparing the hardware resource usage of neuromorphic circuits compared to conventional circuits for ECG classification as well as power consumption.

**Table 2.2: Benchmarking of previous research implementing ECG classification on FPGA**

Publication	Classification method	Lookup tables	RAM	Slice registers	DSP	IO
Gu <i>et al.</i> [76] (2016)	Association-rule mining	10116	91	3433	20	-
Zhai <i>et al.</i> [77] (2017)	Principle component analysis	16133	17	11797	12	-
Madiraju <i>et al.</i> [78] (2018)	Time domain analysis	4324	-	1540	125	30

## 2.7 Summary

In summary, previous research has proven that raw ECG data needs to go through some form of preprocessing before it can be used. Various preprocessing methods have been suggested such as DWT, morphological filtering and spike encoding. Previous research has also shown the various methods in which neurons in SNNs can be modelled, such as the HH, IZH and LIF neuron models. Neuromorphic computing is proven to be an emerging form of technology eyed by multiple industry giants, and its implementation on FPGAs is compared.

## CHAPTER 3

### METHODOLOGY



#### 3.1 Overview

This chapter describes the methods applied in this work to achieve the objectives outlined in Chapter 1. This is done chronologically, from data collection to data preprocessing, emulating the SNN and implementing the digital design. Section 3.2 outlines the data collection method. Section 3.3 discusses the steps involved in the preprocessing of the ECG data. Section 3.4 highlights the SNN and the methods used to model, train and test it, while the techniques used for implementation of the digital design are detailed in Section 3.5.

#### 3.2 Data Collection

As shown in Table 2.1, there is no specific standard for data collection in ECG classification methods. Different datasets, encoding methods and SNN models are



used by different authors in their papers, so it is difficult to make a fair comparison between their results. However, a clear pattern in the training and testing datasets employed is that the MIT-BIH Arrhythmia Database is the most commonly used one as its data. In Ref. [80], datasets are compared based on four criterion which are sources, number of leads, duration and annotations. The two common sources from which ECG data is acquired are medical devices and healthcare devices, where data from medical devices contain more diagnostic information and is thus more informative than data collected via healthcare devices. The number of leads can range from one to 15 leads or more, differentiated by their ability to detect certain abnormalities. The duration of ECG data is classified into long-term data and short-term data, where long-term data collects data from patients over a prolonged period of time to monitor intermittent or underlying symptoms that may be missed in short-term data collection. Annotations refer to labels that describe a signal at specific points along the signal. For ECG data, this includes ECG measurement annotations, beat-level annotations and rhythm-level annotations. Annotation requires huge effort by medical experts.

The dataset from the MIT-BIH Arrhythmia Database [81] is obtained through Physionet [82]. The data was collected by the Massachusetts Institute of Technology in collaboration with the Beth Israel Hospital and consists of 48 30-minute-long excerpts of two-channel ECG recordings from 47 patients. The dataset includes normal ECG readings as well as abnormal ECG readings from 18 different anomalies.

Based on the annotations on the Physionet website, the condition of each ECG record can be observed in the 'Aux' column as shown in Figure 3.1. Table 3.1 shows the heart conditions represented by each of the annotations. In total, 36 records show



normal conditions while 12 show abnormal conditions. Thus, for each category, nine records are chosen for training and validation, while three are used for testing.

Time	Sample #	Type	Sub Chan	Num	AUX	Time	Sample #	Type	Sub Chan	Num	Aux
0:00.167	60	+	0	0	0	0:00.019	7	+	0	0	0
0:00.442	159	N	0	0	0	0:00.231	83	N	0	0	0
0:01.153	415	N	0	0	0	0:01.100	396	N	0	0	0
0:01.906	686	N	0	0	0	0:01.975	711	N	0	0	0

**Figure 3.1: Abnormal rhythm annotation (left) and normal rhythm annotation (right)**

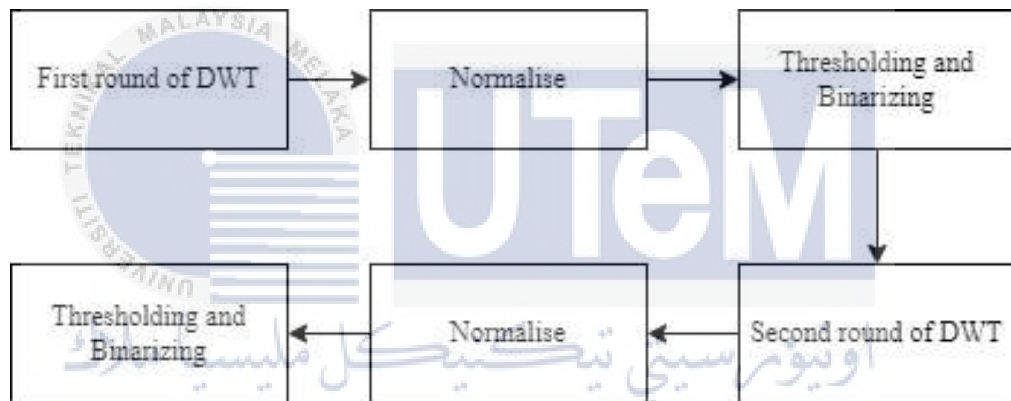
**Table 3.1: Meaning of beat annotations for the MIT-BIH Arrhythmia Database.**

Beat annotation	Meaning
(N	Normal sinus rhythm
(AB	Atrial bigeminy
(AFIB	Atrial fibrillation
(AFL	Atrial flutter
(B	Ventricular bigeminy
(BII	2° heart block
(IVR	Idioventricular rhythm
(NOD	Nodal (A-V junctional) rhythm
(P	Paced rhythm
(PREX	Pre-excitation (WPW)
(SBR	Sinus bradycardia
(SVTA	Supraventricular tachyarrhythmia
(T	Ventricular trigeminy
(VFL	Ventricular flutter
(VT	Ventricular tachycardia

### 3.3 Data Preprocessing

In designing neuromorphic circuits customized for ECG classification, the preprocessing step of ECG data is highly important. Traditionally, preprocessing ECG signals can be done by applying high-pass or low-pass filters to them [21] depending on the type of noise that is being targeted. For example, baseline wander is usually a very low frequency noise, in which a high-pass filter would be applied to the ECG

signal to filter it out. However, recent works have explored more conventional ways of denoising ECG signals, mainly due to the fact that the high- and low-pass filters have very specific cut-off frequencies, which often result in the input signal being distorted above or below a certain frequency. To make the ECG signal usable in an SNN, the preprocessing stage also requires a step to ensure the ECG signal is in a digitized form that is readable by the network. The ECG preprocessing steps are performed using the MATLAB software before moving the digitized data into the SNN in Vivado. The proposed preprocessing algorithm is illustrated in Figure 3.2 and consists of two techniques which are DWT and binarization.



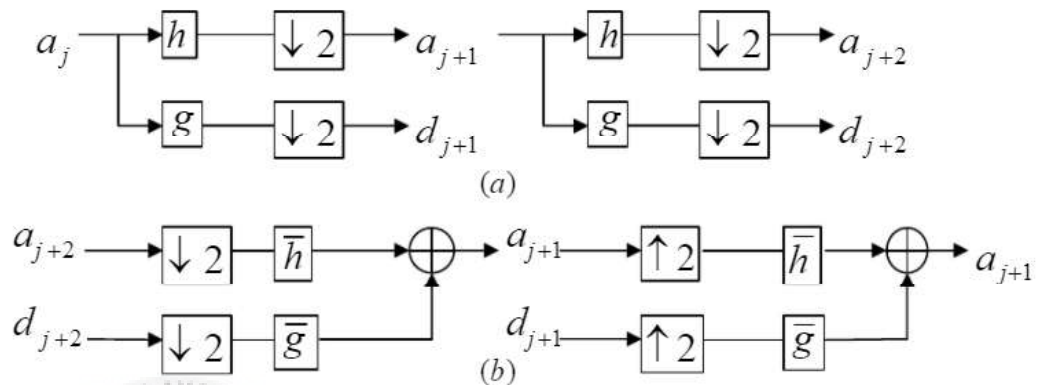
**Figure 3.2: Block diagram of the proposed ECG preprocessing process.**

### 3.3.1 Discrete Wavelet Transform

The input ECG signal is decomposed into several basic functions [32] that are referred to as the wavelets. These wavelets are obtained from a single mother wavelet, which stretches and shifts according to the ECG signal being processed. The filter bank tree for the decomposition and reconstruction stages are shown in Figure 3.4. In Figure 3.4, the approximation coefficients,  $a_j$ , are linked to the lower frequency components in the ECG signal, while detail coefficients,  $d_j$ , enable the shape of the signal to be preserved during the reconstruction stage in Figure 3.3.



**Figure 3.3: DWT process.**



**Figure 3.4: Filter bank trees for: (a) wavelet decomposition (b) wavelet reconstruction**

In DWT, there is a finite-length oscillating waveform, from which scaled and translated copies are created. The oscillating waveform is often referred to as the ‘mother wavelet’  $\psi_{m,n}(t)$ , while the decomposed frequency components are the child wavelets. The wavelet coefficients of the discrete set of child wavelets are computed. For the DWT, the mother wavelet is shifted and scaled dyadically as shown in Eq. 3.1, where  $m$  controls the wavelet scaling and  $n$  controls the wavelet translation.

$$\psi_{m,n}(t) = \frac{1}{\sqrt{2^m}} \psi\left(\frac{t-n2^m}{2^m}\right) \quad \text{Eq. 3.1}$$

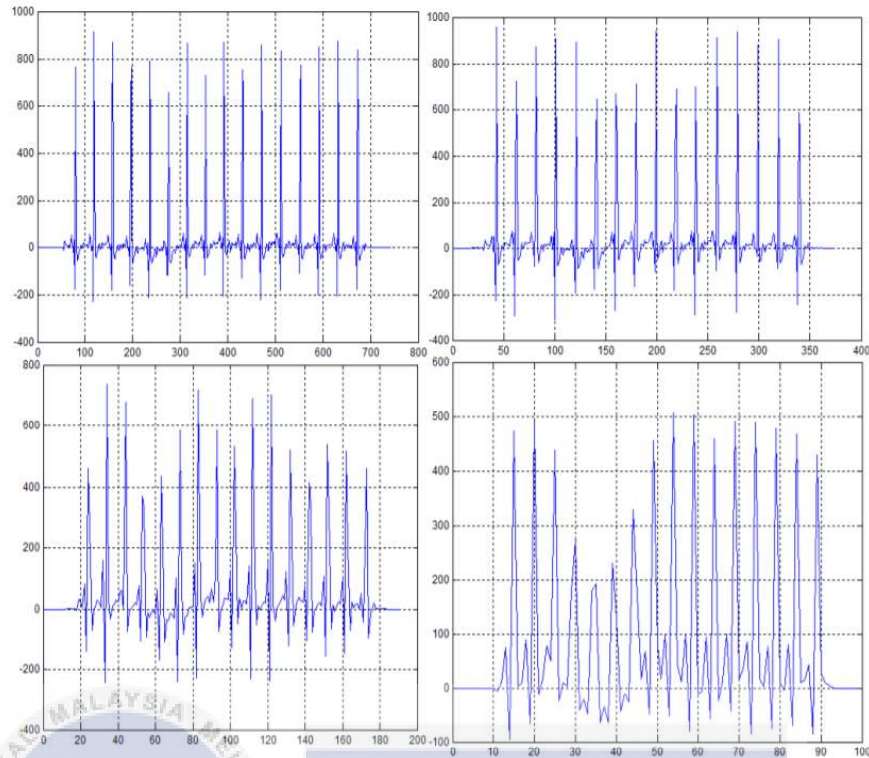
The thresholding step serves to remove some coefficients from the child wavelets. A filter is applied here, using Mallat’s algorithm. This convolutes the mother and child wavelets with a bandpass filter, removing low-amplitude noise below a certain frequency. The Donoho algorithm [83] performs thresholding on most of the detail coefficients. The value of the threshold,  $\lambda$ , is decided based on the level of

decomposition performed. Two methods of thresholding are soft and hard decisions as shown in Eq. 3.2 [84], which differentiate whether a specific noise component is pure noise or coefficient plus noise. In the hard decision, coefficients are only preserved if they are higher than the threshold value, while the soft decision considers all coefficients with noise. Donoho also proposed a formula to approximate the ideal value of  $\lambda$  (Eq. 3.3), where  $M$  is the number of coefficients.

$$\text{Soft: } d_j = \begin{cases} d_j - \lambda & \text{if } d_j > \lambda \\ d_j + \lambda & \text{if } d_j < -\lambda \\ 0 & \text{else} \end{cases} \quad \text{Hard: } d_j = \begin{cases} d_j & \text{if } |d_j| > \lambda \\ 0 & \text{else} \end{cases} \quad \text{Eq. 3.2}$$

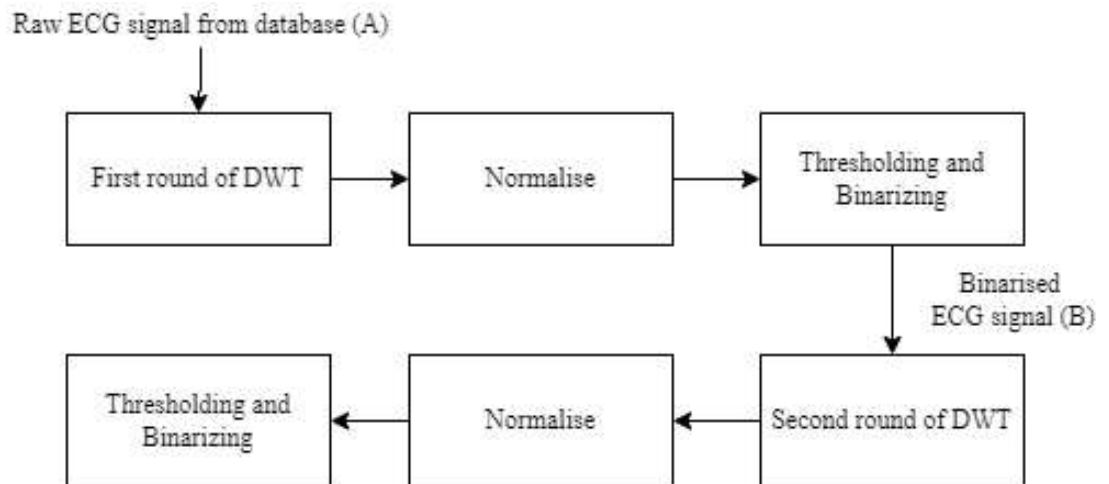
$$\lambda = \sqrt{2 \log M} \quad \text{Eq. 3.3}$$

The proposed DWT serves to compress the data by down sampling it using Daubechies wavelet, adapting the methodology used in Ref. [85]. The DWT process compresses the data so that it is in a lower frequency than the original signal, as shown in Figure 3.5. In this process, the peak of the ECG signal which is the QRS complex is still preserved.



