2022

Hardware Implementations of Spiking Neural Networks and Artificially Intelligent Systems

Alexander J. Leigh
University of Windsor

Hardware Implementations of Spiking Neural Networks and Artificially Intelligent Systems

By

Alexander J. Leigh

A Dissertation
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2022

Hardware Implementations of Spiking Neural Networks and Artificially Intelligent Systems

By

Alexander J. Leigh

APPROVED BY:

S. Mirabbasi, External Examiner
University of British Columbia

R. Carriveau
Department of Civil & Environmental Engineering

M. Azzouz
Department of Electrical & Computer Engineering

B. Balasingam
Department of Electrical & Computer Engineering

R. Muscedere, Co-Advisor
Department of Electrical & Computer Engineering

M. Mirhassani, Co-Advisor
Department of Electrical & Computer Engineering

April 26^{th}, 2022

Declaration of Co-Authorship/Previous Publication

I. Co-Authorship

I hereby declare that this dissertation incorporates material that is result of joint research. Chapters 2-6 of this dissertation were completed under the supervision of Dr. Mitra Mirhassani. Additionally, Chapter 2 was completed under the additional supervision of Dr. Roberto Muscedere. Furthermore, Chapters 3-6 were completed in colaboration with Dr. Moslem Heidarpur. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, statistical analysis, graphing results, and writing, were performed by the author. The contributions of the above-mentioned co-authors were primarily through the provision of checking and comments on the literature review, mathematical derivations, systems architectures, algorithms, providing feedback on refinement of ideas, editing of the manuscript, and advice on selecting peer reviewed journals for publication.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my dissertation, and have obtained written permission from the supervisor to include the above material in my dissertation.

I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

II. Previous Publication

This dissertation includes 3 original papers that have been previously published/accepted for publication in peer reviewed journals and conferences as described below.

Dissertation Chapter	**Publication title/full citation**	**Publication Status**
Chapter 2	A. J. Leigh, M. Mirhassani and R. Muscedere , "An Efficient Spiking Neuron Hardware System Based on the Hardware Oriented Modified Izhikevich Neuron (HOMIN) Model," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 12, pp. 3377 3381, Dec. 2020, doi : 10.1109/TCSII.	Published
Chapter 3	A. J. Leigh, M. Heidarpur and M. Mirhassani,"A Resource-Efficient and High-Accuracy CORDIC-Based Digital Implementation of the Hodgkin-Huxley Neuron", in IEEE Transactions on Biomedical Circuits and Systems	In Revision
Chapter 4	A. J. Leigh, M. Heidarpur and M. Mirhassani,"The Input-DEpendent Variable Sampling (I-DEVS) Method"	Submitted
Chapter 5	A. J. Leigh, M. Heidarpur and M. Mirhassani, "Selective Input Sparsity in Spiking Neural Networks for Pattern Classification", IEEE International Symposium on Circuits and Systems (ISCAS) 2022, May 2022	Accepted by ISCAS 2022

III. General

I declare that, to the best of my knowledge, this dissertation does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with standard referencing practices.

I declare that this is a true copy of the dissertation, including any final revisions, as approved by the dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

Abstract

Artificial spiking neural networks are gaining increasing prominence due to their potential advantages over traditional, time-static artificial neural networks. Custom hardware implementations of spiking neural networks present many advantages over other implementation mediums. Two main topics are the focus of this work. Firstly, digital hardware implementations of spiking neurons and neuromorphic hardware are explored and presented. These implementations include novel implementations for lowered digital hardware requirements and reduced power consumption.

The second section of this work proposes a novel method for selectively adding sparsity to a spiking neural network based on training set images for pattern recognition applications, thereby greatly reducing the inference time required in a digital hardware implementation.

To my grandfather

Acknowledgments

I would like to firstly thank my advisor Dr. Mitra Mirhassani. Not only has she been an exceptional advisor throughout my entire academic career, but she is a truly great person who cares profoundly for her students. Without her guidance, I can unequivocally say that I would not be where I am today academically.

I would also like to express my gratitude to my co-advisor Dr. Roberto Muscedere. He is a wealth of technical knowledge, and his input is always invaluable and significant.

I would like to thank Dr. Moslem Heidarpur, with whom I have had the august pleasure of collaboration in research.

I would like to thank my committee members; Dr. Maher Azzouz, Dr. Bala Balasingam, and Dr. Rupp Carriveau.

I would like to thank my parents and family for their continuous love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Artificial Intelligence (AI) is currently an active and exciting field with ever-expanding applications and relevance. Software and hardware AI systems have seen widespread use in various applications.

AI systems draw inspiration from naturally occurring biological neural systems [1]. The basic elements of biological neural systems are neurons and synapses. Neurons receive stimulus in the form of electric current and transmit information through spikes in their membrane potential [2]. The information is encoded in the timing and the rate of these spikes through mechanisms such as rate coding, although other methods have been proposed [3]. Neurons are connected to each other through synapses and the strength of the connection is called the synaptic weight.

Traditional Artificial Neural Networks (ANNs) use a time-static approach in which the spiking rate of the neuron is modelled by a transfer function that returns a value that is a percentage of what the spiking rate of a spiking neuron would be. Sigmoidal or linear transfer functions are common in traditional ANNs, but not obligatory [1]. Although this approach is useful, it is not a true emulation of biological systems since neurons are not linear nor time-invariant. Spiking Neural

Networks (SNNs) offer a more complete replication of biological neural systems with extended applications compared to traditional ANNs.

1.1 Elements of Spiking Neural Networks

1.1.1 Spiking Neuron Models

The spiking behaviours of biological neurons are quite diverse and many mathematical models with varying description levels have been proposed to model their behaviours. nearly all models are composed of differential equation(s) of the neuron's membrane potential.

Among the most biologically detailed models are the Hodgkin-Huxley neuron model [4] and the Morris-Lecar model [5]. Low-level, high detail models offer more accurate and meaningful descriptions of natural neurological systems. In the case of the Hodgkin-Huxley neuron model [4], all of the behavioural parameters used in the model have physiological meaning which makes the model very descriptive. Furthermore, the Hodgkin-Huxley neuron was the first conductance-based neuron model and is the basis of conductance-based neuron modeling [6]. However, the primary drawback of detailed models is their typically high computational intensity. In the case of the Hodgkin-Huxley neuron model, there are four interdependent differential equations with six parameters that depend on the state of the membrane potential. The complexity of this level of neuron model elongates software simulation and complicates potential hardware implementations.

Other models offer a less detailed description but are still able to capture all of the neuron's behaviours. These models include the Izhikevich model [7], the Adaptive-Exponential Integrate-and-Fire model [8], and the Wilson Model [9]. Models of this level of biological detail tend to be a compromise between complexity and biological parallelism. They are able to replicate the behaviours found in biological neurons,

but their modeling parameters do not necessarily directly model a physiological characteristic of a neuron.

A simpler, phenomenological description of neuronal behaviour can be found in the Leaky Integrate-and-Fire (LIF) Neuron Model [10]. This model was originally proposed in 1907 and offers a very simple explanation of the spiking behaviours observed in neurons.

Each level of biological detail clearly has advantages and drawbacks and model selection should be heavily influenced by the target application. Applications and neuron model fitness for a given application will be elaborated further subsequently.

1.1.2 Synapses and Learning

Many learning mechanisms have been proposed to model and explain biological learning, but among the most prominent learning mechanisms is Spike-Timing-Dependent Plasticity (STDP) [11]. STDP is a positive feedback-based learning mechanism in which the timing of the spikes in the membrane potentials of a given pre-synaptic and post-synaptic neuron pair is compared. If the pre-synaptic neuron spikes before the post-synaptic neuron, then the strength of the synaptic connection, that is, the synaptic weight, increases. In contrast, if the post-synaptic neuron spikes before the pre-synaptic neuron, then the synaptic weight decreases.

Furthermore, recent explorations have found that traditional ANNs can be trained using well defined methods, such as stochastic gradient descent, and the trained weights can be transferred to SNNs of analogous structure [12], and more complicated network structures may only require weight adjustment [13, 14]. This methodology has been used with significant success in hardware implementations of SNNs [15].

1.2 Motivation

1.2.1 Neuron Implementations

Since the fundamental unit of a neural network is the neuron, the design of a neuron in hardware is critical. To emulate biology, the neuron must be energy-efficient and require minimal hardware resources. Additionally, the neuron model selected for a network is an important consideration. Some applications, such as neurological disease modeling require high biological parallelism from the neuron [16], while other applications may not require a high level of biological detail.

Thus, two neuron model implementations are presented in Chapters 2 and 3 with different levels of biological detail and varying accuracy/resource trade-offs as they are presented with different intended applications.

1.2.2 Spiking Neural Network Implementations

Software-based AI solutions have excelled in many applications and have seen widespread use [17]. However, when compared to custom hardware solutions and/or programmable logic such as Field Programmable Gate Array (FPGA) solutions, software solutions are far slower and consume a great deal more power [17, 18]. Given the incredible amounts of data generated and processed in modern machine learning and AI systems, speed and power consumption are increasingly important considerations.

Unlike traditional ANNs, SNNs are time dynamic and have extended applications in real-time control problems [19] and neuronal disease modelling [20], among other fields. Additionally, the spike-based temporally sparse operation of SNNs may lend itself to reduced power consumption in hardware implementations [12, 18, 21].

Within the realm of hardware implementation many options exist. Analog Application-Specific Integrated Circuit (ASIC) implementations typically offer the lowest power consumption and excellent speed, but require extensive design overhead, are often very inflexible after the design phase, and are typically more susceptible to error and noise [22]. Moreover, digital ASIC is more robust to noise, but again suffers from large design time and limited flexibility after fabrication. FPGA solutions offer high flexibility and rapid prototyping as well as many of the speed and power benefits of ASIC to a lesser degree.