**Figure 3.5: Down sampling of ECG signals using Daubechies wavelet. [85]**

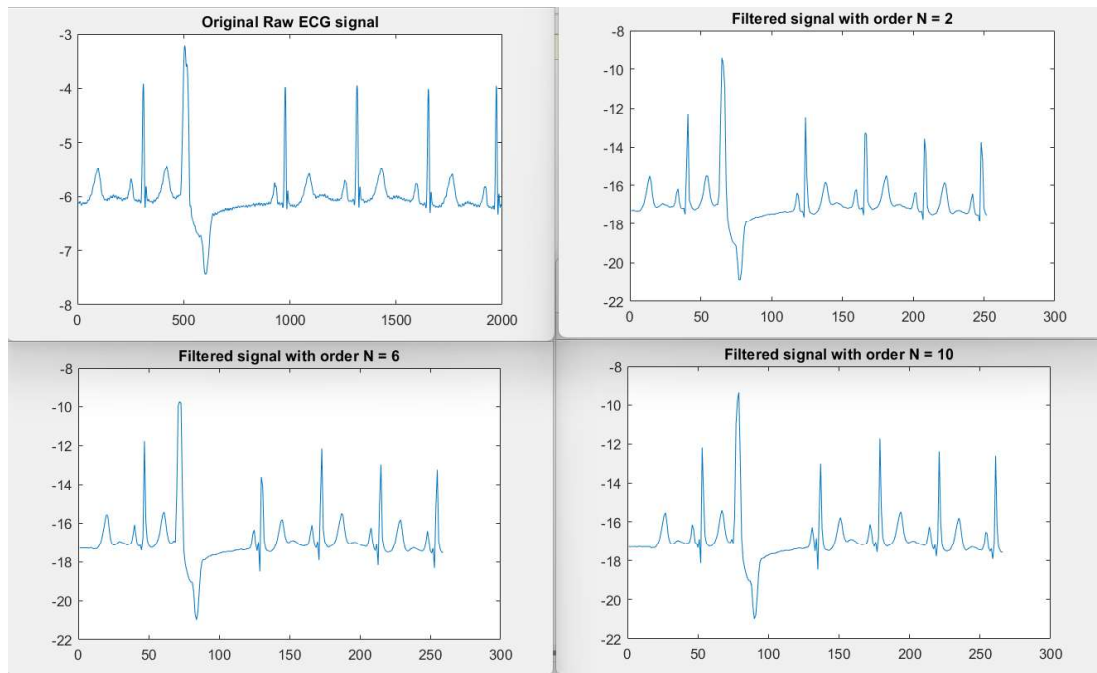
Applying DWT in MATLAB returns detail coefficients as well as approximation coefficients. Detail coefficients of a signal after DWT are basically the noise removed from the signal, plotted onto finer scales while the approximation coefficient is the approximated signal after being filtered, shown in a coarser scale [86] which inherently compressing it to a lower sampling rate.



**Figure 3.6: Inputs of the DWT process.**

In the proposed method, DWT is applied to the signal twice. A clearer illustration of this is shown in Figure 3.6, where signal A is the input for the first DWT and signal b is the input for the second. While the purpose of transforming signal A is mainly for denoising and compressing the signal, the discrete wavelet transformation on signal B is mainly for compressing the signal only, to simplify the system so that the size of the input to the SNN is smaller.

In MATLAB, the Daubechies wavelet is described as 'dbN', where 'db' shows the Daubechies wavelet family and N refers to the order of the wavelets used. The number of orders corresponds to the number of vanishing moments. Generally, the approach in selecting the vanishing moments is that when the vanishing moments are lower, more signal detail is lost, leading to more distortion of the original signal [87]. This theory is demonstrated in Figure 3.7. In this work, db6 is used for the first round of DWT and db2 for the second round. This is to minimize distortion of the signal while at the same time ensuring the size of the ECG signal fitted into the SNN is large enough for detail.

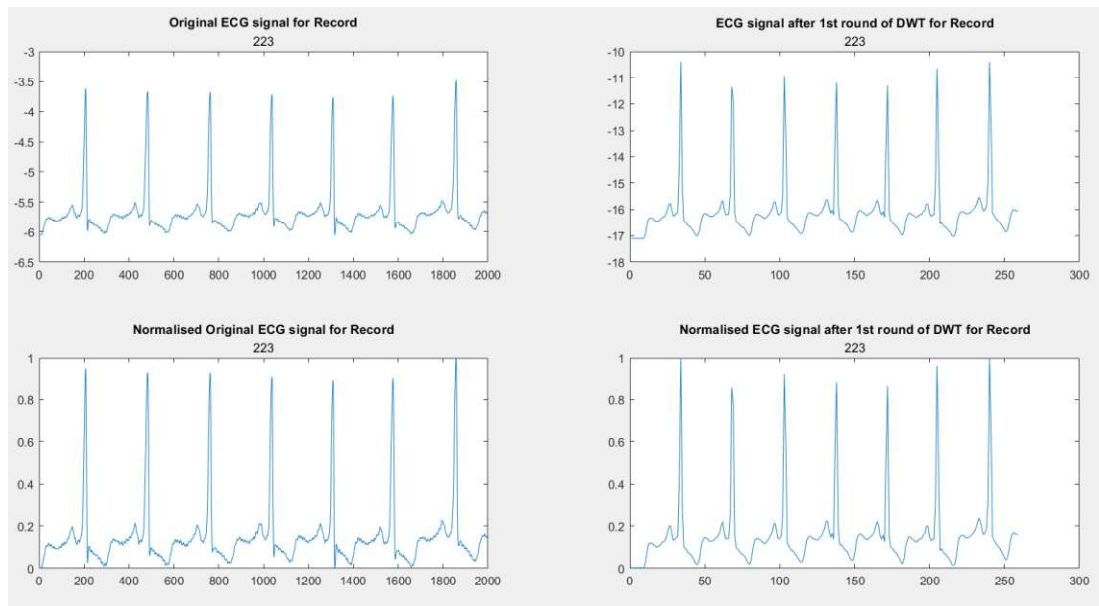


**Figure 3.7: Comparison of the results of DWT with different vanishing moments.**

### 3.3.2 Normalization

The normalization process rescales data so that its maximum amplitude is 1 while its minimum is 0. This is done so that the data and its changes can be easily observed throughout the preprocessing steps. This is illustrated in Figure 3.8, where the original and filtered ECG signals in the second row are easier to compare visually after normalization.

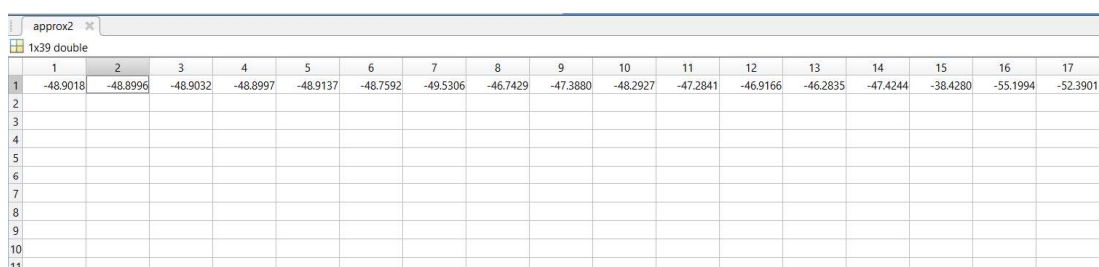




**Figure 3.8: ECG signals before normalization (top row) and ECG signals after normalization (bottom row)**

### 3.3.3 Binarization

In a digital system, data is processed in a digital form. As observed from the output of the DWT, the transformed ECG signal is still in analog form as shown in Figure 3.9. Thus, the data has to go through binarization to convert it to 0's and 1's that can be processed by the SNN.

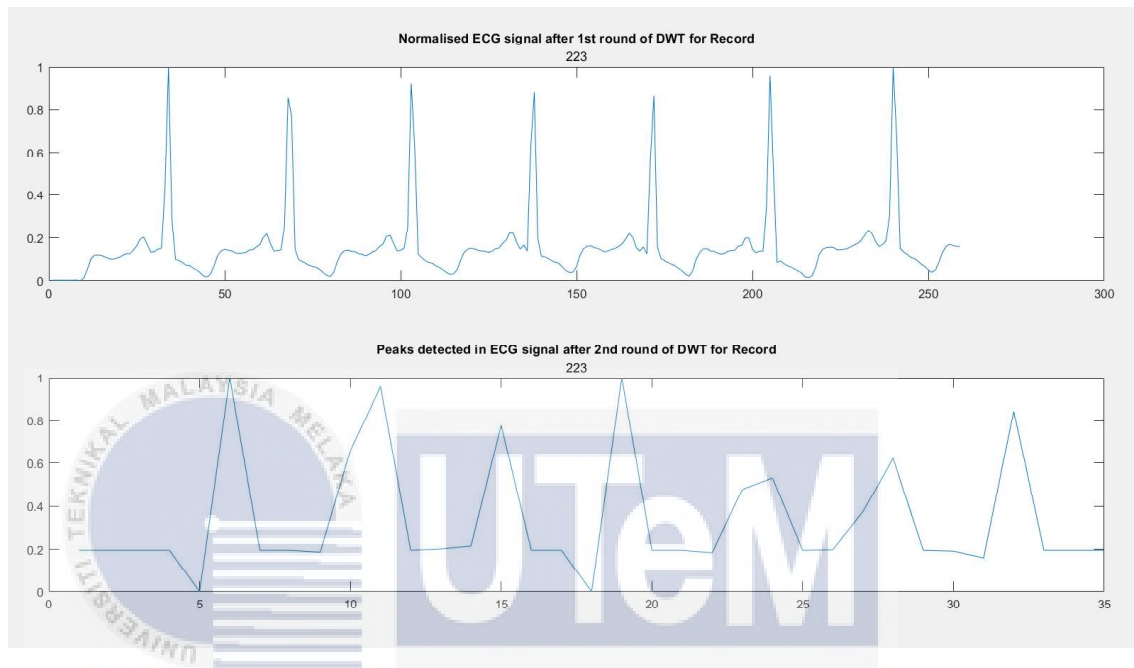


**Figure 3.9: A portion of the analog ECG data shown in the MATLAB workspace after going through DWT.**

In image binarization, values are assigned to each pixel based on their luminance. Thus, any pixel whose greyscale value exceeds the threshold is binarized into a 1, while any pixel below the threshold value is binarized to a 0. For ECG data, the same



technique is applied in MATLAB. The threshold values are selected based on the amplitude of the normalized R peaks, which are generally between 0.6 to 0.7 (60% to 70%) of the R peak's amplitude after the first round of DWT and 0.4 to 0.5 (40% to 50%) after the second round of DWT. This can be seen in Figure 3.10.



**Figure 3.10: Different threshold values are needed after the first round and second round of DWT respectively.**

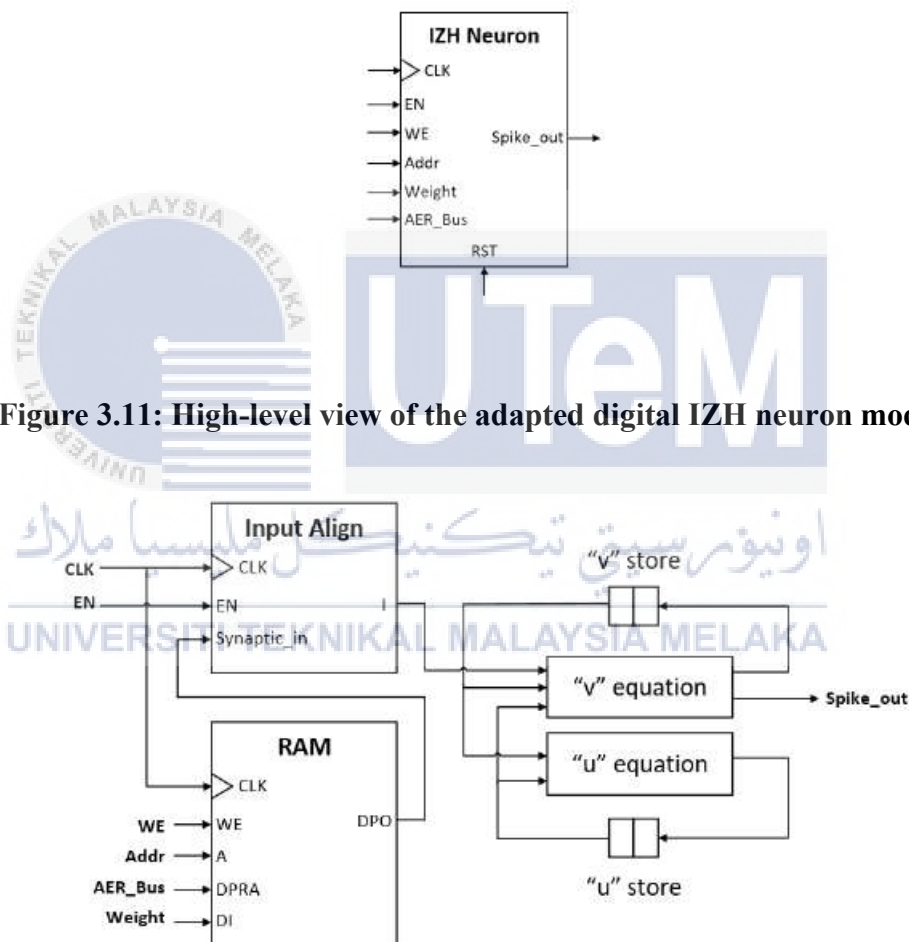
### 3.4 Spiking Neural Network

In this work, an adaptation of the SNN model designed in Ref. [47] is used, where the network is modified to classify ECG readings instead of Modified National Institute of Standards and Technology (MNIST) handwritten digits as in the original design. The neuromorphic circuit is designed at register transfer level (RTL) before proceeding to implementation.

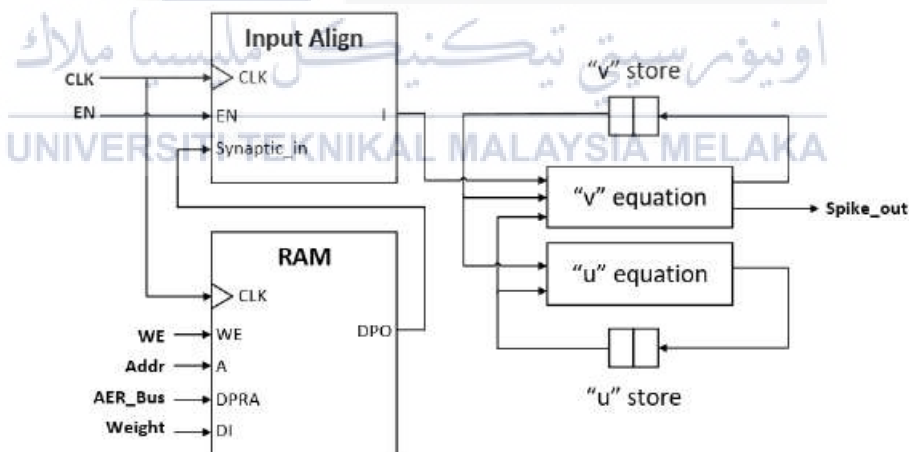
#### 3.4.1 Neuron Model

The neuron model emulated in this work is the IZH neuron. Based on the equations in Section 2.4.2.2, a neuron module is adapted from Ref. [47] that is a digital model

of the IZH neuron to simplify computations. This architecture is shown in Figure 3.11, where the CLK signal provides pace and coordination, the EN signal for neuron activation, the WE signal for Write Enable, Addr signal for the neuron addresses, AER\_Bus signal which is the input carrying information about which neuron spiked previously, and an output signal labelled Spike\_out to show whether or not the neuron spiked. Figure 3.12 shows the internal block diagram of the module in the RTL design.



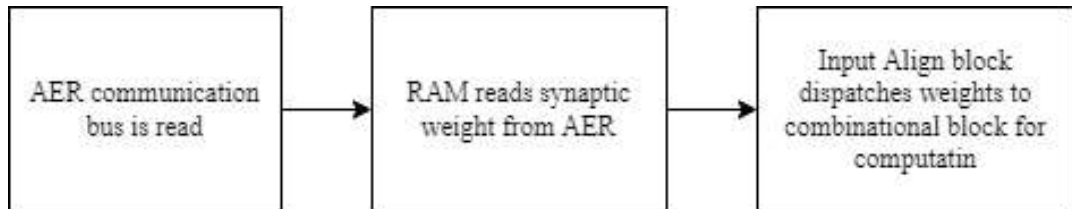
**Figure 3.11: High-level view of the adapted digital IZH neuron module.**



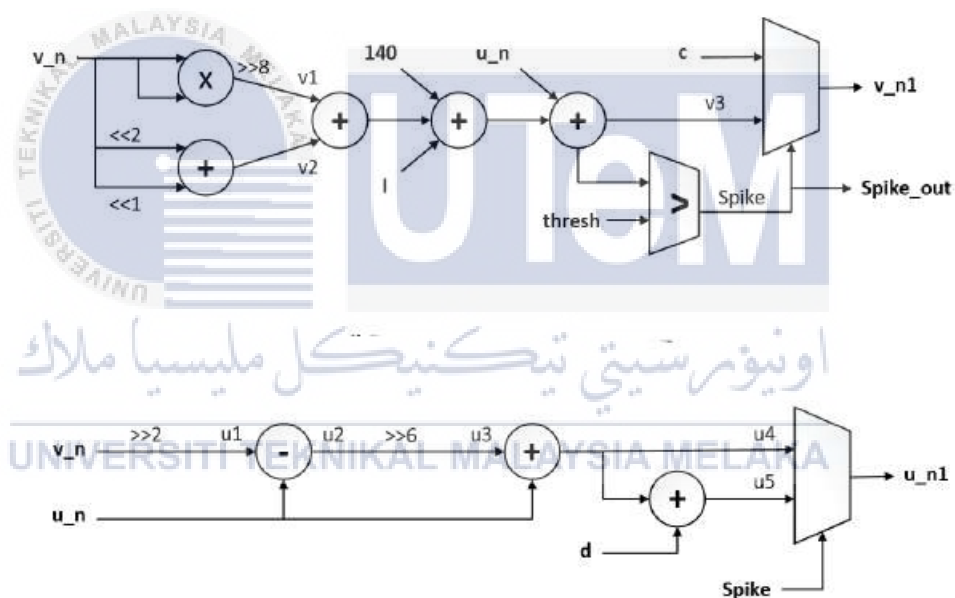
**Figure 3.12: Internal block diagram of the module.**

The overall operation of the neuron module is explained in Figure 3.13. If more than one neuron spiked simultaneously, the AER bus halts all operations in the neuron because it can only transmit one spike at any given time. The Input Align block limits

the synaptic weight's negative value to -140mV to avoid spikes being generated when it is not supposed to. The combinational blocks perform the equations shown in Section 2.4.2.2. This is illustrated in Figure 3.14.



**Figure 3.13: Overall operation of adapted IZH neuron module.**



**Figure 3.14: Digital implementation of the “v” equation (above) and the “u” equation (below) in the equations shown in Section 2.4.2.2.**

### 3.4.2 Training

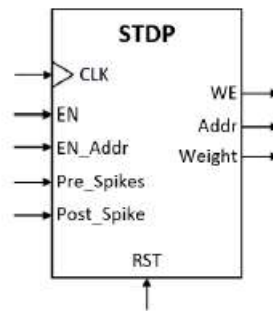
Training methods for SNNs generally are based around the STDP rule, where the timing of a neuron firing is crucial in the learning process. In Ref. [47], three conditions are satisfied in regards to the neurons before any training starts. First, the input layer of the SNN is simulated for some time with a particular frequency, intended

to return all the neurons to their initial state. Next, all the synapses that exist in the network connecting the input layer to the output layer is initialized to 0 to ensure the output neurons stay in the resting state, even if the input layer is simulated. Lastly, the STDP training module of the output neurons are connected to their corresponding training neuron. Only when these conditions are fulfilled can the training begin.

Training the SNN involves manually firing the neurons in such timing so that the network correctly associates the input stimulus. This can be done in two ways. First, the training neuron that is linked to the correct output neuron is fired after the input data has been introduced into the network. Second, all training neurons except the correct one are fired right before introducing the data into the network. According to the STDP rule, synapses are strengthened between neurons if the pre-synaptic neuron contributes to the firing of a post-synaptic neuron. This is to say that the post-synaptic neuron should fire after the arrival of the pre-synaptic neuron to prove the contribution of the pre-synaptic neuron.

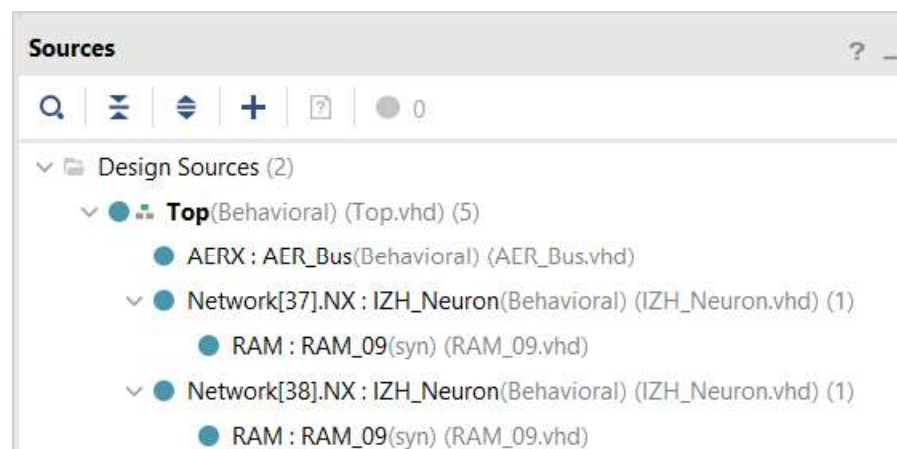
#### **3.4.2.1 Spike Timing-Dependent Plasticity in the SNN**

In Ref. [88], the STDP learning mechanism is modelled with digital logic. Adapting this approach in Ref. [47], the digital block in Figure 3.15 is modelled to represent the STDP module. The block has six inputs and three outputs.

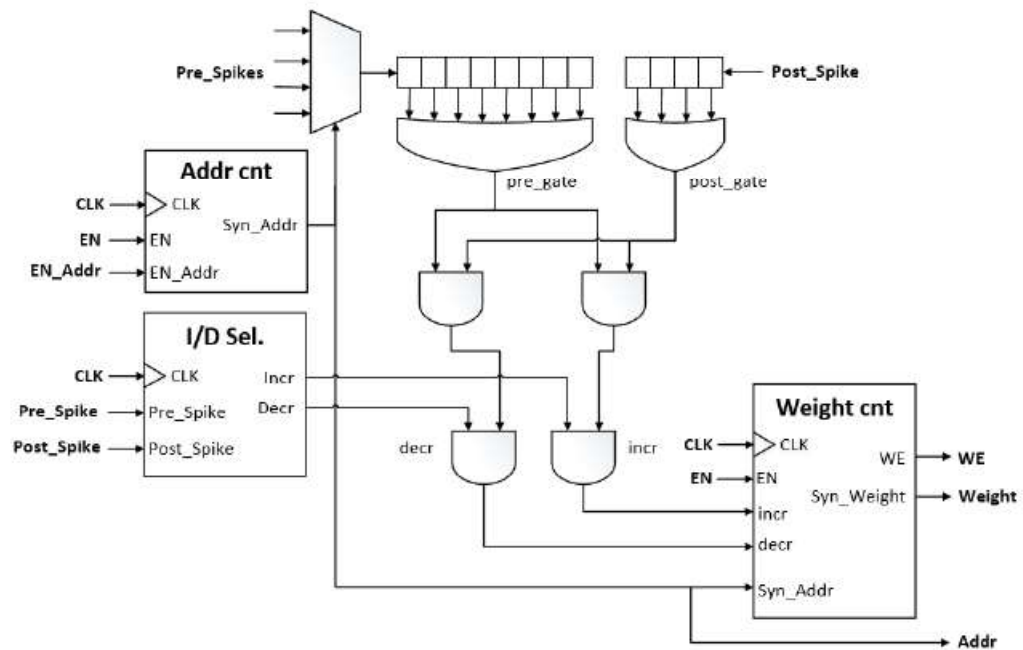


**Figure 3.15: STDP module modelled in the proposed SNN.**

The figure shows that the six inputs are CLK, EN, EN\_Addr, Pre\_Spikes, Post\_Spike and RST while the outputs are WE, Addr and Weight. The three outputs write to the RAM of the neurons, where each neuron has its own RAM as shown in the Vivado hierarchy in Figure 3.16. The CLK input is for the clock signal, which coordinates and paces the module's actions. EN is responsible for receiving the enable signal which activates the learning process, and EN\_Addr changes the neuron connections accordingly while applying the STDP rule. Pre\_Spikes reads the spikes of the previous neuron, and Post\_Spike of the next neuron. The inner architecture of these inputs and outputs is shown in Figure 3.17.



**Figure 3.16: Hierarchy of the design sources in Vivado, where each neuron has its own RAM.**



**Figure 3.17: Internal block diagram of the STDP module.**

The internal architecture of the STDP module consists of three main blocks which are the Addr cnt, I/D Sel and Weight cnt. If there is a spike from the previous neurons connected to the module read on the Pre\_Spikes input, the I/D Sel block decides whether the weight counter needs to increase or decrease the synaptic weight of connections and indicates this by activating the respective output signal accordingly. The spike activates the pre\_gate signal, and the post\_gate signal is activated if the training neuron module spikes. With these gate signals being activated, plus an Increase signal indicated by the I/D Sel block, an increment pulse is generated for the synaptic weight counter. The value by which the weight of the synaptic connection increases depending on how long this pulse lasts. Then, the WE signal is activated to update the neuron RAM with the weights. This module overall models the STDP learning rule with the coordination of the impulses.

### 3.4.3 Testing

A testbench script is used for testing the functionality of the network. The testbench manipulates input signals to the system so that the system's response to it can be observed and verified. For this design, the test data is input to the system via the testbench file. Since there are two neurons in the hidden and output layer respectively for the two classifications and 35 neurons in the input layer for the 35-bit ECG data, there are a total of 39 neurons in the network, from neuron 0 to neuron 38. One ECG record is fed into the SNN at a time, and the corresponding neuron trained to recognize the particular category is observed. In this case, neuron 37 is trained to fire for normal data while neuron 38 should fire for abnormal data.

## 3.5 Design Synthesis and Implementation

After the behavioral simulation has confirmed that the system is functioning as desired, RTL analysis can be performed in Vivado where the design is converted into an RTL circuit. RTL is an abstraction layer [89] where data transfer between registers is modelled. The coding that is done is considered RTL design, while RTL analysis is done on the schematic that shows the connection of registers throughout which data moves. Subsequently, synthesis is a process that shows the RTL design as a network of logic gates. Lastly, implementation of the design can be done, where the chip layout is adjusted.

### 3.5.1 ZedBoard Zynq-7000 ARM/FPGA SoC Development Board

The Zedboard is the development board for the Xilinx Zynq®-7000 system-on-chip (SoC), which allows complete simulation and analysis of the SoC's operation. The device is commonly used for implementation of hardware algorithms, including neuromorphic system designs [75]. For implementation, the constraints file of the

Zedboard is needed in the design, which also suffices for implementation on the IDE. The Zedboard architecture consists of two fundamental sections which are the processor subsystem (PS) and the programmable logic (PL). The PL section is optimum for the implementation of high-speed logic, arithmetic and data flow subsystems, while the PS supports software routines and operating systems [90]. This is illustrated in Figure 3.18.

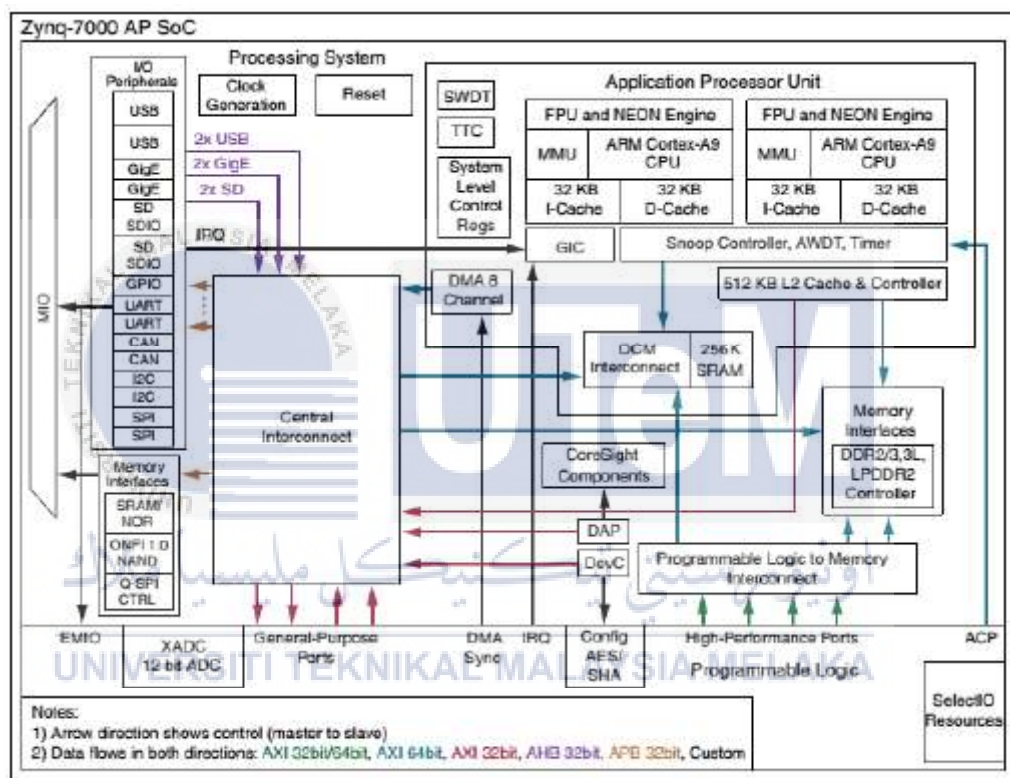


Figure 3.18: Block diagram of the Zedboard.