Thus, given the potential benefits of hardware implementations of SNNs, an FPGA implementation of an SNN is a logical target. Many hardware SNNs have been proposed in academic literature [15,23,24] with various advantages and drawbacks. Many proposals focus on implementing STDP learning in hardware [25, 26], but this imposes limitations on the complexity of the problem the system can solve since these approaches, even in software require an extensive number of synapses [27, 28].

Therefore, for practical applications, pre-trained SNN hardware provides an efficient solution to practical applications that require on-edge inference. The work of the second part of this dissertation presents the development of digital hardware implementations of SNNs for edge inference in pattern recognition. Simple architectures are employed and novel methods for network size reduction compared to a baseline fully-connected architecture are explored and presented and the results are compared with the baseline implementation.

1.3 Objectives

This dissertation has two main objectives. The first is the evaluation of hardware implementation methods and novel implementation techniques for neuromorphic hardware. The second is the development of novel methods by which the size of

SNNs can be reduced to result in faster inferences and lower silicon area requirements in digital hardware.

1.4 Dissertation structure

The dissertation is divided into two main parts. The first part, comprised of Chapters 2 - 4 presents works that firstly target improvements to FPGA hardware implementations of spiking neurons. Several neuron models are investigated, and results are presented and discussed. Subsequently, the second part, comprised of Chapters 5 and 6 present novel methods for digital hardware resource reductions in the implementation of SNNs for on-edge inference in image pattern recognition problems and corresponding hardware implementations.

Chapter 2 presents mathematical modifications to the Izhikevich neuron model [7] that simplify the digital hardware requirements for implementation.

Chapter 3 presents an implementation of the Hodgkin-Huxley neuron model [4], one of the most biologically meaningful neuron models, using the Co-Ordinate Rotational DIgital Computer (CORDIC) algorithm to reduce the complexity of a digital hardware implementation.

Chapter 4 presents a sampling-based hardware implementation methodology by which the power consumption of a biologically detailed digital neuron can be reduced.

Chapter 5 presents the concept of input sparsity applied to SNNs and its significance in reducing the hardware requirements of digital on-edge inference networks. The novel Selective Input Sparsity (SIS) method is introduced and used to implement a digital SNN.

Chapter 6 presents Field-Programmable Gate Array (FPGA) digital hardware implementations of the networks described in Chapter 5.

Chapter 7 summarizes and broadly discusses the work to form a conclusion and suggest possible future extensions and continuations.

Chapter 2

An Efficient Spiking Neuron Hardware System Based on the Hardware-Oriented Modified Izhikevich Neuron (HOMIN) Model

The basic element of a neural network is a neuron, which uses voltage spikes to transmit information, and the information is transmitted primarily in the timing of these spikes [3]. Biologically accurate neuron behavioural models, most notably the Izhikevich [7] and Adaptive-Exponential Integrate-and-Fire (AdEx) [8] neuron models successfully capture the natural spiking phenomena exhibited by real neurons. However, biological neurons and accurate mathematical models are neither time-invariant nor linear, making hardware realizations of these models challenging. Many hardware implementations of Spiking Neurons in analog [31–33], digital [34–38], and mixed signal [39, 40] systems have been proposed. However,

implementations with high accuracy to the mathematical neuron model are computationally expensive. Furthermore, digital hardware realizations of a biologically accurate neuron model in its presented form would be computationally intensive, making large networks of highly accurate neurons not feasible on a single Integrated Circuit die.

Thus, it is evident that compromises must be made between the accuracy of the neuron to biological behaviour and the computational requirements of the implemented hardware system to make the neuron a practical candidate for use in a spiking ANN. The model must have a low computational cost while simultaneously maintaining a sufficient level of accuracy and performance.

Spiking Neural Networks (SNNs) are more accurate to the function of biological neural networks than conventional ANNs, implying the potential applications of SNNs is far more diverse as they show potential not only in pattern recognition but also in biomedical applications [41].Therefore, exploration into viable hardware for improved SNNs is significant.

Many previous digital hardware implementations of spiking neurons are able to successfully replicate complex neuron behaviours [35,36]. However, to achieve their performance, they have very high resource requirements, and few optimizations are performed to lower hardware costs. Additionally, many system parameters must be externally set to change the neuron's behaviours. All these drawbacks are restrictive of the size of a potential network of such neurons.

Furthermore, biological neurons use rate coding and temporal coding, methods by which information is transmitted in the neuron's firing rate and the timing of the firings [1, 3, 41]. This implies that the firing rate and timing are of higher significance than the shape of the neuron's membrane voltage in the time domain.

To fulfill the required compromises, a variation of the Izhikevich Neuron Model [7] for a computationally low-cost digital hardware realization has been developed.

This Hardware-Oriented Modified Izhikevich Neuron (HOMIN) spiking neuron model modifies the original Izhikevich model, which results in an imperfect match to the Izhikevich model that still shows all cortical neuron behaviours, to greatly reduce hardware costs and required resources while simultaneously allowing for reduced projected interconnection requirements within a spiking neural network.

The formulation of the model is presented in detail, along with mathematical justification. Subsequently a hardware implementation of this new spiking neuron model is detailed, and comparisons are made.

2.1 Mathematical Preparations for Hardware Implementation

2.1.1 Formulation of the Hardware-Oriented Modified Izhikevich Neuron (HOMIN) Model

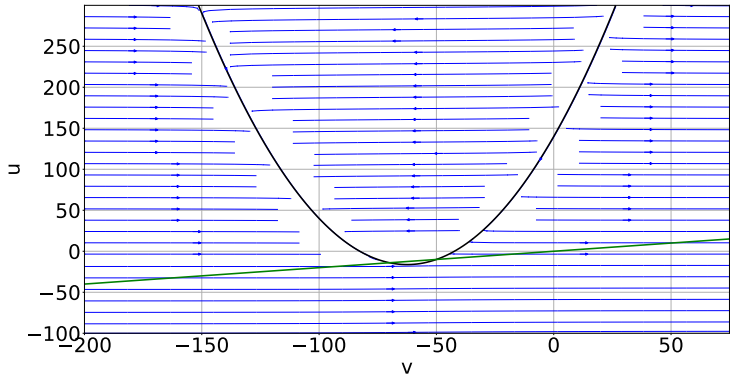
The original spiking neuron model proposed by Izhikevich in 2003 [7] is a mathematically simple approximation of biological reality that is able to reproduce a large variety of spiking patterns exhibited in real neural systems.

The model is a system of two interdependent differential equations with an auxiliary reset condition. The system is characterized by the following two equations and reset condition:

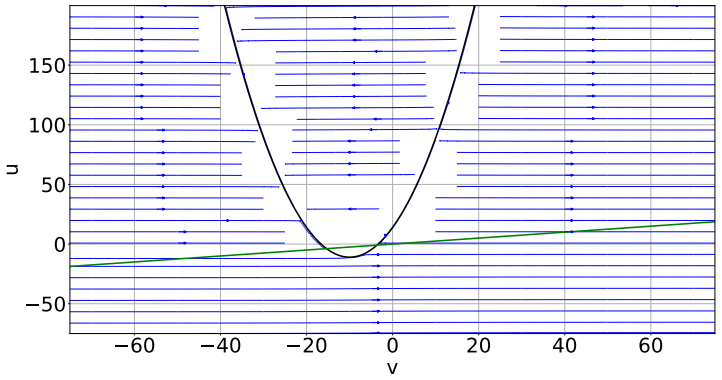
$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{2.1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2.2}$$

$$if v \geq 30mV, then \begin{cases} v \rightarrow c \\ u \rightarrow u + d \end{cases} \tag{2.3}$$



(A) Izhikevich Model for I=0 and regular spiking behaviour (a=0.02, b=0.2, c=-65, d=8).



(B) HOMIN Model for I=0 and regular spiking behaviour (d=6).

FIGURE 2.1: Phase portraits for the original Izhikevich model and the HOMIN model for zero input current. In both phase portraits, the black line shows the nullcline of $\frac{dv}{dt}$, the green line shows the nullcline of $\frac{du}{dt}$, and trajectories are shown in blue.

where v is the membrane potential of the neuron, u is the membrane recovery variable, and I is the input current to the neuron. The parameters a, b, c, and d describe the recovery variable time scale, sensitivity of the recovery variable, after-spike reset value of the membrane potential, and the after-spike reset of the recovery variable respectively.

Figure 2.1a shows the phase of the original Izhikevich Model. The neuron approaches firing when the value of u is below the curve defined by the nullcline of Equation 2.1 and to the right of the value of the membrane potential c.

To achieve different spiking behaviours, a and b experience the smallest amount of change in the Izhikevich model. Parameter c is varied to adjust the position of the reset potential on the phase portrait relative to the nullcline of Equation 2.1 and d is varied to effect the behaviour of the system when a spike and subsequent reset occurs. For spiking patterns in which spikes must occur in quick secession, c is positioned at a steeper section of the nullcline of Equation 2.1.

The proposed HOMIN model simplifies the modifications necessary to generate varying spiking patterns by holding parameters a, b, and c constant and using a relatively less steep nullcline of $\frac{dv}{dt}$ for the system to increase the influence of parameter d on the type of spiking behaviour exhibited by the system. Since parameter d directly affects the reset position of the system, a less steep nullcline implies that parameter d becomes far more influential to the system's post-spike behaviour. Thus, it was found that if parameter d is set sufficiently, it can solely determine the post-spike system state such that it can determine if the system returns to a position of trajectory tending toward an eminent spike or a delayed spike. Consequently, the HOMIN model can exhibit all excitatory neuron behaviours while only varying parameter d to adjust the behaviour of the system at the time of an after-spike reset. The relative steepness of the $\frac{dv}{dt}$ nullcline was decreased by lowering the value of the coefficient of the v^2 term. Since the HOMIN model is formulated with the goal of a simple digital hardware implementation, the scaled value was selected as a power of 2, meaning it will translate into a simple arithmetic shift in fixed point operations. This reduces multiplication operations in a digital hardware implementation, making the system significantly faster and less resource intensive. Figure 2.1b shows the phase portrait of the HOMIN model. Since it is only parameter d that changes in the HOMIN model, it is noted that the phase plane of the HOMIN model, unlike the Izhikevich model, does not change depending on the firing mode since parameter d only alters the post-spike reset position of the system. It is evident from Figure 2.1 that the HOMIN model exhibits a much steeper $\frac{dv}{dt}$ nullcline than the original Izhikevich Model.