### 3.6 Summary

The methods proposed in this work involve DWT and binarization to preprocess the raw ECG data into a binary input for the SNN. In the SNN itself, STDP is employed as the training mechanism for ECG classification. Implementation of the RTL design is then performed for the Zedboard with a constraints file.



## CHAPTER 4

### RESULTS AND DISCUSSION



#### 4.1 Overview

In this chapter, the findings of this work are presented and discussed. This is arranged according to the corresponding objective, first being to investigate the building blocks of a neuromorphic design, which is detailed in Section 4.2. The findings fulfilling the second objective are presented in Section 4.3, from analyzing spike-based plasticity. Section 4.4 discusses the results from implementation on a neuromorphic circuit, while Section 4.5 compares the findings with findings by other authors.

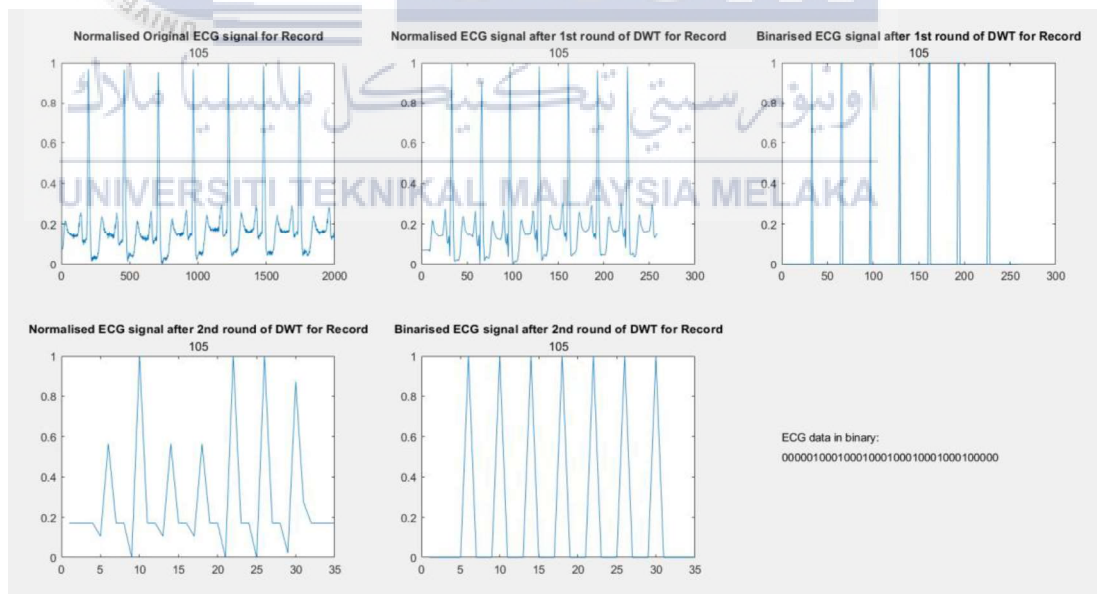
#### 4.2 Building Blocks of a Neuromorphic Design

This work contributes the processes and techniques involved in designing a neuromorphic circuit customized for ECG classification. In doing this, several SNN

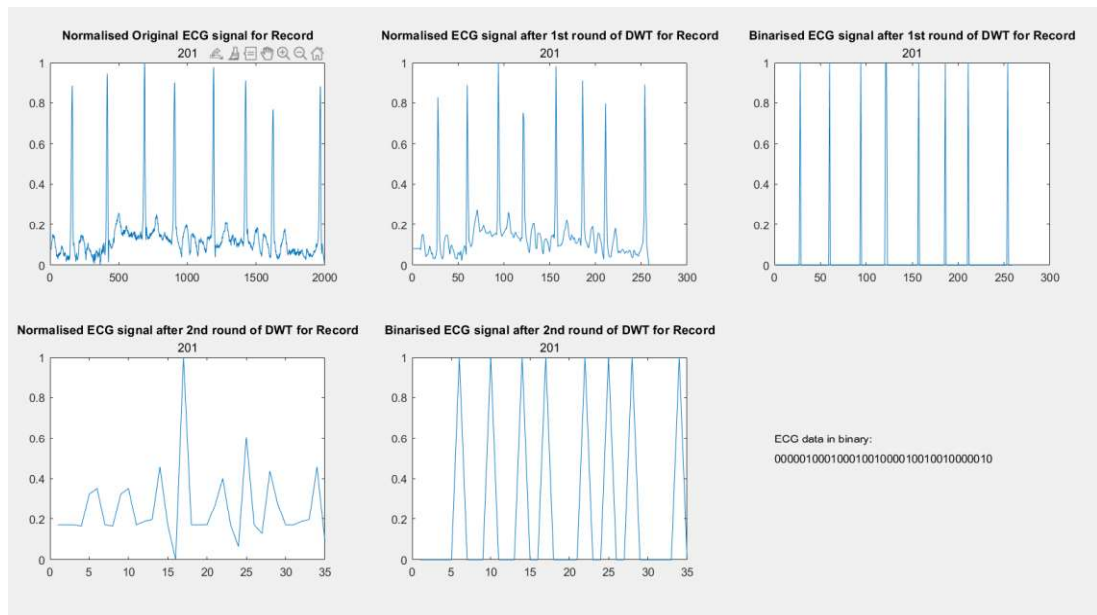
building blocks that make up a neuromorphic design were studied and implemented. This refers to the steps involved in enabling neuromorphic computing, such as preprocessing the input data, training the network and testing the network for classification functionality. These building blocks come together to form a complete neuromorphic design that employs SNN for classification.

#### 4.2.1 Preprocessing Block

For digital implementation of an SNN such as in this work, the data is processed by the network digitally [47], [91]. When the data being used is not in digital or binary form, it requires preprocessing [92],[93],[94] before being fed into the network as input. The DWT is applied to the ECG data obtained from the MIT-BIH arrhythmia database (MITDB). Figures 4.1 and 4.2 show the results of each individual step of preprocessing performed on normal and abnormal ECG signals respectively.

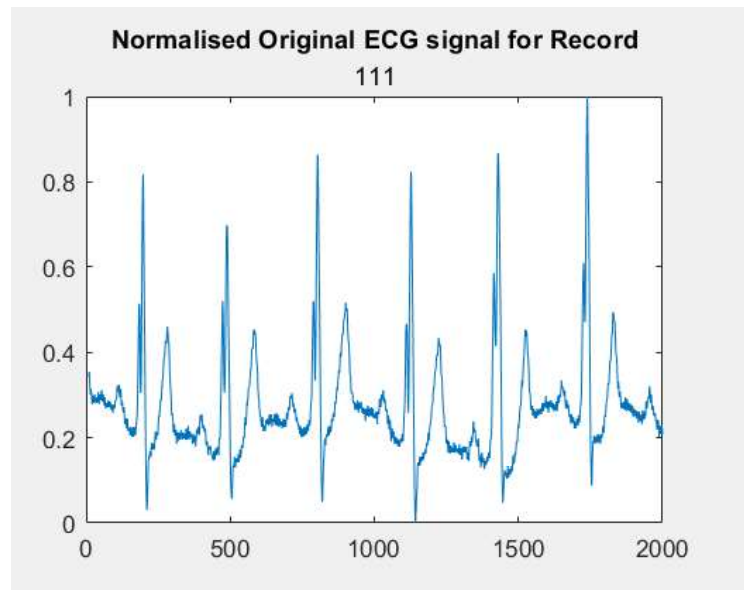


**Figure 4.1: Results of each preprocessing step performed in MATLAB for record 105 (normal).**

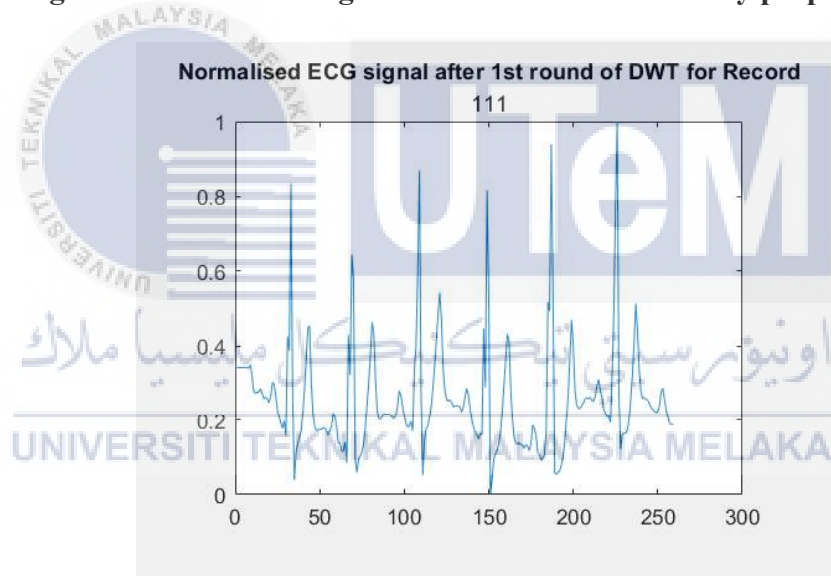


**Figure 4.2: Results of each preprocessing step performed in MATLAB for record 201 (abnormal).**

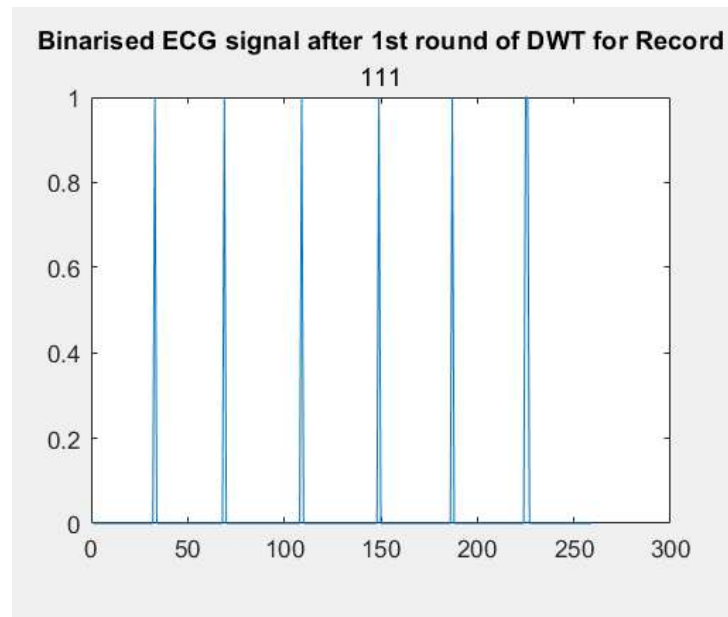
The preprocessing step applied to the ECG data is mainly the DWT. The first round of DWT uses Daubechies db6 wavelet to filter the original ECG signal by removing the high- and low-frequency elements in the signal [95], resulting in the differences in the ECG signal shown in Figure 4.3 and Figure 4.4. Due to the R peaks in the ECG signal still being prominent after the first round of DWT as shown in the figure, the peaks are preserved before the second round of DWT by binarizing the signal at this point. By doing this, it is observed in Figure 4.5 that the R peaks are binarized as 1's and the rest of the signal as 0's as shown in the third plot in each of the figures.



**Figure 4.3: The ECG signal for record 111 before any preprocessing.**

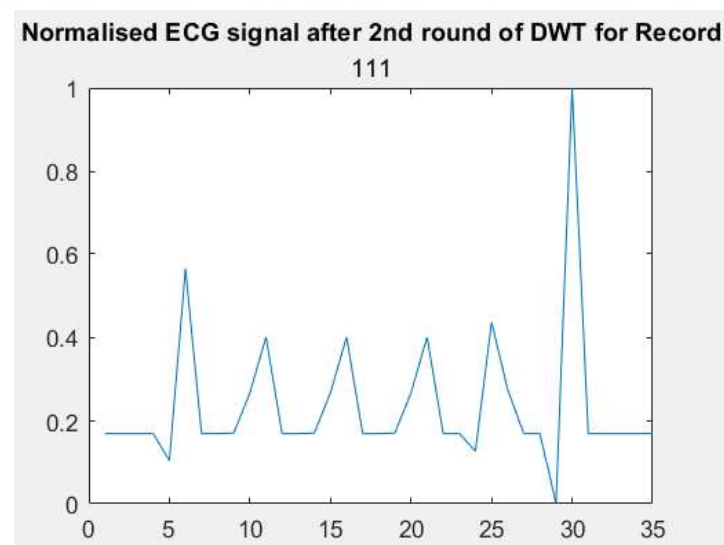


**Figure 4.4: The ECG signal for record 111 after being filtered by the db6 wavelet.**



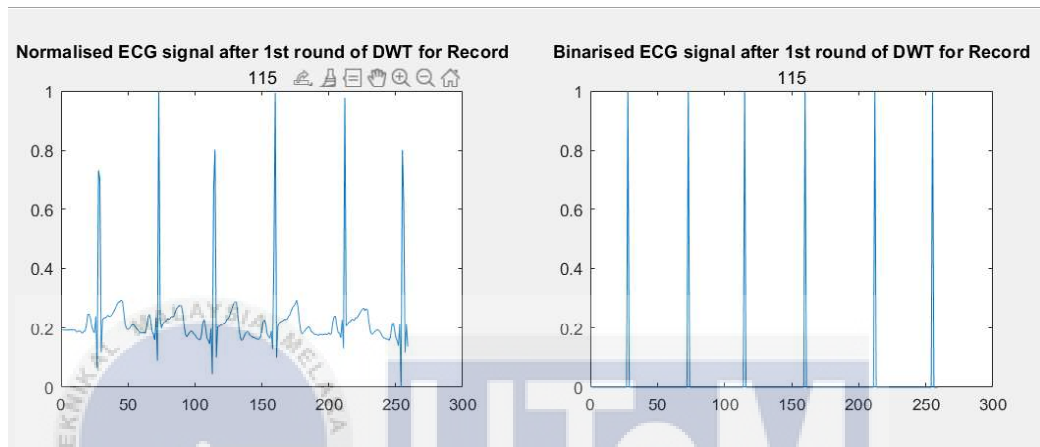
**Figure 4.5: The binarized ECG signal for record 111 after being filtered by the db6 wavelet.**

The second round of DWT is performed using Daubechies' db2 wavelet, which compresses the signal by down sampling it [96], resulting in a 35-bit representation of the signal as shown in Figure 4.6. It can be seen at this point that the R peaks observed in the earlier stages of preprocessing are still preserved and can be easily seen. This is binarized to get a 35-bit binary representation of the ECG signal.

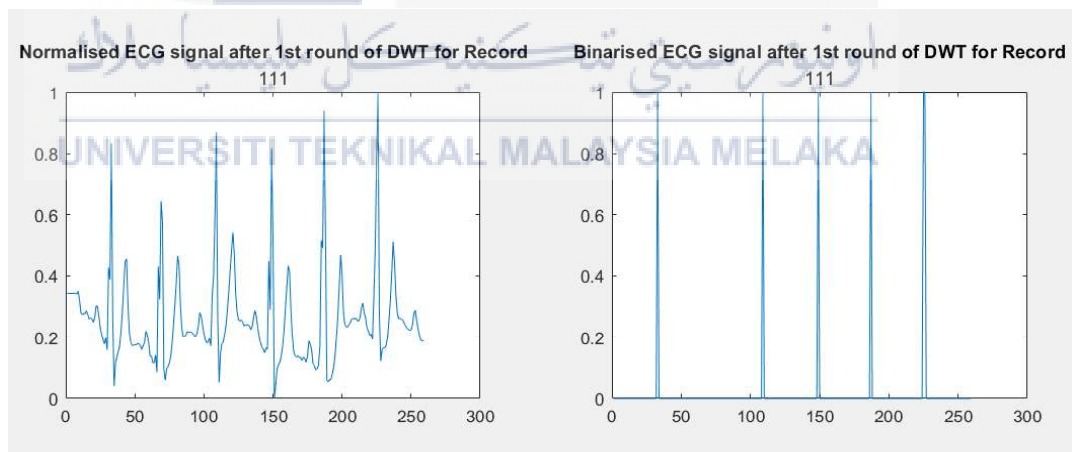


**Figure 4.6: The ECG signal for record 111 after being filtered by the db6 and db2 wavelets.**

One particular downside of the preprocessing method used in this work is that threshold value used at the binarization stages. This is illustrated in Figures 4.7 and 4.8, where a threshold value of 0.7 is used for both. This indicates that the points with amplitude higher than 0.7 in the normalized ECG signal after the first round of DWT will be binarized to 1, while the rest are binarized as 0.



**Figure 4.7: Peak detection in Record 115 with a threshold of 0.7.**



**Figure 4.8: Peak detection in Record 111 with a threshold of 0.7.**

For record 111, the shape of the original ECG signal is still preserved, but the R peaks have different amplitudes. When 0.7 is used as the threshold value, most of the R peaks are binarized as 1's, but the second R peak is missed out due to its slightly lower amplitude that is below 0.7.

#### 4.2.2 Training Block

With the 35-bit binary ECG data, the SNN in Ref.[47] can be adapted to classify the data. For training, the data is arranged in an array in the Top script and fed into the network via the Digit signal. The pseudocode for feeding the data into the network is shown in Figure 4.9, where the switching of data is controlled by the signal Image\_Signal which is two bits, Image\_Signal(0) and Image\_Signal(1). To achieve this, the training data is arranged in an array as shown in Table 4.1.

**Table 4.1: Arrangement of training data in arrays.**

Normal data array (normalSequence)		Abnormal data array (abnormalSequence)	
Position in array	ECG record no.	Position in array	ECG record no.
0	100	0	102
1	105	1	107
2	106	2	200
3	112	3	201
4	114	4	203
5	121	5	207
6	205	6	210
7	222	7	221
8	223	8	232

Two counter variables are introduced, n\_counter for normal data and a\_counter for abnormal data. The mechanism used is to increment each counter by one whenever the Image\_Signal control signal changes. In this case, the normal data is inserted when Image\_Signal(0) is 1, otherwise abnormal data is inserted if Image\_Signal(1) is 1. The pseudocode for increment of array pointer is shown in Figure 4.10, and the result showing insertion of multiple ECG records for training is shown in Figure 4.11.

```

if (Image_Signal = 1) then
  Digit = normalSequence (n_counter);
else if (Image_Signal = 2) then
  Digit <= abnormalSequence(a_counter);
end if;

```

**Figure 4.9: Pseudocode for insertion of multiple data.**

```

if rising_edge(Image_Signal(0)) then
  if n_counter reached 9 then
    set n_counter back to 0;
  else increment n_counter by 1;
  end if;

else if rising_edge(Image_Signal(1)) then
  if a_counter reached 9 then
    set a_counter back to 0;
  else increment a_counter by 1;
  end if;

```

**Figure 4.10: Pseudocode for increment of array pointer.**

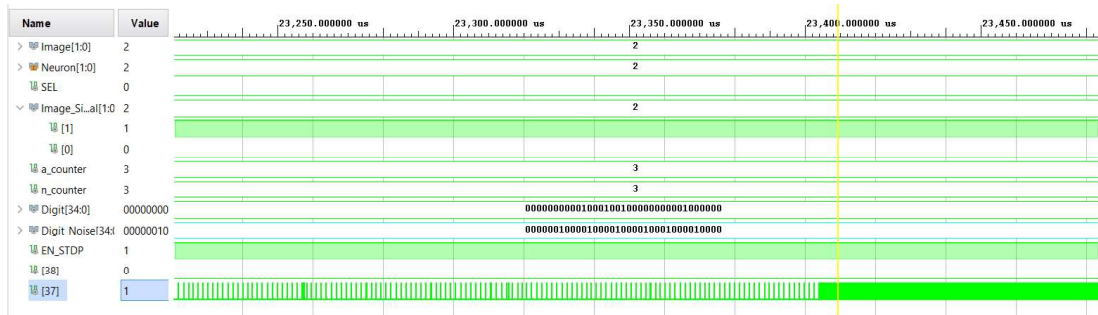


**Figure 4.11: Insertion of training data automated by the counters moving through the normal and abnormal data arrays.**

### 4.2.3 Testing Block

For testing the network, three ECG records are used for each category. The test data is inserted into the Digit\_Noise signal, which is modified through the testbench of the Top script. The output of the SNN is observed at neurons 37 and 38, which are the output neurons for normal and abnormal categories respectively. When normal data is inserted into the network via Digit\_Noise, neuron 37 spikes as shown in Figure 4.12. On the other hand, if abnormal data is inserted into the network via Digit\_Noise, neuron 38 spikes as shown in Figure 4.13.





**Figure 4.12: Neuron 37 spikes to classify record number 103 correctly as normal.**



**Figure 4.13: Neuron 38 spikes to classify record number 217 correctly as abnormal.**

The accuracy of this classification is generally correct, although the exact figure for accuracy percentage is unable to be determined based on the behavioral simulation purely, because the results at this stage are only observed at the neuron's waveform.

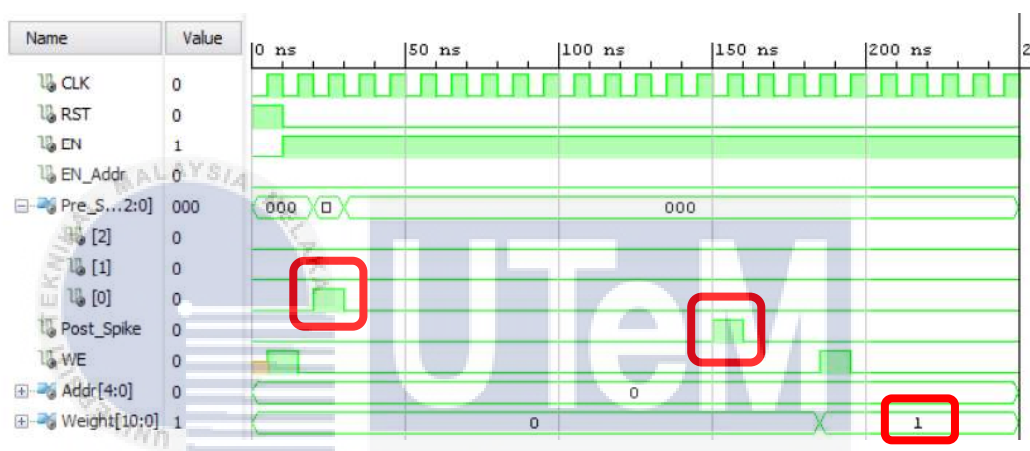
In this situation, there is a rare instance of both neurons spiking at the same time as shown in Figure 4.14, which cannot be quantified. With the observation of the neurons spiking mostly correctly at this stage, the design process is continued.



**Figure 4.14: The incorrect output neuron randomly spikes while the correct output neuron is spiking.**

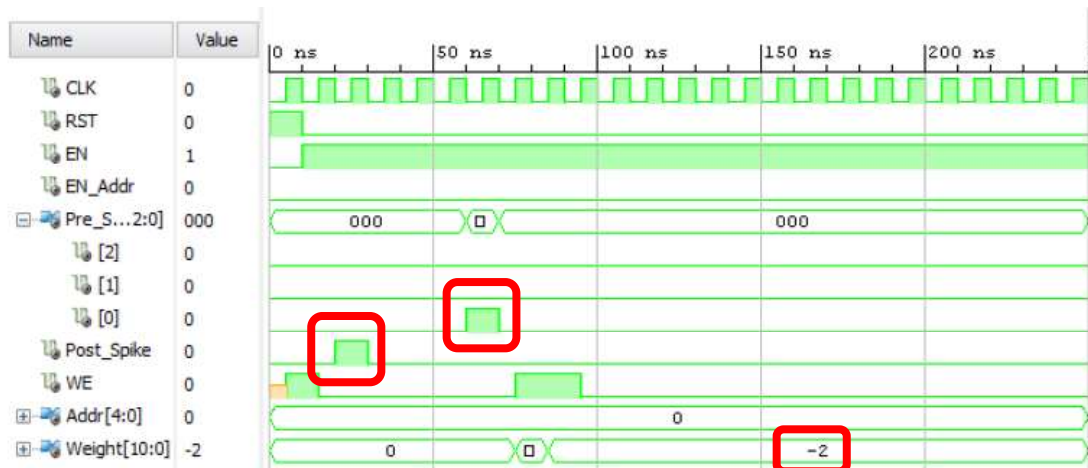
### 4.3 Analysis of Spike-based Plasticity

Adapting the SNN architecture proposed in [47] as detailed in Section 3.3.2, the mechanism and working principle of STDP is visually represented in Figure 4.15. The simulation shows a neuron connected to a previous layer of three neurons. The Pre\_Spike signal contains three binary numbers representing the synapses of that previous layer of neurons. The Post\_Spike signal goes high when the neuron fires, telling the observer that the neuron has fired.



**Figure 4.15: STDP when the pre-synaptic neuron fires before the post-synaptic neuron does.**

In Figure 4.15, one of the synapses on Pre\_Spike signal spikes at around 20ns. At 150ns, the Post\_Spike signal goes high to indicate that the post-synaptic neuron has fired. The theory applied here is that due to the neuron firing after the pre\_synaptic neuron fired, it is assumed that the pre-synaptic neuron contributed to the firing of the neuron by increasing the voltage at the membrane of the neuron, making it closer to the threshold value. From this theory, the weight of the synapse between the post-synaptic neuron and the first pore-synaptic neuron is increased from 0 to 1, indicating that the synapse has strengthened.



**Figure 4.16: STDP when the pre-synaptic neuron fires after the post-synaptic neuron does.**

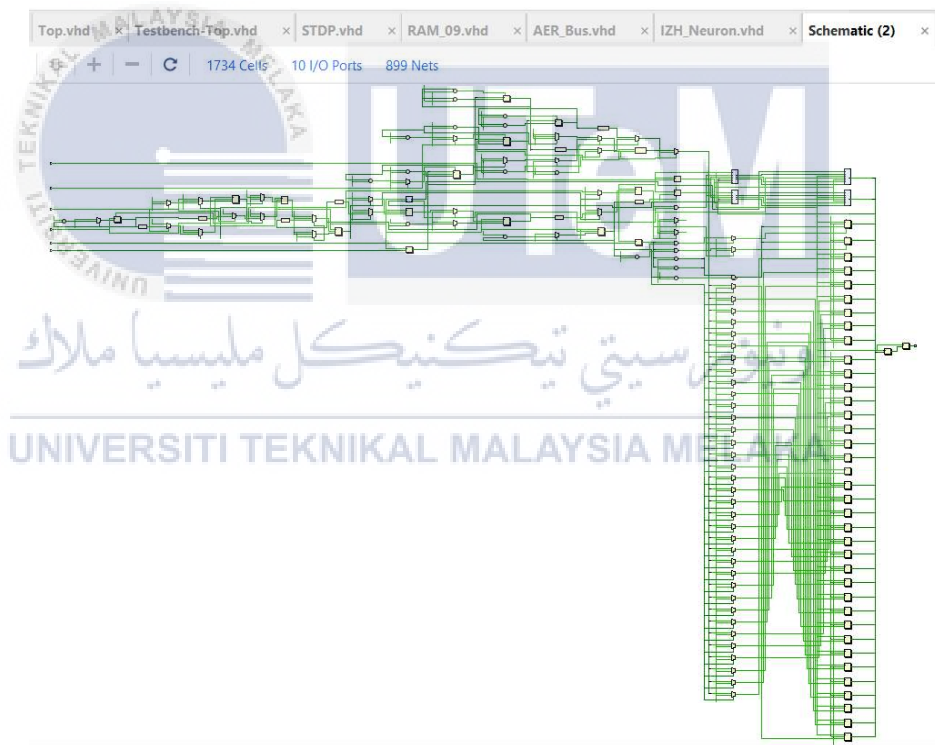
Figure 4.16 shows the opposite scenario of Figure 4.15, where the Post\_Spike signal goes high before the Pre\_Spike signal does. According to the STDP rule, it is assumed that the synapse between the first pre-synaptic neuron and the post-synaptic neuron are less significant, thus the weight of that synapse is decreased from 0 to -2. In this work, STDP rule is applied to train the neurons, where neurons 37 and 38 are the neurons in the output layer trained to recognize normal and abnormal ECG data respectively.

#### 4.4 Implementation on a Neuromorphic Circuit

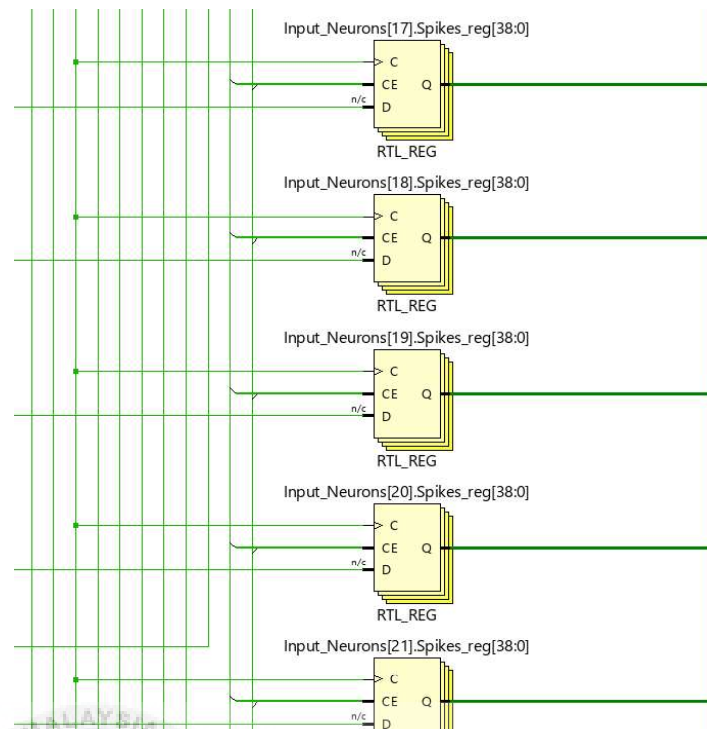
After the neuromorphic RTL design is completed and behavioral simulation shows the desired result, it can then proceed to the implementation stage. The implementation in this work is targeted for an FPGA which is Zedboard, detailed in Section 2.5.1. Several steps are involved in this process, which are performed in the Vivado environment.