To reduce the required representation range in digital hardware, the HOMIN model uses a scaled range for the operation of the system. All variables in the HOMIN model are scaled down by a factor of 10 compared to the Izhikevich model. Since v^2, for a scaled v, will cause reduction by a factor of 100, the coefficient of this term must be scaled up by a factor of 10 to compensate, causing this to change from 0.04 to 0.4 in the original Izhikevich model. This coefficient was then modified to $0.25 = 2^{-2}$, which causes a sufficient decline in steepness in the system's nullcline.

Furthermore, parameters a and b were approximated by fixed powers of 2 such that they result in shifts in digital hardware and do not require loading circuitry. After all modifications, the HOMIN model can be described by:

$$\frac{dv}{dt} = (2^{-2})v^2 + (2^2)v + v + 14 - u + I \tag{2.4}$$

$$\frac{du}{dt} = (2^{-6})((2^{-2})v - u) \tag{2.5}$$

$$if\, v \geq 3mV, then \begin{cases} v \rightarrow c \\ u \rightarrow u + d \end{cases} \tag{2.6}$$

where any multiplication by a power of 2 translates into a simple arithmetic shift in a digital hardware implementation. Not only does this model greatly reduce the hardware requirements by reducing the number of multiplications, it also reduces the required loading circuitry since parameters a, b, and c are system constants instead of variable parameters as in the original Izhikevich model.

2.1.2 Discretization of the HOMIN Model

To be implemented into digital hardware, the HOMIN model was discretized using Euler's method and the time step dt was selected as $dt = 0.03125 = 2^{-5}$ for convenience in a digital hardware implementation. The final form of the discretized

HOMIN model is:

$$v[n+1] = v[n] + 2^{-5}(2^{-2}v^2 + 2^2 v[n] + v[n] + 14 - u[n] + I[n]) \tag{2.7}$$

$$u[n+1] = u[n] + 2^{-11}(2^{-2}v[n] - u[n]) \tag{2.8}$$

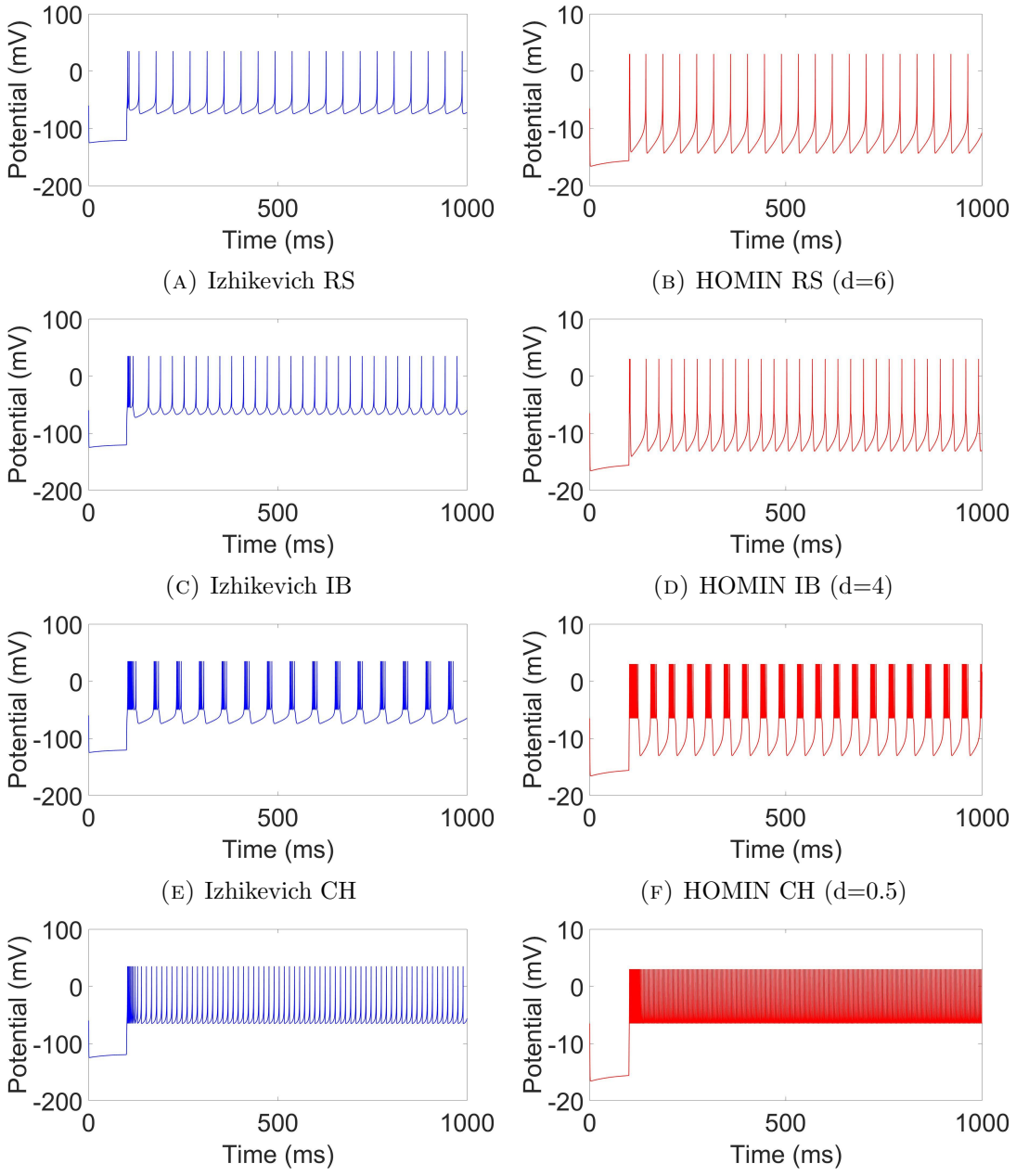
Figure 2.2 shows MATLAB simulations of voltage traces of the HOMIN model when varying only parameter d for a constant applied input current.

Figure 2.3 shows raster plots for both the Izhikevich Neuron and the HOMIN neuron for 1000 randomly connected excitatory neurons. It is noted from this figure that the HOMIN model exhibits a higher firing activity level at the time points in which the firing is clustered as well as between clusters. This can be attributed to a higher input sensitivity for the HOMIN model. Additionally, there is a small error in the timing of the spiking clusters between the two models. However, in a network both discrepancies could be remedied by appropriate synaptic weights.

From Figure 2.2, it is evident that the four spiking phenomena shown can be produced exclusively through variations of the value of d within a resolution that can be achieved using an 6-bit unsigned digital fixed-point value in which there are 3 integer bits and 3 fractional bits. This implies that, in a network of HOMIN neurons, the behavioural characteristics of each neuron can be controlled using only a 6-bit parameter, which is a major reduction in interconnection requirements in comparison to the original Izhikevich neuron's interconnection requirements to achieve the same spiking patterns.

Since information is encoded in the timing of the spikes in a spiking neural network [2], timing comparison was performed. Following the error analysis performed in [34], the Mean Relative Error (MRE) was used to analyse the timing performance of the HOMIN model. MRE is defined as:

$$MRE\% = \frac{\sum_{i=1}^{n} \frac{t_{HOMIN} - t_{Izh}}{t_{Izh}}}{n} 100\% \tag{2.9}$$



(A) Izhikevich RS

(B) HOMIN RS (d=6)

(C) Izhikevich IB

(D) HOMIN IB (d=4)

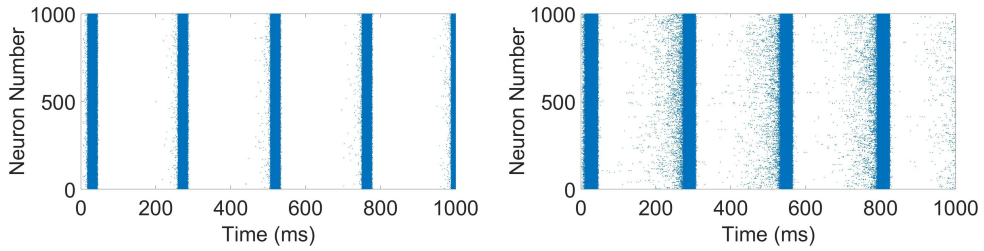
(E) Izhikevich CH

(F) HOMIN CH (d=0.5)

(G) Izhikevich LTS

(H) HOMIN LTS (d=0.375)

FIGURE 2.2: MATLAB simulations of the membrane potential as a function of time. (A), (C), (E), (G) show the Izhikevich model for a constant input current of I=150 and (B), (D), (F), (H) show the HOMIN model for a corresponding scaled constant input of I=15. Input was applied at t = 0.1ms. The values of the parameter d that correspond to the spiking patterns for the HOMIN model are labelled. Regular Spiking (RS), Initial Bursting (IB), Chattering (CH), and Low-Threshold Spiking (LTS) behaviours are shown.



(A) (B)

FIGURE 2.3: Raster Plots of 1000 randomly connected excitatory neurons for (a) an Izhikevich Neuron and (b) The HOMIN Neuron

TABLE 2.1: Error Measures of the HOMIN Model for MATLAB simulations conducted at I =15

Spiking Behaviour	**MRE (%)**
Regular Spiking	2.087
Initial Bursting	2.192

Table 2.1 shows these error quantities. It is noted that an MRE for tonic spiking of 4.09% was reported in [36] and 1.21% in [34] , implying that the error observed in the HOMIN model compared to the Izhikevich model is sufficiently small given the presented hardware resource savings.

Figure 2.4 shows a MATLAB simulation of a network of three HOMIN neurons in which the output membrane potential of Neurons 1 and 2 are used to generate the input current for Neuron 3. Initially Neuron 1 causes Neuron 3 to spike, but Neuron 2 does not. After a period of associative, unsupervised learning where Neurons 1 and 2 are stimulated together, Neuron 2 is able to cause Neuron 3 to spike after the learning period. This serves as a simple example to show that the HOMIN model is a viable candidate for unsupervised learning in a network.

2.2 FPGA Implementation

The HOMIN neuron model was realized in digital hardware. Verilog HDL was used to implement the design on a simple Altera DE0 Cyclone III FPGA board.

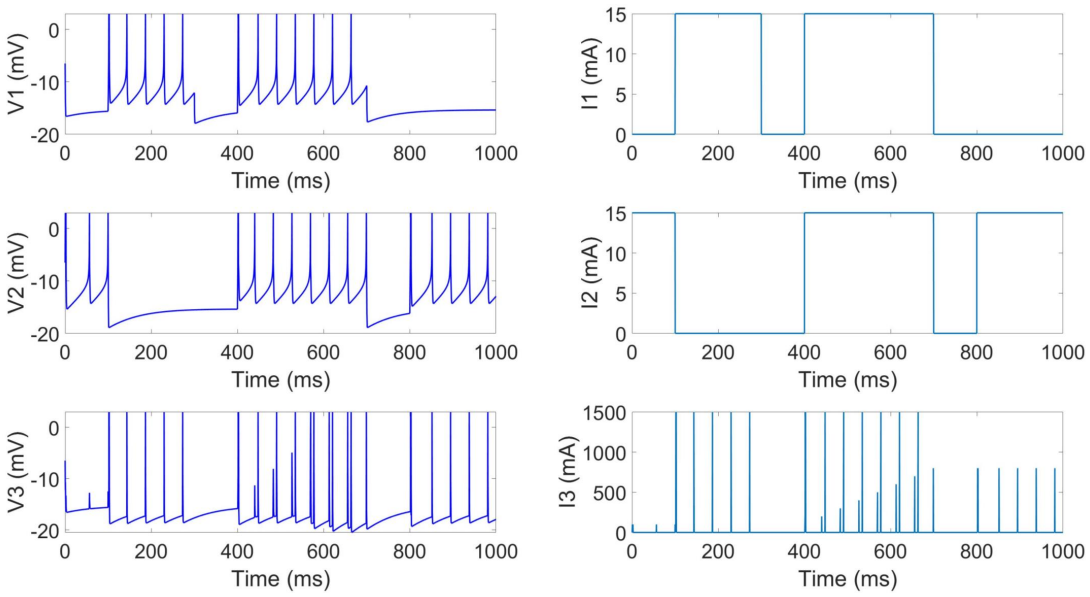
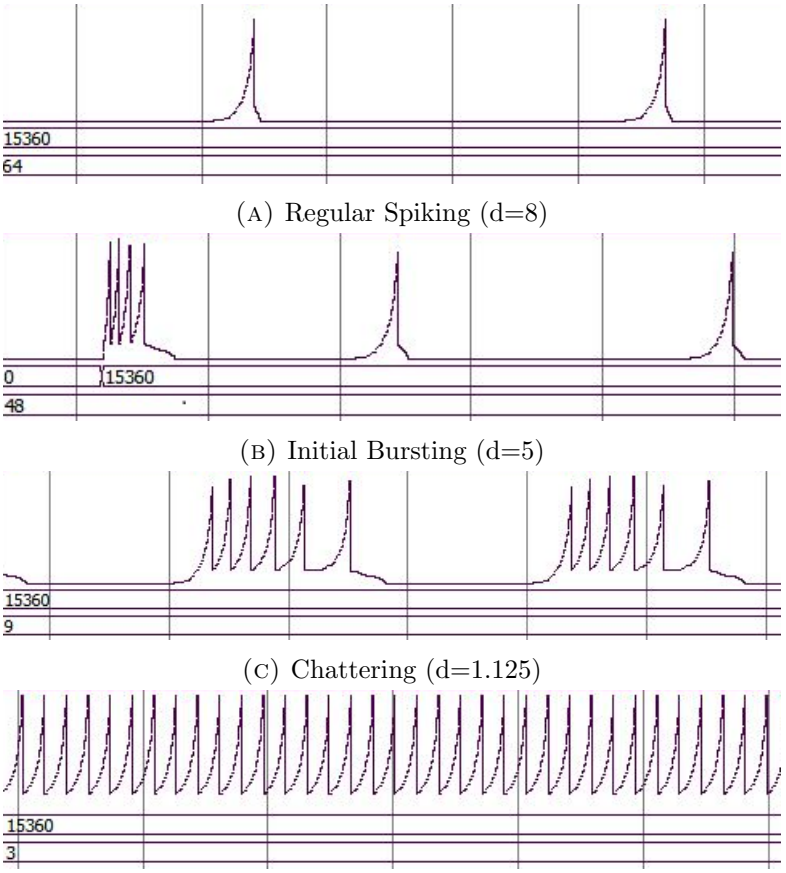


FIGURE 2.4: A simulation of three HOMIN Neurons showing the membrane potential and corresponding input current for each neuron. After the associative learning period, spikes from Neuron 2 are able to cause spikes from Neuron 3.

A fixed point implementation was selected with a 16-bit data path of which 1 bit is a sign bit, 6 bits represent the integer part, and 9 bits represent the fractional part. The bits were selected to best represent the relevant information for the spiking behaviour.

Since no relevant information from the neuron's behaviour is present in the low negative values of the membrane voltage, the membrane state variable was clamped to stay above -8 to prevent overflow in the multiplication involved in generating the v^2 term. With the clamp in place, the multiplication was designed such that the upper bits of the product are not evaluated, and the lower bits are approximated using a rounding constant, resulting in an approximation of the product with a far lower resource usage and no full multiplier.

The parameter d was taken as an 8-bit input value to the system where 5 bits are integer part, and 3 bits are fractional part. Although MATLAB simulations showed that the value of d could be represented within the range of a 6-bit fixed point number, an increased system sensitivity to lower values in parameter d was



(A) Regular Spiking (d=8)

(B) Initial Bursting (d=5)

(C) Chattering (d=1.125)

(D) Low-Threshold Spiking (d=0.375)

FIGURE 2.5: Quartus ModelSim simulations of the implemented digital hardware system.

anticipated in the selected fixed point implementation, thus it was made wider to compensate.

Figure 2.5 shows simulations of the digital hardware system using Quartus ModelSim. Each of the four behaviours shown in MATLAB simulations were successfully recreated in the digital hardware system.

Following successful simulations of the design, the hardware was synthesized on the FPGA board. Figure 2.6 shows images of output captured on an oscilloscope. The digital output was viewed in analog form with 4-bit resolution on the oscilloscope by converting 4-bits of the membrane potential to analog using the RGB connector of the DE0 board. Before conversion, the membrane potential was translated to a

TABLE 2.2: FPGA Resource Usage Comparisons Between the Implemented HOMIN Neuron and Previously Proposed FPGA Hardware Systems Based on the Izhikevich Neuron Model

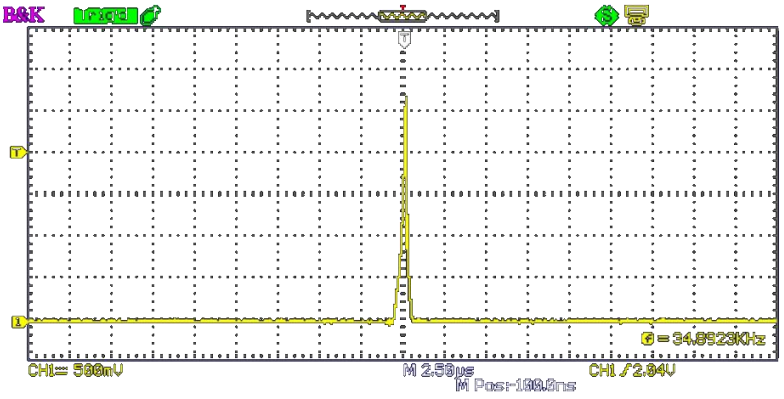
Resource	**HOMIN**	**MNIN** [34]	**Izhikevich** [34]
Logic Elements	356	490	857
Flip Flops	41	408	551
4-input LUTs	346	459	1268
I/O Pins	26	34	34
8*18 MULT	0	0	1

positive value domain. Any observed noise in the signal is clearly attributable to the low-resolution conversion to analog. Again, the desired spiking patterns were successfully recreated.

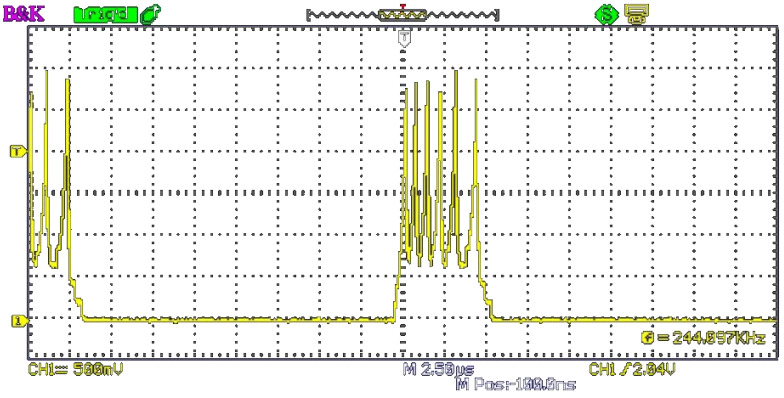
It was expected and observed that the implemented hardware system was less sensitive to input stimulus than the simulated MATLAB model. Additionally, it was found that the values of parameter d to achieve each of the spiking patterns were different in the hardware implementation as expected. This is due to the use of fixed point arithmetic to implement the hardware system.