#### 4.4.1 Synthesis

The completed RTL design is synthesized, which is a step where the design is converted into a netlist and can be viewed as gate-level circuits [97]. The synthesis tool will not be executed if it detects errors in the construct of the RTL code, such as syntax errors and synthesis errors. Synthesis errors may come from RTL constructs or structures used in the code which can be simulated behaviorally but cannot be logically converted into gates for the FPGA implementation. Thus, successful synthesis ensures that the VHDL code is synthesizable. It produces two schematic circuits, which are the RTL schematic and technology schematic.



**Figure 4.17: RTL schematic of the synthesized design.**

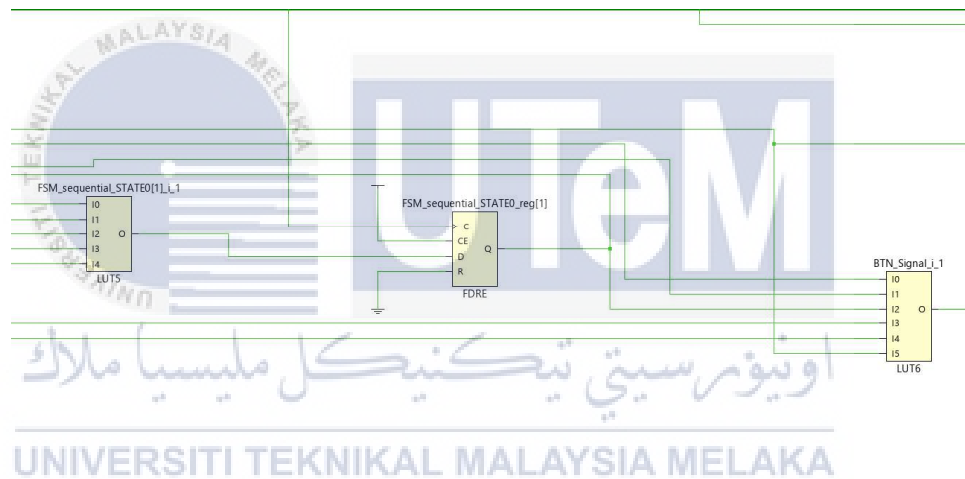


**Figure 4.18: Close-up of gates in RTL schematic.**

The RTL schematic is the design shown at earlier stages of the synthesis step, prior to completion of the technology mapping. This schematic is shown to be the closest possible graphical representation goal of the RTL design that has been coded [97]. The RTL schematics in Figures 4.17 and 4.18 show a pre-advanced layout in terms of more conventional blocks such as adders, counters and multipliers, as well as logic gates for combinatorial logic, according to the Xilinx RTL and Technology Schematic Viewers Tutorial (UG685).



**Figure 4.19: Technology schematic of the synthesized design.**



**Figure 4.20: Close-up of the Zedboard-specific elements shown in the technology schematic.**

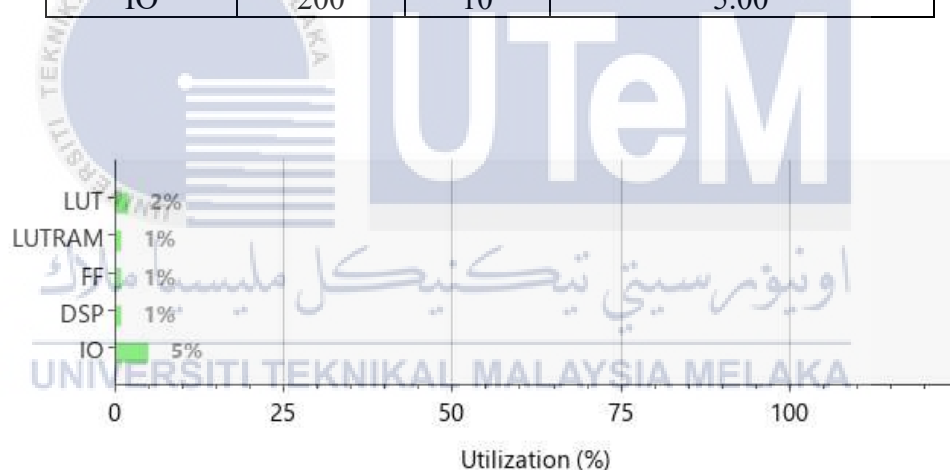
The technology schematic in Figures 4.19 and 4.20 shows more detailed information of the actual elements used in the target FPGA after the synthesis process finishes the optimization and technology targeting stage [98]. Thus, this schematic represents logic elements such as LUTs and I/O buffers which are specific to and already optimized to the Zedboard FPGA.

#### 4.4.2 Hardware Resource Utilization

The hardware utilization on the target FPGA can be calculated by Vivado, where a utilization report is generated after implementation. The utilization report shows the amount of hardware resources [99] on the Zedboard that would be used by this design implementation. The utilization of this design is tabulated in Table 4.2 and graphically represented in Figure 4.21.

**Table 4.2: Hardware resource utilization report.**

Resource	Available	Utilized	Utilization percentage (%)
LUT	53200	908	1.71
LUTRAM	17400	38	0.22
FF	106400	919	0.86
DSP	220	2	0.91
IO	200	10	5.00



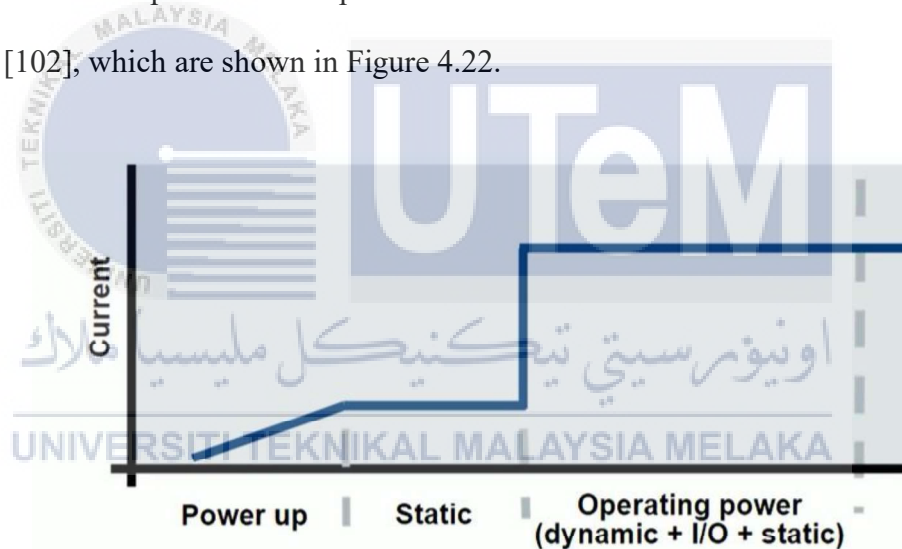
**Figure 4.21: Graphical representation of hardware resource utilization percentages.**

From the table, it is observed that the proposed design occupies 908 out of 53200 look up tables, 919 out of 106400 slice registers and 10 out of 200 bonded input outputs of the Zedboard. In total, approximately 1.058% of the Zedboard FPGA resources are utilized by the design. This indicates that less hardware is needed in the implementation of this design [100], making it more cost-effective for hardware implementation.



#### 4.4.3 Power Report

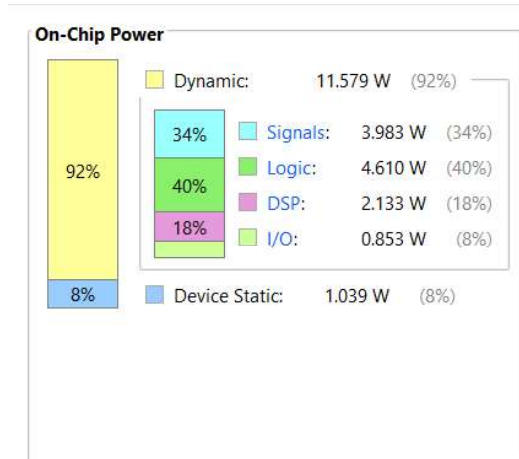
The Vivado tool allows power reports to be generated at two points in the design process, which are post-synthesis and post-implementation. There are also two modes in which the power report may be generated, which are vector-based and vectorless. The main difference between them is that vector-based mode requires a simulation activity file [101], or a SAIF file. The role of the SAIF file is to provide information on the switching activity for each individual register and net in the design. However, in this work, the power reports are based on vectorless mode, where the power usage is estimated without the simulation activity file. The power reports generated by Vivado show the power consumption when the device is in static mode and dynamic mode [102], which are shown in Figure 4.22.



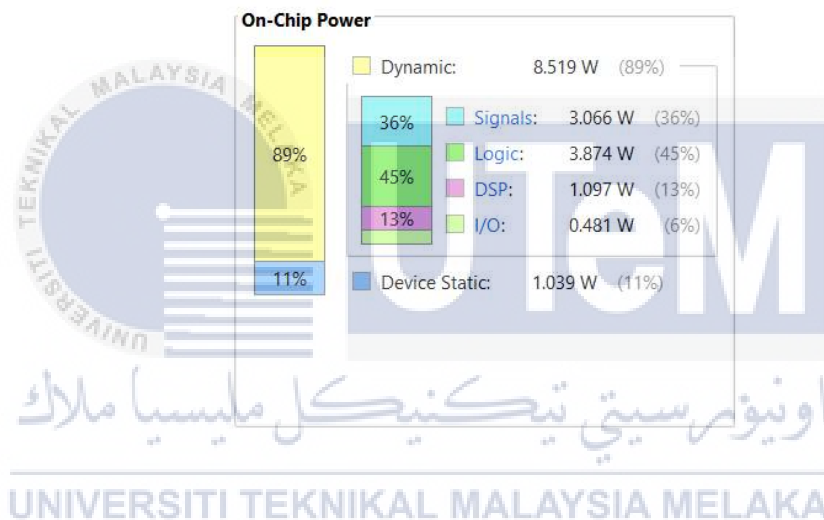
**Figure 4.22: The stages of power consumption in an FPGA.**

The power reports generated for post-synthesis and post-implementation in this work are shown in Figures 4.23 and 4.24 respectively. It is observed that the estimated power consumption after synthesis is 12.618W, 92% of which is dynamic power while the estimated power consumption after implementation is 9.558W, 89% of which is dynamic power.





**Figure 4.23: Post-synthesis power report.**



**Figure 4.24: Post-implementation power report.**

Comparing the power reports post-synthesis and post-implementation, it can be seen that the overall power estimation decreases after implementation. The post-implementation power estimation of 9.558W for a non-Von Neumann processor is comparatively lower than conventional Von Neumann processors, which can consume up to 100W [103]. Furthermore, as aforementioned, these figures are based on vectorless estimation, where the tool calculates the probability of switching rate, which may be inaccurately optimized, as the default values are used in this estimation. High switching activity prediction would cause an increase in dynamic power consumption. Thus, dynamic power consumption could still be further reduced by

optimizing the design to reduce switching rates. The switching rates should be determined based on the ECG classification function of the design, involving further simulation to generate a SAIF file.

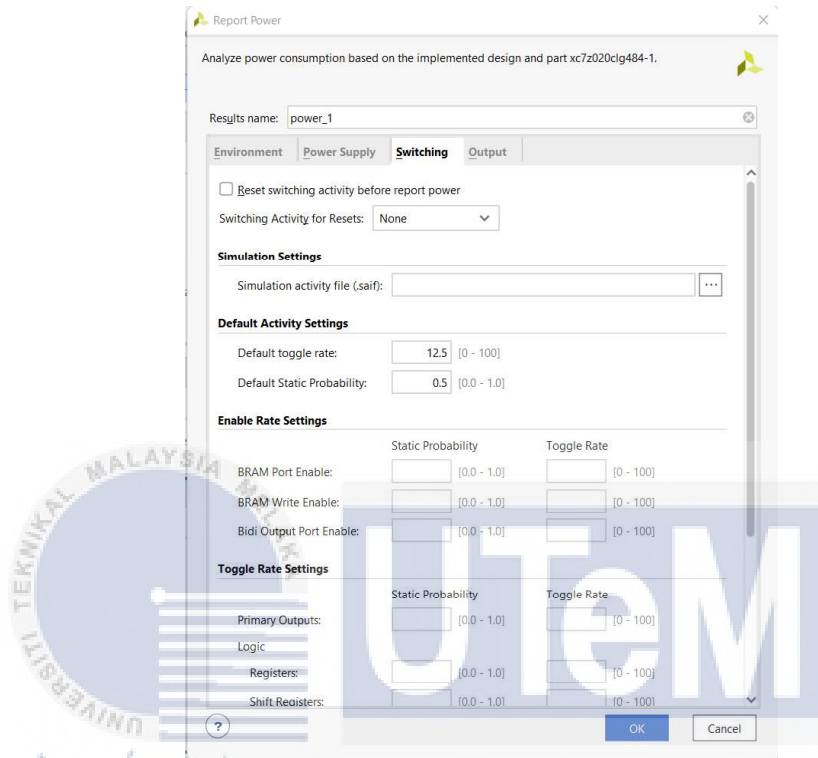


Figure 4.25: Default switching activity settings in power estimation.

#### 4.5 Performance Comparison with Previous Research

Previous research implementing ECG classification on an FPGA is tabulated in Table 4.3, including the outcome of the proposed design. It is observed that in terms of hardware resource utilization, the proposed design is rather on par with results of previous research, where conventional computing is employed on the FPGA for ECG classification. In the area of power, there is high potential for further development in future work, as described in Chapter 5.

**Table 4.3: Comparison of proposed design with previous research.**

Publication	Lookup tables	RAM	Slice registers	DSP	IO	Total
Gu <i>et al.</i> [76] (2016)	10116	91	3433	20	-	13660
Zhai <i>et al.</i> [77] (2017)	16133	17	11797	12	-	27959
Madiraju <i>et al.</i> [78] (2018)	4324	-	1540	125	30	6019
Proposed	908	38	919	2	10	1877

#### 4.6 Summary

The main contribution of this work which investigates building blocks and techniques to design a neuromorphic circuit customized for ECG classification is detailed in this chapter. The building blocks of a neuromorphic design are investigated, which are preprocessing the input data, training the SNN with it, and testing the SNN. The STDP mechanism is also analyzed by performing behavioral simulation to observe the behavior of the STDP module. The implementation of the design on a neuromorphic circuit is performed, returning a hardware resource utilization percentage of about <2% overall.