2.2.1 Hardware Resource Usage

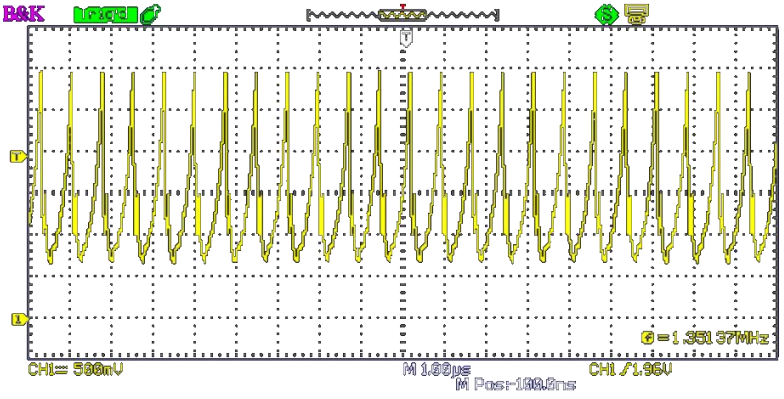
As anticipated, the FPGA implementation of the HOMIN model required far fewer digital hardware resources than previously proposed hardware based on the Izhikevich neuron model. Table 2.2 shows comparisons with previously reported Izhikevich neuron model implementations. The information is taken from [34], where the reported hardware usage is based on the XILINX Vertex II Pro FPGA board. To form a reasonable comparison, a reported slice in [34] was considered equivalent to a logic element in the Altera DE0 board used to implement the HOMIN model. Additionally, a Slice Flip Flop on the XILINX Vertex II Pro was considered equivalent to a flip flop. All of the metrics are shown in Table 2.2.



(A) Regular Spiking (d=6)



(B) Chattering (d=0.75)



(C) Low-Threshold Spiking (d=0.125)

FIGURE 2.6: Oscilloscope window captures of the implemented neuron for different spiking patterns for varying values of parameter d at a constant input current of $I = 28mA$.

As is evident from Table 2.2, the HOMIN model has substantially lower digital hardware resource requirements than the original Izhikevich model and other previously proposed Izhikevich Neuron approximations. Additionally, since this implementation simply serves to prove the advantages of the HOMIN model, the implementation does not include substantial hardware optimization.

2.3 Conclusion

In conclusion, modifications to the Izhikevich spiking neuron model [7] were made to create the proposed HOMIN spiking neuron model which results in a simpler digital hardware implementation while simultaneously reducing the number of required interconnections for use in an artificial neural network since the spiking behaviour of an implemented HOMIN neuron can be modified with a single parameter in contrast to the four parameter changes required by the original Izhikevich neuron. The hardware savings and connection savings allow for a larger and far more parallel neural network implementation on an FPGA as few hardware resources are used per neuron and for parameter connections. A clear extension from the proposed neuron model and implementation is the realization of a network of digital spiking neurons.

Chapter 3

A Resource-Efficient and High-Accuracy CORDIC-Based Digital Implementation of the Hodgkin-Huxley Neuron

3.1 Introduction

Real biological neural systems are among the most intricate systems that naturally exist in humans. Biological neurons exhibit many diverse spiking behaviours, through which information is transmitted in a neural system. While traditional Artificial Neural Networks (ANNs) are bio-inspired systems that use a time-static methodology for computation, Spiking Neural Networks (SNNs) more closely emulate real biological systems through dynamic time behaviour. Given their closer replication of biological behaviour, SNNs, like traditional ANNs, have shown success in image classification and speech recognition [42] and also have further potential applications such as neuronal function and disease modelling [41]. Thus,

they are a topic of great research interest.

Neuromorphic engineering is a highly active field of research which includes the implementation of spiking neurons in electrical hardware with the goal of amalgamation into a spiking neural network. Since basic unit of a biological neural system is the neuron, an accurate description of the behaviour of a neuron is of great importance. Mathematical characterizations of these spiking behaviours have been proposed in many models [4, 5, 7–10], with each proposed characterization presenting advantages and drawbacks. Some models, such as the Hodgkin-Huxley Neuron Model [4] and the Morris-Lecar Model [5], are highly detailed and have true biological meaning [6] where all parameters used in the differential equation model directly represent a physical parameter observed in the neuron. Other neuron models offer a higher-level behaviour description and require fewer differential equations and model parameters, but still capture the essence of neuronal behaviour such as the Izhikevich Model, [7], the Adaptive-Exponential Integrate and Fire (AdEx) Model [8], and the Wilson Model [9], even though their parameters do not directly represent biological qualities. An even simpler description neuronal behaviour exists in the Leaky Integrate-and-Fire Neuron Model [10].

Given its relatively complex nature in comparison to the Leaky Integrate-and-Fire or Izhikevich Neuron Model due to its large array of differential equations and accompanying parameters, the Hodgkin-Huxley Neuron Model has seen limited proposed hardware implementations [20, 43–50]. However, the biological significance of the model due to its foundation on physical biological parameters makes it an excellent candidate for use in the creation of a neuromorphic system of high parallelism to a real biological neural system. Its faithful accuracy and correspondence to biological reality gives the model substantial potential for specific effectiveness in the previously mentioned application of modelling real neural systems. [20, 41] Specifically, these applications include modeling real neurological systems with the intent of expanded understanding of the progression of neurological disease [16]. Although software simulations can be done, custom hardware-based systems are

generally much faster that software [15], meaning that many different conditions and outcomes can be assessed quickly to expedite the collection of information. Conductance-based models, such as the Hodgkin-Huxley model, are well-suited to neural system modeling [51].

Field-Programmable Gate Array (FPGA) is an excellent medium for spiking neuron implementation due to its flexibility in comparison to Application-Specific Integrated Circuit (ASIC) digital designs. Furthermore, although analog designs typically present lower power and area consumption, they are typically less robust and more susceptible to noise and error than FPGA designs [22]. Additionally, FPGA implementations of neural networks present a major speed advantage compared to software implementations [15] at the expense of flexibility. For these reasons, the FPGA platform presents an excellent compromise between speed, performance, and flexibility [52–55].

Among the limited existing FPGA hardware proposals are several different methods. Firstly, relatively direct implementations of the Hodgkin-Huxley Neuron have been proposed [20]. The clear drawback to this approach is the exceedingly large hardware resource requirements, making large networks impractical. Other methods have proposed partial use of the COordinate Rotation DIgital Computer (CORDIC) Algorithm with Look-Up Tables (LUTs) to implement exponential terms [47]. However, aside from this improvement, the hardware requirements are still very high due to the remaining multiplication and division terms.

Moreover, some methods propose modifications to the original equations of the Hodgkin-Huxley Neuron's alpha and beta parameters [43, 44]. Although many of these proposals prove effective in reducing hardware resource requirements, error and input current limitations are often introduced compared to the original model. In [43], an accurate implementation of the Hodgkin-Huxley neuron is proposed with fixed parameters. However, the hardware resource utilization has been improved in a newer implementation [44] and the fixed parameters limit the

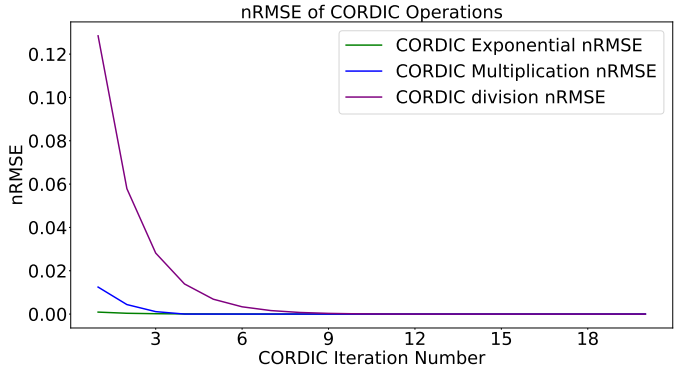
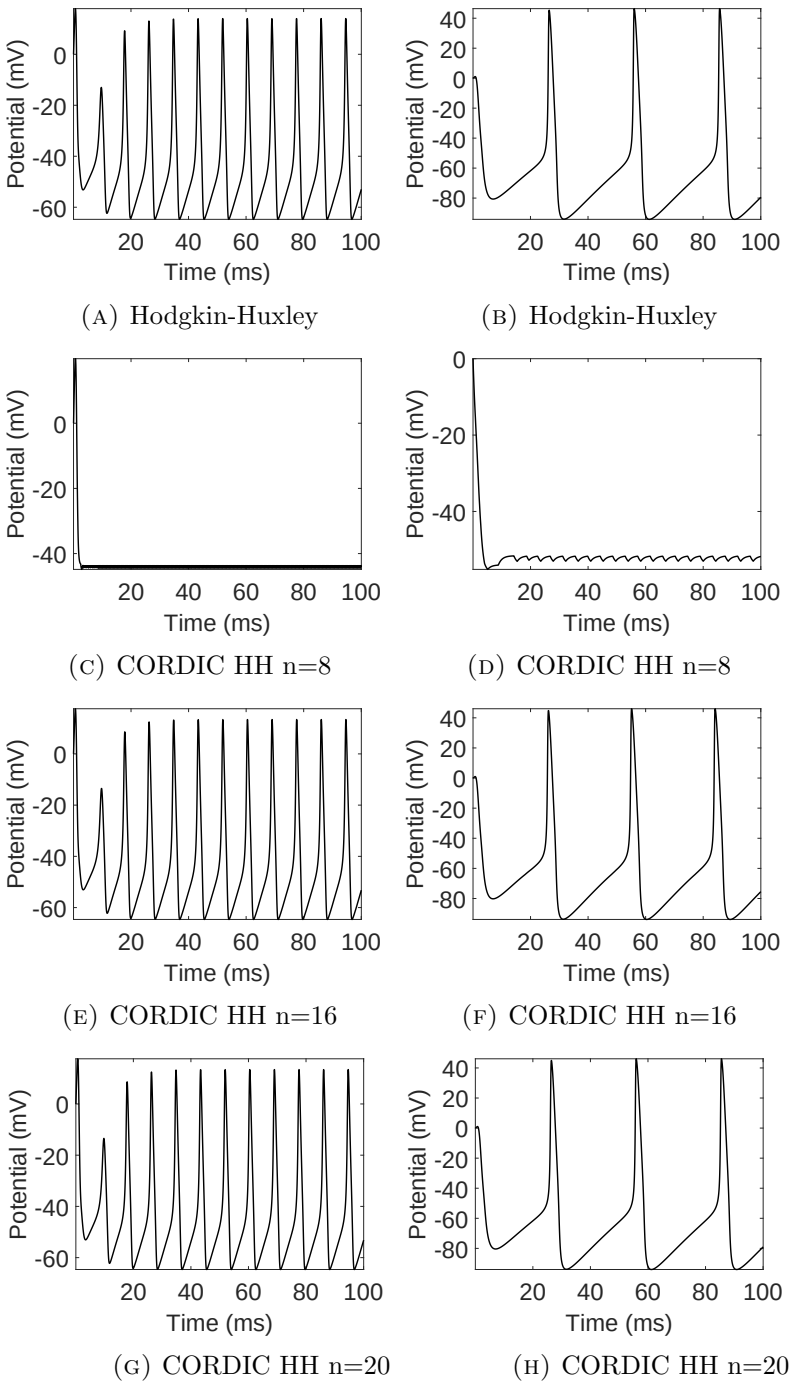


FIGURE 3.1: nRMSE of the CORDIC algorithm applied to multiplication, division, and the exponential function for 10000 trials at each CORDIC iteration number from 1 to 20. The input operands were normally distributed random numbers in the typical operand range for each operation as found in the Hodgkin-Huxley neuron model.

flexibility of the implementation in biomedical applications. Furthermore, in [44] an implementation with relatively low hardware requirements is proposed, again with fixed parameters but with an error accuracy. The elevated error and fixed parameters again may limit its efficacy in biomedical applications. It is important to emphasize again that all of the parameters of the Hodgkin-Huxley neuron have physiological significance and may change depending on the type of neuron cell of interest [56, 57].

An FPGA digital hardware implementation of the Hodgkin-Huxley Neuron Model is proposed using the COordinate Rotation DIgital Computer (CORDIC) Algorithm for all non-linear terms to substantially reduce the hardware resource requirements of the system while maintaining a high level of accuracy to the neuron model. Since the CORDIC Algorithm uses iterative shift and add operations, the hardware resources required to implement these terms are greatly reduced through use of this computation method. The proposed implementation shows substantial hardware resource savings compared to previously implementations.

Background information on the Hodgkin-Huxley Neuron Model and CORDIC Algorithm is presented, the iteration number determination is explained, an FPGA



(A) Hodgkin-Huxley (B) Hodgkin-Huxley

(C) CORDIC HH n=8 (D) CORDIC HH n=8

(E) CORDIC HH n=16 (F) CORDIC HH n=16

(G) CORDIC HH n=20 (H) CORDIC HH n=20

FIGURE 3.2: Software simulations of the membrane potential as a function of time of (A) and (B) the original Hodgkin-Huxley Neuron and (C), (E), (G) and (D), (F), (H) the CORDIC Hodgkin-Huxley Neuron for two different parameter and current operational points. (A), (C), (E), (G) show operation for Parameter Set 1 for a current of I = 500μA, while (B), (D), (F), (H) show operation for Parameter Set 2 for a current of I = 20μA. Between 16 and 20 iterations using the CORDIC algorithm, qualitative changes in the neuron's behaviour are not observed. Thus, an upper bound on the necessary iterations of the CORDIC algorithm exists.

hardware implementation is detailed, and the achieved results are compared with the theoretical expectations from the model behaviour and with previously proposed designs.

3.2 Background

3.2.1 The Hodgkin-Huxley Neuron Model

The Hodgkin-Huxley Neuron Model was proposed by Hodgkin and Huxley [4]. As the first conductance-based mathematical model of biological neuronal behaviour, it is one of the most significant neuron models in computational neuroscience because it has served as the basis for many conductance-based neuronal model descriptions that have followed [6]. The model consists of four differential equations, given as:

$$I = C_m \frac{dV}{dt} + g_K n^4 (V - V_K) + g_{Na} m^3 h (V - V_{Na}) + g_l (V - V_l) \tag{3.1}$$

$$\frac{dn}{dt} = \alpha_n (1 - n) - \beta_n n \tag{3.2}$$

$$\frac{dm}{dt} = \alpha_m (1 - m) - \beta_m m \tag{3.3}$$

$$\frac{dh}{dt} = \alpha_h (1 - h) - \beta_h h \tag{3.4}$$

where:

$$\alpha_n = \frac{0.01(V + 50)}{1 - e^{-(0.1V+5)}} \tag{3.5}$$

$$\alpha_m = \frac{0.1(V + 35)}{1 - e^{-(0.1V+3.5)}} \tag{3.6}$$

$$\alpha_h = 0.07 e^{\frac{-(V+60)}{20}} \tag{3.7}$$

$$\beta_n = 0.125 e^{\frac{-(V+60)}{80}} \tag{3.8}$$

$$\beta_m = 4e^{\frac{-(V+60)}{18}} \tag{3.9}$$

$$\beta_h = \frac{1}{e^{-(0.1V+3)} + 1} \tag{3.10}$$

where V is the membrane potential, I is the input synaptic current, and the remaining parameters are fixed for a given operation point and defined as:

C_m : membrane capacitance per area

V_{Na} : sodium equilibrium potential

V_K : potassium equilibrium potential

V_l : potential at which leakage current due to other ions is zero

g_{Na} : conductance to sodium

g_K : conductance to potassium

g_l : leakage conductance due to other ions

It is important to note that the above expressions for α and β in Equations 3.5 - 3.10 in this work are the same in mathematical structure as the original neuron proposed by Hodgkin Huxley in [4], but use different constants. The constants used come from adjustments proposed to the Hodgkin-Huxley neuron to explain type 3 excitability in squid giant axons [56]. Experimental data was used to derive the original constants, and the constants depend on the type of neuron being studied which makes this minor difference irrelevant [57], and methods for deriving these constants for a desired behaviour have been proposed [58].

Adjustments in the parameters will affect the spiking behaviour of the neuron and result in the generation of different neuronal behaviours. The Hodgkin-Huxley model gives a detailed description of a real neuron's behaviour since all parameters have biological meaning through their direct representation of a physical quantity in a neuron [6].

Given the large number of non-linear terms in the model, the error associated with approximations of these terms for hardware implementation can impair the neuron's behaviour quickly. Thus, maintaining a relatively high level of accuracy when computing these terms is highly important for preserving the biologically significant behaviour of the model. A high-accuracy computation method in digital hardware, such as the method described subsequently would be an excellent candidate for use in implementation.

3.2.2 COordinate Rotation DIgital Computer (CORDIC) Algorithm

The COordinate Rotation DIgital Computer (CORDIC) Algorithm [29] is a rotation-based, iterative algorithm used to simplify digital hardware requirements when implementing more complicated functions such as the cosine function or exponential function. The algorithm involves iterative rotation of the input by angles that result from the inverse tangent of a power of 2, meaning the rotation can be performed using a simple shift operation, where the shift direction is determine through comparison of the target rotation angle and the current cumulative rotation. Since the algorithm employs only shift and add operations, the CORDIC Algorithm can be used to implement many complex functions with relatively simple hardware, while the accuracy of the implementation is determined by the number of iterations performed. Thus, a compromise between computation speed and accuracy must be made.

A CORDIC implementation of the exponential function is given by Algorithm 1. Euler's number is e, x is the exponent of e, x_{frac} and x_{int} are the fractional and integer parts of x respectively, z is the result, i is the number of iterations, and k is the CORDIC iteration number. $a(i)$ is the ith value of $e^{2^{-i}}$ in radians.

TABLE 3.1: Hodgkin-Huxley neuron parameter sets used for testing

Parameter Set 1	$C_m = 1\mu\text{F}/cm^2$ V_{Na} = 55.12mV V_K = -72.14mV V_l = -49.42mV g_{Na} = 120mS g_K = 36mS g_l =0.3mS	Parameter Set 2	$C_m = 1\mu\text{F}/cm^2$ V_{Na} = 50mV V_K = -100mV V_l = -85mV g_{Na} = 50mS g_K = 5mS g_l = 0.1mS

Algorithm 1: The CORDIC exponential algorithm.

```
1  for i ← −k to −1 do
       // iterate n=k times
2      if x_frac(i) > 2^−i then
3          x_frac(i + 1) = x_frac(i) − 2^−i
4          z(i + 1) = z(i)a^i
5      end
6  end
7  while x_int(i) != 0 do
8      if x_int(i) > 0 then
9          z(i + 1) = z(i) ∗ e x_int(i) = x_int(i) - 1
10     else
11         z(i + 1) = z(i)/e x_int(i) = x_int(i) + 1
12     end
13 end
```

Multiplication is implemented using the CORDIC Algorithm according to Algorithm 2. x is the multiplicand, y is the multiplier, z is the product, i is the number of iterations, and k is the CORDIC iteration number.

Subsequently, division is performed using Algorithm 3. x is the dividend, y is the divisor, z is the quotient, i is the number of iterations, and k is the CORDIC iteration number.

Figure 3.1 shows the normalized Root-Mean Square Error (nRMSE) of the CORDIC algorithm when applied to multiplication, division, and the exponential function in the typical operational range of the Hodgkin-Huxley Neuron. It is apparent that as the number of iterations increases, the error converges to zero, making the

Algorithm 2: The CORDIC multiplication algorithm.

```
for i ← −l to k do
    // iterate n=2k+1 times
    if x(i) ≥ 0 then
        x(i + 1) = x(i) − 2^−i
        z(i + 1) = z(i) + 2^−i y
    else
        x(i + 1) = x(i) + 2^−i
        z(i + 1) = z(i) − 2^−i y
    end
end
```

Algorithm 3: The CORDIC division algorithm.