## CHAPTER 5

### CONCLUSION AND FUTURE WORKS



#### 5.1 Overview

This chapter concludes the findings and objectives achieved throughout this work in Section 5.2. In Section 5.3, future works for improvement of this work are suggested. The lifelong learning gained from the completion of this work is also outlined in Section 5.4.

#### 5.2 Conclusion

In this work, each of the objectives have been achieved. Overall, the main contribution of this work is the methods required to design a neuromorphic circuit on FPGA employing SNN to classify ECG data. This fulfills the research gap discussed in Section 1.3, which is on the implementation of neuromorphic circuits for ECG classification using FPGAs.

To achieve the first objective, the common building blocks and techniques used for a neuromorphic circuit based on SNN were investigated. The common building blocks include pre-processing, training and testing of the network. These are considered the main building blocks because they are the three high-level phases in the process of designing an SNN for neuromorphic implementation and are explored in detail in Section 4.2. The preprocessing block in this work takes an analog ECG signal and converts it into a spike train of binary digits, where the 1's indicate a spike. For the training block, the technique applied is the spike-based plasticity mechanism. Since the ECG data has two classifications which are normal data and abnormal data, the network has two training neurons, each trained to recognize one type of data. As for the testing block, the result is observed at the output neurons linked to their respective training neurons for each category. The output neurons spike accordingly, whether normal or abnormal data is inserted for testing.

For the second objective, spike-based plasticity mechanism in neuromorphic circuit is also analyzed. STDP is applied here, whereby synapses between neurons are strengthened when the pre-synaptic neuron spikes before the post-synaptic neuron does, assuming that the pre-synaptic spike contributed to the post-synaptic spike. In a neuromorphic circuit, a digital representation of this mechanism has to be implemented. The model is shown in Figure 3.15, where the STDP module has six input signals, and three output signals that write to the RAM of the corresponding neurons whether the weights are increased or decreased. The results of behavioral simulation are shown in Section 4.3, to visually represent the behavior of the STDP module.

Lastly, the third objective involves ECG implementation on a neuromorphic circuit, which has been achieved is performed in this work as well, where the design process is executed from the RTL design to synthesis and implementation. After successful behavioral simulation of ECG classification in the RTL design, the design is synthesized to ensure all elements in the design are synthesizable on an FPGA. This step outputs a gate-level netlist of the RTL design as described in Section 4.4.1. Then, the implementation stage maps the netlist onto the resources of the target FPGA according to the constraints file provided, which in this case is the Zedboard. Sections 4.4.2, 4.4.3 and 4.5 analyze the hardware resource utilization of the design and benchmark it with other papers published in the same area.

### 5.3 Future Works

Due to timing and resource constraints during the development of this work, several aspects of improvement were not able to be explored. These aspects are detailed in the following subtopics.

#### 5.3.1 Adaptive Peak Detection

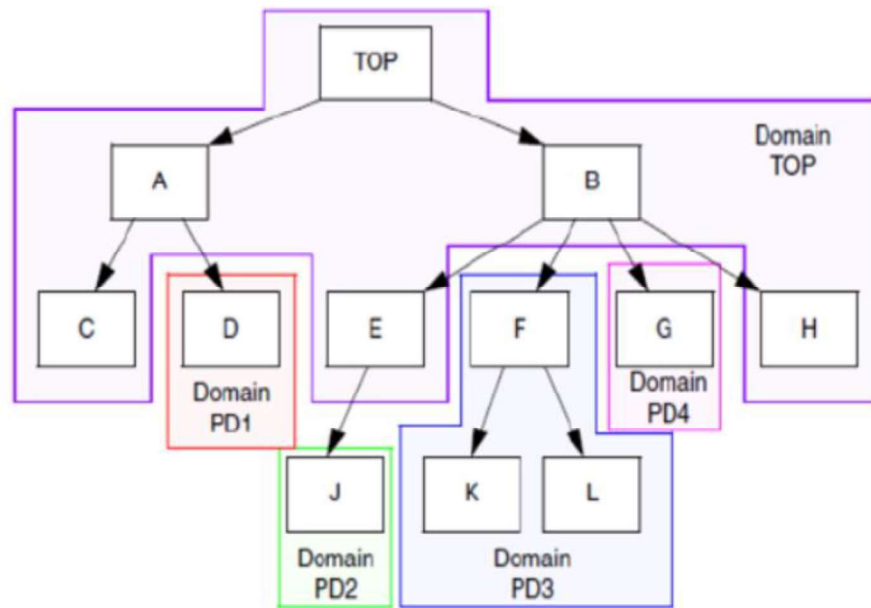
In this work, DWT is used in the preprocessing stage to filter out the R peaks in the ECG signals and encode them as binary highs. In the binarizing process, there is a thresholding step that enables the peaks to be encoded by converting points higher than the threshold to 1's and those below the threshold to 0's. This poses an issue when the same threshold value is unable to detect peaks for all ECG signals. This is illustrated in Figures 4.7 and 4.8, where a threshold value of 0.7 would be able to filter out the peaks in Record 115 (Figure 4.7) but not in Record 111 (Figure 4.8). As a result, a threshold of 0.7 is used but with manual correction for extreme cases where the thresholding value is unable to accurately detect the peaks.

In future works, adaptive thresholding could be implemented into the preprocessing system designed, where the shape of the ECG signal is viewed as a whole and the peaks, which are a small percentage of the signal that have higher amplitude than the others, can be visually detected, adaptively adjusting the threshold values applied.

### 5.3.2 Unified Power Format

In this work, the power estimation is performed using Vivado tools in vectorless mode. This means that the switching activity is only an estimate, and not modelled based on the application of the system for ECG classification. While the power consumption could be improved with a more accurate SAIF file, a better method would be applying the Unified Power Format (UPF).

UPF is the IEEE 1801 Standard for Design and Verification of Low Power Integrated Circuits, which is a collective group of commands for specifying the design intent of multi-voltage electronic systems. Using UPF, various aspects of power management can be specified using one set of power design specification commands throughout the design flow, from RTL design up to implementation. An effective technique used in UPF is power-gating, where the chip layout is divided into different power domains as illustrated in Figure 5.1. This is so that if a domain of the chip is inactive, it can be powered off to reduce power consumption.



**Figure 5.1: A visual representation of how a chip is divided into different power domains for power-gating.**

While UPF is typically applied in the designing of ASICs, the technique can also be adopted for FPGA designs [104]. In future works, the inclusion of UPF in this design could potentially lower the power consumption drastically, further stretching the benefits of neuromorphic computing.

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

### 5.3.3 Hardware Deployment

With fine-tuning and design improvements made to the proposed design in this work, the design could be deployed to hardware in future works. This could be done on the Zedboard, which is the target device this design was developed for. Besides hardware deployment being the next step after implementation of the design, it would also provide a more accurate picture of the system's functionality. In terms of ECG classification, the hardware would be a platform for the user to observe which singular category the data is classified into, rather than observing the spiking neurons. This also goes for the power consumption, as bench measurement [105] is the most accurate way to measure power consumption.



#### 5.4 Lifelong Learning

The lifelong learning aspects obtained from this work comprise of technical skills as well as academic and personal development skills. In terms of technical skills, several skills applied in this work were self-taught based on online learning materials, such as coding in VHDL, which is the HDL used for RTL design in this work. The overall process of neuromorphic circuit design and analysis in this work was also part of the technical skills that were self-taught and independently researched. This has led to the academic skills obtained throughout the completion of this work, where continuous research and reading has been done for every step of the way, instilling the ability to compare and cross-reference different materials to ensure the correct information is applied. Lastly, personal skills were developed during the completion of this work as well. This includes responsibility, self-discipline, and self-motivation to continuously progress throughout the many stages of this project.

#### 5.5 Summary

In conclusion, the three objectives of this work have been achieved. Some future works that could bring this work to its full potential include using adaptive peak detection in the preprocessing stage, writing UPF for the design, as well as deploying it on hardware. Several lifelong learning is also gained from this work which are technical, academic and personal development skills.

## REFERENCES

- [1] M. Bansal, "Cardiovascular disease and COVID-19," *Diabetes Metab. Syndr. Clin. Res. Rev.*, vol. 14, no. 3, pp. 247–250, 2020, doi: 10.1016/j.dsx.2020.03.013.
- [2] S. Parvaneh, J. Rubin, S. Babaeizadeh, and M. Xu-Wilson, "Cardiac arrhythmia detection using deep learning: A review," *J. Electrocardiol.*, vol. 57, pp. S70–S74, 2019, doi: 10.1016/j.jelectrocard.2019.08.004.
- [3] U. R. Acharya, H. Fujita, O. S. Lih, Y. Hagiwara, J. H. Tan, and M. Adam, "Automated detection of arrhythmias using different intervals of tachycardia ECG segments with convolutional neural network," *Inf. Sci. (Ny.)*, vol. 405, pp. 81–90, 2017, doi: 10.1016/j.ins.2017.04.012.
- [4] W. Damelin, S. B., & Miller Jr, *The mathematics of signal processing (No. 48)*. Cambridge University Press, 2012.
- [5] Pavel and S. David, "Algorithms for efficient computation of convolution," in *Design and Architectures for Digital Signal Processing*, InTech, 2013, pp. 179–209.

- [6] R. Tang, W. Wang, Z. Tu, and J. Lin, "AN EXPERIMENTAL ANALYSIS OF THE POWER CONSUMPTION OF CONVOLUTIONAL NEURAL NETWORKS FOR KEYWORD SPOTTING Raphael Tang Weijie Wang Jimmy Lin David R . Cheriton School of Computer Science University of Waterloo," *2018 IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 5479–5483, 2018.
- [7] A. Balaji, F. Corradi, A. Das, S. Pande, S. Schaafsma, and F. Catthoor, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *J. Low Power Electron.*, vol. 14, no. 4, pp. 508–519, 2018, doi: 10.1166/jolpe.2018.1582.
- [8] Z. Yan, J. Zhou, and W. F. Wong, "Energy efficient ECG classification with spiking neural network," *Biomed. Signal Process. Control*, vol. 63, no. May 2020, p. 102170, 2021, doi: 10.1016/j.bspc.2020.102170.
- [9] J. Upadhyay, N. K., Jiang, H., Wang, Z., Asapu, S., Xia, Q., & Joshua Yang, "Emerging memory devices for neuromorphic computing," *Adv. Mater. Technol.*, vol. 4, no. 4, 2019.
- [10] W. Covi, E., Donati, E., Liang, X., Kappel, D., Heidari, H., Payvand, M., & Wang, "Adaptive extreme edge computing for wearable devices," *Front. Neurosci.*, vol. 15, 2021.
- [11] M. F. Chen, W. H., Khwa, W. S., Li, J. Y., Lin, W. Y., Lin, H. T., Liu, Y., ... & Chang, "Circuit design for beyond von Neumann applications using emerging memory: From nonvolatile logics to neuromorphic computing," *2017 18th Int. Symp. Qual. Electron. Des.*, no. 23–28.

- [12] Z. Dai, S., Chen, D., Xiong, F., & Chen, “Motion Artifact Reduction in Electrocardiogram Using Adaptive Filtering Based on Skin-Potential Variation Monitoring,” in *EAI International Conference on Body Area Networks*, pp. 465–472.
- [13] A. Rana and K. K. Kim, “A Novel Spiking Neural Network for ECG signal Classification,” *J. Sens. Sci. Technol.*, vol. 30, no. 1, pp. 20–24, 2021, doi: 10.46670/jsst.2021.30.1.20.
- [14] H. Martis, R. J., Acharya, U. R., & Adeli, “Current methods in electrocardiogram characterization,” *Comput. Biol. Med.*, vol. 48, pp. 133–149, 2014.
- [15] K. Celin, S., & Vasanth, “ECG signal classification using various machine learning techniques,” *J. Med. Syst.*, vol. 42, no. 12, pp. 1–11, 2018.
- [16] L. P. and P. W. L. Biel, O. Pettersson, “ECG analysis: A new approach in human identifications,” *IEEE Trans. Instrum. Meas.*, vol. 50, no. 3, pp. 808–812, 2001.
- [17] P. R. F. do Vale Madeiro, J. P., Cortez, P. C., da Silva Monteiro Filho, J. M., & Rodrigues, “Techniques for Noise Suppression for ECG Signal Processing,” *Dev. Appl. ECG Signal Process.*, pp. 53–87, 2019.
- [18] S. Jadhav, D. G., Pattnaik, S. S., & Das, “Memetic algorithm with local search as modified swine influenza model-based optimization and its use in ECG filtering,” *J. Optim.*, 2014.
- [19] O. Lenis, G., Pilia, N., Loewe, A., Schulze, W. H., & Dössel, “Comparison of

baseline wander removal techniques considering the preservation of ST changes in the ischemic ECG: a simulation study,” *Comput. Math. Methods Med.*, 2017.

- [20] J. P. Antman, E. M., Anbe, D. T., Armstrong, P. W., Bates, E. R., Green, L. A., Hand, M., ... & Ornato, “ACC/AHA guidelines for the management of patients with ST-elevation myocardial infarction: a report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines (Committee to Revise the 1999 Guidelines for the Managem,” *J. Am. Coll. Cardiol.*, vol. 44, no. 3, pp. 1–211, 2004.
- [21] V. Verma, R., Mehrotra, R., & Bhateja, “A new morphological filtering algorithm for pre-processing of electrocardiographic signals,” in *Proceedings of the Fourth International Conference on Signal and Image Processing 2012 (ICSIP 2012)*, 2013, pp. 193–201.
- [22] J. X. Wan, X. K., Wu, H., Qiào, F., Li, F. C., Li, Y., Yan, Y. W., & Wei, “Electrocardiogram baseline wander suppression based on the combination of morphological and wavelet transformation-based filtering,” *Comput. Math. Methods Med.*, 2019.
- [23] B. Liu, Z., Wang, J., & Liu, “ECG signal denoising based on morphological filtering,” in *2011 5th International Conference on Bioinformatics and Biomedical Engineering*, pp. 1–4.
- [24] S. M. Sun, Y., Chan, K. L., & Krishnan, “ECG signal conditioning by morphological filtering,” *Comput. Biol. Med.*, vol. 32, no. 6, pp. 465–479, 2002.

- [25] C. D. Schuman, J. S. Plank, G. Bruer, and J. Anantharaj, “Non-Traditional Input Encoding Schemes for Spiking Neuromorphic Systems,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2019-July, 2019, doi: 10.1109/IJCNN.2019.8852139.
- [26] N. Sengupta, N., & Kasabov, “Spike-time encoding as a data compression technique for pattern recognition of temporal data,” *Inf. Sci. (Ny)*, vol. 406, pp. 133–145, 2017.
- [27] A. Sboev, A., Vlasov, D., Rybka, R., & Serenko, “Solving a classification task by spiking neurons with stdp and temporal coding,” *Procedia Comput. Sci.*, vol. 23, pp. 494-500., 2018.
- [28] J. V. Arthur and K. Boahen, “Learning in silicon: Timing is everything,” in , pp. 7.],” *Adv. Neural Inf. Process. Syst.*, pp. 75–82, 2006.
- [29] R. M. Petro, B., Kasabov, N., & Kiss, “Selection and optimization of temporal spike encoding methods for spiking neural networks,” *IEEE Trans. neural networks Learn. Syst.*, vol. 31, no. 2, pp. 358–370, 2019.
- [30] N. E. Hough, M., De Garis, H., Korkin, M., Gers, F., & Nawa, “SPIKER: Analog waveform to digital spiketrain conversion in ATR’s artificial brain (cam-brain) project,” *nternational Conf. Robot. Artif. life (Vol. 92)*. Citeseer.
- [31] J. Schrauwen, B., & Van Campenhout, “BSA, a fast and accurate spike train encoding scheme. In . (Vol. 4, pp. ). I.,” in *Proceedings of the International Joint Conference on Neural Networks, 2003*, pp. 2825–2830.
- [32] P. S. No, M. Alhamdi, and S. No, “Analysis of Human Electrocardiogram for Arrhythmia Auto- Classification and Biometric Recognition Systems Using

Analytic and Autoregressive Modeling Parameters,” 2015.

- [33] X. Zhang, D., Wang, S., Li, F., Wang, J., Sangaiah, A. K., Sheng, V. S., & Ding, “An ECG signal de-noising approach based on wavelet energy and sub-band smoothing filter,” *Appl. Sci.*, vol. 9, no. 22, p. 4968, 2019.
- [34] A. Daqrouq, K., Ajour, M., Al-Qawasmi, A. R., & Alkhateeb, “The discrete wavelet transform based electrocardiographic baseline wander reduction method for better signal diagnosis,” *J. Med. Imaging Heal. Informatics*, vol. 8, no. 8, pp. 1590–1597, 2018.
- [35] A. Aqil, M., Jbari, A., & Bourouhou, “ECG Signal Denoising by Discrete Wavelet Transform,” *Int. J. Online Eng.*, vol. 13, no. 9, 2017.
- [36] E. M. Shemi, P. M., & Shareena, “Analysis of ECG signal denoising using discrete wavelet transform,” in *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, 2016, pp. 713–718.
- [37] S. A. Ahmed, A. S., Rijab, K. S., & Alagha, “A Study of Chosen an Optimum Type of Wavelet Filter for De-Noising an ECG signal,” 2020.
- [38] K. J. M. Syama, S., Sweta, G. S., Kavyasree, P. I. K., & Reddy, “Classification of ECG signal using machine learning techniques,” in *2019 2nd International Conference on Power and Embedded Drive Control (ICPEDC)*, 2019, pp. 122–128.
- [39] R. Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., Adam, M., Gertych, A., & San Tan, “A deep convolutional neural network model to classify heartbeats,” *Comput. Biol. Med.*, vol. 89, pp. 389–396, 2017.

- [40] G. A. Infantolino, Z., & Miller, “Psychophysiological methods in neuroscience,” in *Noba Textbook Series: Psychology*, DEF Publishers, Champaign, 2019.
- [41] D. Ghosal, N. Majumder, S. Poria, N. Chhaya, and A. Gelbukh, “DialogueGCN: A graph convolutional neural network for emotion recognition in conversation,” *EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, vol. 2, pp. 154–164, 2020, doi: 10.18653/v1/d19-1015.
- [42] S. Albawi, S., Mohammed, T. A., & Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6.
- [43] Y. Guo, T., Dong, J., Li, H., & Gao, “Simple convolutional neural network on image classification,” in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pp. 721–724.
- [44] V. Beer, M., Urenda, J., Kosheleva, O., & Kreinovich, “Why Spiking Neural Networks Are Efficient: A Theorem,” in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 59–69.
- [45] S. Lamba and R. Lamba, “Spiking Neural Networks Vs Convolutional Neural Networks for Supervised Learning,” *Proc. - 2019 Int. Conf. Comput. Commun. Intell. Syst. ICCIS 2019*, vol. 2019-Janua, no. 1, pp. 15–19, 2019, doi: 10.1109/ICCIS48478.2019.8974507.



- [46] et al. Moore, Simon W., Paul J. Fox, “Bluehive - A Field-Programmable Custom Computing Machine for Extreme-Scale Real-Time Neural Network Simulation,” in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*, 2012, pp. 133–140.
- [47] E. G. Merino Mallorquí, “Digital system for spiking neural network emulation,” Universitat Politècnica de Catalunya, 2017.
- [48] J. C. Bittner, K. C., Milstein, A. D., Grienberger, C., Romani, S., & Magee, “Behavioral time scale synaptic plasticity underlies CA1 place fields,” *Science (80-. )*, vol. 357, no. 6355, pp. 1033–1036, 2017.
- [49] O. Brzosko, Z., Mierau, S. B., & Paulsen, “Neuromodulation of Spike-Timing-Dependent plasticity: past, present, and future,” *Neuron*, vol. 103, no. 4, pp. 563–581, 2019.
- [50] D. Soni, “Spiking neural networks, the next generation of machine learning,” *Data Sci. Mach. Learn.*, 2010.
- [51] G. Bauer, F. C., Muir, D. R., & Indiveri, “Real-time ultra-low power ECG anomaly detection using an event-driven neuromorphic processor,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 6, pp. 1575–1582, 2019.
- [52] A. Amirshahi and M. Hashemi, “ECG Classification Algorithm Based on STDP and R-STDP Neural Networks for Real-Time Monitoring on Ultra Low-Power Personal Wearable Devices,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 6, pp. 1483–1493, 2019, doi: 10.1109/TBCAS.2019.2948920.
- [53] A. E. Kolagasioglu, “Energy Efficient Feature Extraction for Single-Lead ECG



- [61] H. Ismail, A. A., Shaheen, Z. A., Rashad, O., Salama, K. N., & Mostafa, “A Low Power Hardware Implementation of Izhikevich Neuron using Stochastic Computing,” in *2018 30th International Conference on Microelectronics (ICM)*, 2018, pp. 315–318.
- [62] L. Lapicque, “Definition experimentale de l’excitabilite,” *Soc Biol*, vol. 77, pp. 280–283, 1909.
- [63] M. C. Brunel, N., & Van Rossum, “Lapicque’s 1907 paper: from frogs to integrate-and-fire,” *Biol. Cybern.*, vol. 97, no. 5, pp. 337–339, 2007.
- [64] G. Moore, “Moore’s law,” *Electronics Magazine* 38(8), p. 114, 1965.
- [65] W. Maass, “Noise as a Resource for Computation and Learning in Networks of Spiking Neurons,” in *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880.
- [66] W. McCulloch, W.S., Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.
- [67] S. Ielmini, D., & Ambrogio, “Emerging neuromorphic devices,” *Nanotechnology*, vol. 31, no. 9, 2019.
- [68] M. D. et Al., “Loihi: A Neuromorphic Manycore Processor with On-Chip Learning,” *IEEE Micro*, vol. 38, no. 1, pp. 82–99.
- [69] J. Hsu, “IBM’s new brain [News],” *IEEE Spectr.*, vol. 51, no. 10, pp. 17–19, 2014.
- [70] H. Lin, C. K., Wild, A., Chinya, G. N., Cao, Y., Davies, M., Lavery, D. M., & Wang, “Programming spiking neural networks on Intel’s Loihi,” *Computer*

(*Long. Beach. Calif.*), vol. 51, no. 3, pp. 52–61, 2018.

- [71] M. V. D. J. Sawada, F. Akopyan, A. S. Cassidy, B. Taba and R. A. P. Datta, R. Alvarez-Icaza, A. Amir, J. V. Arthur, A. Andreopoulos, “TrueNorth Ecosystem for Brain-Inspired Computing: Scalable Systems, Software, and Applications,” 2016.
- [72] D. S. Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., ... & Modha, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science (80-. )*, vol. 345, no. 6197, pp. 668–673, 2014.
- [73] F. Vandesompele, A., Walter, F., & Röhrbein, “Neuro-evolution of spiking neural networks on SpiNNaker neuromorphic hardware,” *2016 IEEE Symp. Ser. Comput. Intell.*, pp. 1–6.
- [74] L. A. Furber, S. B., Galluppi, F., Temple, S., & Plana, “The spinnaker project,” in *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665.
- [75] A. Valancius, S., Richter, E., Purdy, R., Rockowitz, K., Inouye, M., Mack, J., ... & Akoglu, “FPGA Based Emulation Environment for Neuromorphic Architectures,” in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020, pp. 90–97.
- [76] G. Gu, X., Zhu, Y., Zhou, S., Wang, C., Qiu, M., & Wang, “A real-time FPGA-based accelerator for ECG analysis and diagnosis using association-rule mining,” *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 2, pp. 1–23, 2016.
- [77] F. Zhai, X., Ali, A. A. S., Amira, A., & Bensaali, “ECG encryption and

identification based security solution on the Zynq SoC for connected health systems,” *J. Parallel Distrib. Comput.*, vol. 106, pp. 143–152, 2017.

- [78] R. Madiraju, N. S., Kurella, N., & Valapudasu, “FPGA Implementation of ECG feature extraction using Time domain analysis,” 2018.
- [79] R. Talukder, S., Singh, R., Bora, S., & Paily, “An efficient architecture for QRS detection in FPGA using integer Haar wavelet transform,” *Circuits, Syst. Signal Process.*, pp. 1–16, 2020.
- [80] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun, “Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review,” *Comput. Biol. Med.*, vol. 122, no. December 2019, p. 103801, 2020, doi: 10.1016/j.combiomed.2020.103801.
- [81] R. G. M. G. B. Moody, “MIT-BIH arrhythmia database,” 1992. <https://physionet.org/physiobank/database/mitdb/> (accessed May 02, 2021).
- [82] and H. E. S. A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, “PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals,” *Circulation*, vol. 101, no. 23, pp. 215–220.
- [83] D. L. Donoho and I. M. Johnstone, “Adapting to Unknown Smoothness via Wavelet Shrinkage,” *J. Am. Stat. Assoc.*, vol. 90, no. 432, p. 1200.
- [84] T. Ogden, *Essential Wavelets for Statistical Applications and Data Analysis*. Birkhäuser, 1996.

- [85] M. A. Mohamed and M. A. Deriche, "An Approach for ECG Feature Extraction using Daubechies 4 (DB4) Wavelet," *Int. J. Comput. Appl.*, vol. 96, no. 12, pp. 36–41, 2014, doi: 10.5120/16850-6712.
- [86] P. Sörnmo, L., & Laguna, "Evoked Potentials," in *Bioelectrical signal processing in cardiac and neurological applications (Vol. 8)*, Academic Press, 2005, pp. 181–336.
- [87] K. et al. Mariyappa, N., Sengottuvel, S., Rajesh, P. C. & Parasakthi, "Denoising of multichannel MCG data by the combination of EEMD and ICA and its effect on the pseudo current density maps," *Biomed. Signal Process. Control*, vol. 18, pp. 204–213, 2015.
- [88] A. Goodfellow, I., Bengio, Y., & Courville, *Deep learning*. MIT press., 2016.
- [89] J. A. Thomas, D. E., Lagnese, E. D., Walker, R. A., Rajan, J. V., Blackburn, R. L., & Nestor, *Algorithmic and Register-Transfer Level Synthesis: The System Architect's Workbench: The System Architect's Workbench*, 85th ed. Springer Science & Business, 1989.
- [90] M. A. Tambara, L. A., Kastensmidt, F. L., Medina, N. H., Added, N., Aguiar, V. A., Aguirre, F., ... & Silveira, "Heavy ions induced single event upsets testing of the 28 nm xilinx zynq-7000 all programmable soc," *2015 IEEE Radiat. Eff. Data Work.*, pp. 1–6.
- [91] M. Bobin, C., Bichler, O., Lourenço, V., Thiam, C., & Thévenin, "Real-time radionuclide identification in  $\gamma$ -emitter mixtures based on spiking neural network," *Appl. Radiat. Isot.*, vol. 109, pp. 405–409, 2016.

- [92] F. Corradi *et al.*, “ECG-based Heartbeat Classification in Neuromorphic Hardware,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2019-July, no. July, pp. 1–8, 2019, doi: 10.1109/IJCNN.2019.8852279.
- [93] Z. F. M. Apandi, R. Ikeura, and S. Hayakawa, “Arrhythmia Detection Using MIT-BIH Dataset: A Review,” *2018 Int. Conf. Comput. Approach Smart Syst. Des. Appl. ICASSDA 2018*, 2018, doi: 10.1109/ICASSDA.2018.8477620.
- [94] M. Wess, “Neural Network based Electrocardiography Anomaly Detection,” no. 0926401.
- [95] D. Sundararajan, *Discrete wavelet transform: a signal processing approach*. John Wiley & Sons, 2016.
- [96] G. Y. Kim, S. H., & Hong, “Performance Improvement of Aerial Images Taken by UAV Using Daubechies Stationary Wavelet,” *J. Adv. Navig. Technol.*, vol. 20, no. 6, pp. 539–543, 2016.
- [97] R. Kalaivaani, P. T., & Krishnamoorthi, “Design and implementation of low power bio signal sensors for wireless body sensing network applications,” *Microprocess. Microsyst.*, vol. 79, 2020.
- [98] V. Taraate, *VHDL Design and RTL Tweaks*. Springer, 2020.
- [99] R. Sujina, S., & Remya, “An Effective Method For Hardware Multiplication Using Vedic Mathematics,” in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2018, pp. 1499–1504.
- [100] T. Baloch, A., Memon, T. D., Memon, F., Lal, B., Viyas, V., & Jan, “Hardware

synthesize and performance analysis of intelligent transportation using canny edge detection algorithm,” *Int. J. Eng. Manuf.(IJEM)*, vol. 11, pp. 22–32, 2021.

- [101] M. He, M., Park, J., Nahiyani, A., Vassilev, A., Jin, Y., & Tehranipoor, “RTL-PSC: Automated power side-channel leakage assessment at register-transfer level,” *2019 IEEE 37th VLSI Test Symp.*, pp. 1–6.
- [102] R. Seeram, S. S. S. G., Polireddi, S. N. N., Somanathan, G. R., & Bhakthavatchalu, “Synthesis of Synchronous Gray Code Counters by Combining Mentor Graphics HDL Designer and Xilinx VIVADO FPGA Flow,” *2020 Int. Conf. Commun. Signal Process.*, pp. 738–742.
- [103] C. K. Inoue, K., & Pham, “The Memorism Processor: Towards a Memory-Based Artificially Intelligence Complementing the von Neumann Architecture,” *SICE J. Control. Meas. Syst. Integr.*, vol. 10, no. 6, pp. 544–550, 2017.
- [104] D. Macko, “Adoption of abstract power-management specification to FPGA-based design,” in *2016 International Conference on Emerging eLearning Technologies and Applications (ICETA)*, 2016, pp. 199–204.
- [105] J. Scheer, R., Bergheim, Y., Heintges, D., Rahner, N., Gries, R., & Andert, “An FPGA-based Real-time Spatial Harmonics Model of a PMSM Considering Iron Losses and the Thermal Impact,” *IEEE Trans. Transp. Electr.*, 2021.