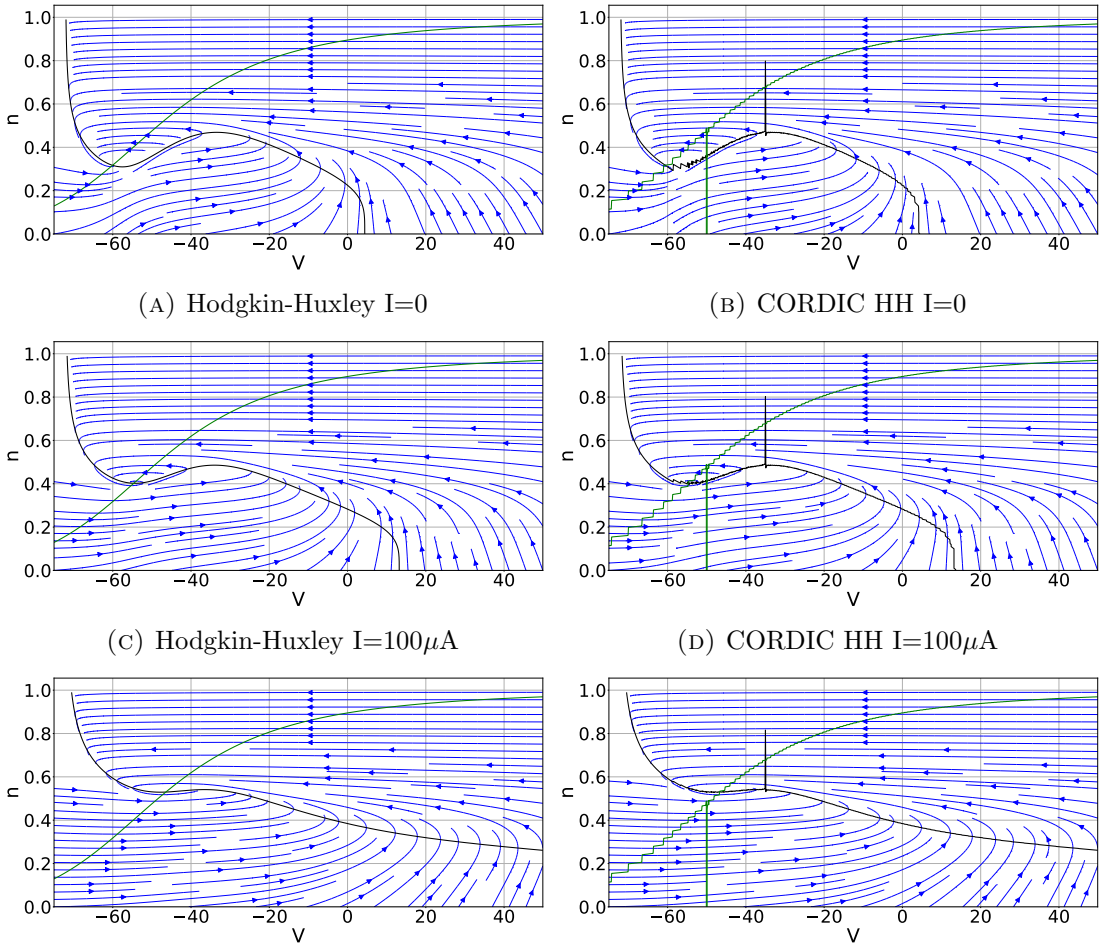
```
for i ← −k to k do
    // iterate n=2k+1 times
    if x(i) ≥ 0 then
        x(i + 1) = x(i) − 2^−i y
        z(i + 1) = z(i) + 2^−i
    else
        x(i + 1) = x(i) + 2^−i y
        z(i + 1) = z(i) − 2^−i
    end
end
```

CORDIC algorithm an excellent high-accuracy method for implementing these operations in digital hardware.

As will be presented in Section 3.4, the algorithms described in Algorithms 1 to 3 greatly reduce the hardware requirements for computing the non-linear terms of the Hodgkin-Huxley equations, resulting in smaller digital hardware.

3.3 Proposed Method

An implementation of the Hodgkin-Huxley neuron where the CORDIC algorithm is used to compute all nonlinear terms (multiplication, division, and exponential)



(A) Hodgkin-Huxley I=0

(B) CORDIC HH I=0

(C) Hodgkin-Huxley I=100μA

(D) CORDIC HH I=100μA

(E) Hodgkin-Huxley I=500μA

(F) CORDIC HH I=500μA

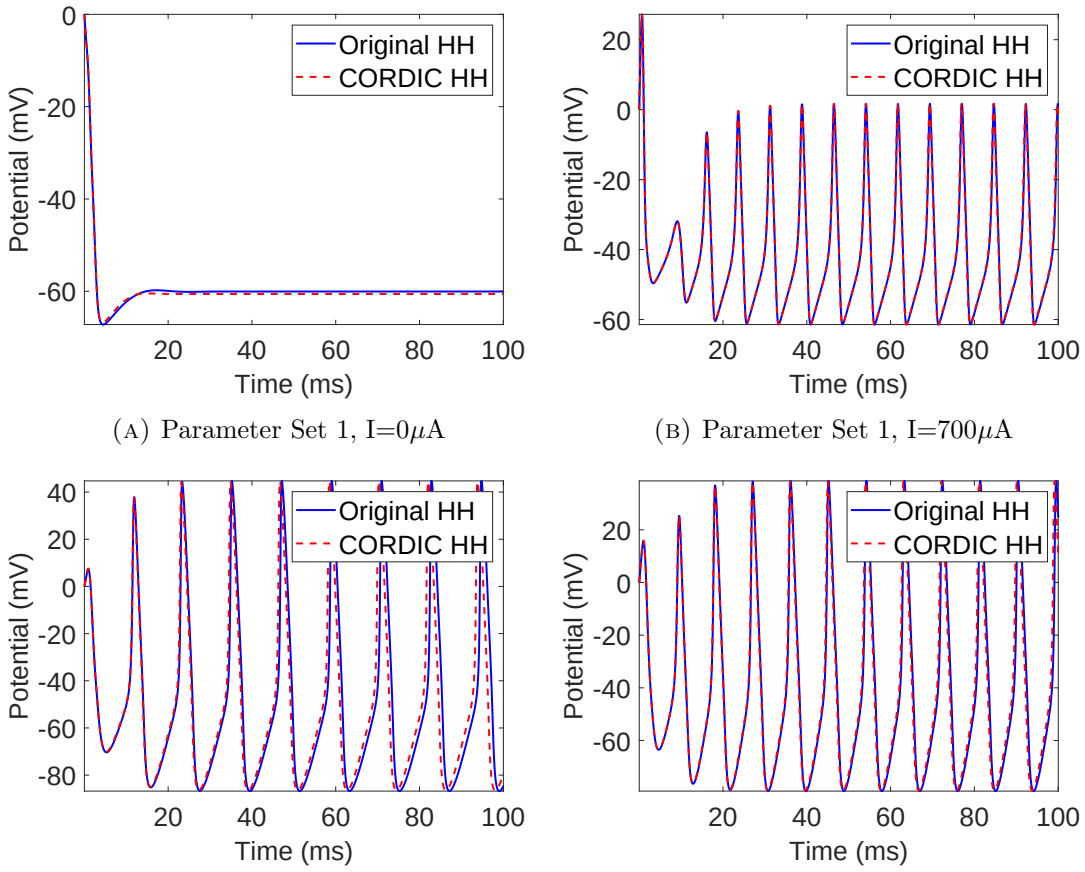
FIGURE 3.3: V-n phase portraits for (A), (C), (E) the original Hodgkin-Huxley neuron and (B), (D), (F) the CORDIC Hodgkin-Huxley neuron for varying currents where $\frac{dm}{dt} = 0$ and $\frac{dh}{dt} = 0$. V is defined in Equation 3.1 and n is defined in Equation 3.2. These phase portraits show that the proposed CORDIC Hodgkin-Huxley implementation show very similar bifurcations.

is proposed. Given the high accuracy of the CORDIC algorithm, this does not propose any approximation or modification of the neuron model, but rather a direct implementation that reduces the required digital hardware resources. Strategic scheduling and design (described subsequently) allowed for a reasonable maintenance of the system throughput, even though the speed of real biological systems is very low relative to the capabilities of digital electric hardware [3, 41].

3.3.1 Verification

To effectively utilize the CORDIC algorithm to reduce the hardware size of the Hodgkin-Huxley neuron, the effects of the use of the algorithm in the equations had to be thoroughly evaluated. MATLAB simulations were conducted for varying input currents and system parameters to compare the performance of the Hodgkin-Huxley neuron using the CORDIC Algorithm to the original neuron. Table 3.1 shows the parameter sets used for testing and verification. Parameter Set 1 can be found in [56], while Parameter Set 2 can be found in [46]. These two sets of parameters were used to confirm the accuracy of the CORDIC algorithm for implementation with different neuron operation conditions. Figure 3.2 shows simulations comparing the original Hodgkin-Huxley neuron to implementations with varying CORDIC iteration numbers. It is noted that the accuracy of the implementation relative to the original model improves with increasing iteration number as expected.

The objective of the simulations was firstly to determine numbers of iterations resulting in the lowest achieved error under all operating conditions, and secondly to determine the most feasible compromise between the iteration number and the accuracy of the output such that the throughput of the system is not severely afflicted. Based on simulation, it was determined that 10 iterations for exponential terms, 14 iterations for multiplication terms, and 8 iterations for division terms yields an error lower than previously proposed implementations [43,44] while maintaining a sufficient throughput. Note that these numbers refer to the value of k in Algorithms 1 to 3. The higher accuracy requirement for the multiplication operation compared to the others was expected as there are 17 multiplication operations, compared to 6 exponential operations and 3 divisions. This implies that the accuracy of the multiplication has the greatest influence on the accuracy of the implementation. Figure 3.4 shows the comparison of the original Hodgkin-Huxley Neuron to the CORDIC implementation for varying input currents, while Table 3.2



(A) Parameter Set 1, I=0μA

(B) Parameter Set 1, I=700μA

(C) Parameter Set 2, I=100μA

(D) Parameter Set 2, I=200μA

FIGURE 3.4: Software simulations of the membrane potential as a function of time for the original Hodgkin-Huxley Neuron and the CORDIC Hodgkin-Huxley Neuron for varying parameter sets and input currents. This shows the excellent qualitative matching of the CORDIC Hodgkin-Huxley neuron to the original model.

shows corresponding quantitative error. It is evident that the selected CORDIC implementation qualitatively follows the behaviour of the Hodgkin-Huxley neuron.

3.3.2 Error Analysis

Quantitative error analysis was performed to better characterize the accuracy of the proposed CORDIC Hodgkin-Huxley neuron to the original model. The Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Normalized Root Mean

TABLE 3.2: Error evaluation for select CORDIC Algorithm iteration numbers for exponential, multiplication, and division terms.

Current	**Parameter Set**	**Iteration Number**	**nRMSE (%)**	**MRE (%)**
500μA	1	8	26.705	98.433
		10	27.035	114.724
		12	0.860	0.096
		14	0.446	1.684
		16	0.226	0.862
		18	0.647	2.429
		20	0.033	0.121
50μA	2	8	27.046	45.548
		10	26.728	42.321
		12	1.530	0.608
		14	0.451	0.164
		16	0.289	0.180
		18	0.111	0.030
		20	0.213	0.116
700μA	1	8	28.158	104.685
		10	5.2011	2.939
		12	0.829	1.116
		16	0.640	1.530
		18	0.034	0.080
		20	0.004	0.010

Square Error (NRMSE) and Correlation are presented to characterize the matching of the original Hodgkin-Huxley neuron to the CORDIC-based model. These error quantities are used in multiple previously proposed Hodgkin-Huxley implementations [43, 44] and are described by the following:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|V_{CORDIC} - V_{HH}| \tag{3.11}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(V_{CORDIC} - V_{HH})^2}{n}} \tag{3.12}$$

$$NRMSE = \frac{RMSE}{\Delta V_{max} - \Delta V_{min}} \tag{3.13}$$

$$Correlation(V_{CORDIC}, V_{HH}) = \frac{cov(V_{CORDIC}, V_{HH})}{\sigma_{CORDIC} \cdot \sigma_{HH}} \tag{3.14}$$

TABLE 3.3: Error measurements for the proposed CORDIC implementation of the Hodgkin-Huxley neuron using Parameter Set 1 (PS1) and Parameter Set 2 (PS2). The low observed error implies that the proposed CORDIC implementation closely matches the original neuron model.

Measure of Error	PS1	PS2	**Average**
MAE	0.23	0.20	0.215
RMSE	0.28	0.23	0.255
NRMSE	0.40	0.20	0.30
Correlation	0.99	0.99	0.99

V_{CORDIC} and V_{HH} are the membrane potential values for the CORDIC implementation and the original Hodgkin-Huxley neuron respectively. The error is assessed by comparing the membrane potential for a single spike in the original and proposed neuron for dt = 10ms. The error quantities are presented in Table 3.3. Ten different input currents ranging from 0 to 2mA for each of the two parameter sets were used to evaluate the error. The value reported for each parameter set in Table 3.3 is the average of all ten trials for that parameter set. It is obvious that the proposed CORDIC implementation exhibits very close matching to the original model. This was expected given the exceptional accuracy of the CORDIC algorithm [29].

3.3.3 Bifurcation Analysis of the Hodgkin-Huxley Neuron and CORDIC Implementation

A bifurcation is a qualitative change in the phase portrait of a dynamical system, causing a change in the stability and behaviour of the system. Neurons exhibit bifurcations when switching between spiking and resting states [30]. Thus, the characterization of a neuron's bifurcations an important piece of information when analysing its behaviour, particularly for networks.

To characterize the bifurcations of the Hodgkin-Huxley Neuron, firstly the Jacobian Matrix **J** describing the interaction between the membrane potential V and

each of the auxiliary variables n, m, and h, must be defined as:

$$\mathbf{J}(V,x) = \begin{bmatrix} \frac{\delta}{\delta V}\frac{dV}{dt} & \frac{\delta}{\delta x}\frac{dV}{dt} \\ \frac{\delta}{\delta V}\frac{dx}{dt} & \frac{\delta}{\delta x}\frac{dx}{dt} \end{bmatrix} \tag{3.15}$$

where x represents n, m, or h as defined in Equations 3.2 - 3.4.

$$\mathbf{J}(V,n) = \begin{bmatrix} \frac{-1}{C}(g_K n^4 + g_{Na} m^3 h + g_l) & 4g_K n^3 (V - V_K) \\ (1-n)(\frac{\delta \alpha_n}{\delta V} V + \alpha_n) + n(\frac{\delta \beta_n}{\delta V} V - \beta_n) & -V(\alpha_n + \beta_n) \end{bmatrix} \tag{3.16}$$

$$\mathbf{J}(V,m) = \begin{bmatrix} \frac{-1}{C}(g_K n^4 + g_{Na} m^3 h + g_l) & 3g_{Na} m^2 h (V - V_{Na}) \\ (1-m)(\frac{\delta \alpha_m}{\delta V} V + \alpha_m) + m(\frac{\delta \beta_m}{\delta V} V - \beta_m) & -V(\alpha_m + \beta_m) \end{bmatrix} \tag{3.17}$$

$$\mathbf{J}(V,h) = \begin{bmatrix} \frac{-1}{C}(g_K n^4 + g_{Na} m^3 h + g_l) & g_{Na} m^3 (V - V_{Na}) \\ (1-h)(\frac{\delta \alpha_h}{\delta V} V + \alpha_h) + h(\frac{\delta \beta_h}{\delta V} V - \beta_h) & -V(\alpha_h + \beta_h) \end{bmatrix} \tag{3.18}$$

Therefore, the Jacobian Matrices that characterize the behaviour of the Hodgkin-Huxley Neuron are given by matrices in Equations 3.16 - 3.18. The partial derivatives of the α and β functions are given by:

$$\frac{\delta \alpha_n}{\delta V} = \frac{0.01 - (0.06 + 0.001V)e^{-0.1(V+50)})}{(1 - e^{-0.1(V+50)})^2} \tag{3.19}$$

$$\frac{\delta \alpha_m}{\delta V} = \frac{0.1 - (0.45 + 0.01V)e^{-0.1(V+35)})}{(1 - e^{-0.1(V+35)})^2} \tag{3.20}$$

$$\frac{\delta \alpha_h}{\delta V} = \frac{-0.07}{20} e^{\frac{-(V+60)}{20}} \tag{3.21}$$

$$\frac{\delta \beta_n}{\delta V} = \frac{0.125}{80} e^{\frac{-(V+60)}{80}} \tag{3.22}$$

$$\frac{\delta \beta_m}{\delta V} = \frac{4}{18} e^{\frac{-(V+60)}{18}} \tag{3.23}$$

TABLE 3.4: Fixed Points and corresponding eigenvalues for the Hodgkin-Huxley Neuron and the CORDIC HH Neuron for varying current using Parameter Set 1. The proximity of the fixed points between the CORDIC HH and the original Hodgkin-Huxley neuron implies close behavioural similarity.

	Hodgkin-Huxley	**CORDIC HH**
Current	**Fixed Point (V,n,m,h)**	**Fixed Point (V,n,m,h)**
0μ A	(-60.048,0.317,0.053,0.598)	(-60.665,0.318,0.049,0.627)
50μ A	(-56.782,0.368,0.077,0.481)	(-56.629,0.383,0.079,0.471)
100μ A	(-54.621,0.402,0.098,0.405)	(-54.775,0.412,0.096,0.421)
200μ A	(-51.643,0.450,0.134,0.309)	(-51.6,0.444,0.134,0.304)
400μ A	(-47.84,0.509,0.195,0.21)	(-47.800,0.514,0.196,0.208)
500μ A	(-46.443,0.53,0.221,0.180)	(-46.518,0.534,0.221,0.182)
700μ A	(-44.182,0.562,0.269,0.140)	(-44.5,0.556,0.261,0.144)

$$\frac{\delta\beta_h}{\delta V} = \frac{0.1e^{-0.1(V+30)}}{(1+e^{-0.1(V+30)})^2} \tag{3.24}$$

The fixed points of the system occur when the time derivatives of V, n, m, and h are all equal to zero. Again, V, n, m, and h are defined in Equations 3.1 - 3.4. The behaviour around these fixed points influences the bifurcation behaviour of the neuron. Thus, the position and behaviour around these fixed points is important for characterizing the neuron's behaviour.

A numerical solution was found for the fixed points at varying currents. The solutions are shown in Table 3.4. Figure 3.3 shows the V-n phase portraits for both the original neuron and the CORDIC implementation. The V-n phase portraits are shown for $\frac{dm}{dt} = 0$ and $\frac{dh}{dt} = 0$. From Table 3.4 it is evident that the CORDIC implementation has fixed points that closely follow the original Hodgkin-Huxley neuron. From Figure 3.3 it can be seen that the qualitative behaviour around the fixed points at given input current values is consistent between the two neurons. Therefore, the use of the CORDIC algorithm for implementing the Hodgkin-Huxley neuron does not impose a change in the behaviour of the neuron.

3.4 FPGA Hardware Implementation

The proposed CORDIC Hodgkin-Huxley neuron was implemented in digital hardware on the Xilinx Kintex-7 FPGA. The system was discretized using Euler's method, and based on the operational range of the neuron's input current and membrane potential, a 22-bit signed fixed point arithmetic was employed using 10 integer bits and 12 fractional bits. The bit width was selected to ensure that overflow and underflow do not occur during operation. The time step was selected as $dt = 2^{-5}$.

The input parameters of the Hodgkin-Huxley neuron are 16-bit signed fixed point values with different radix points for each parameter based on its value range. C_m is assumed to be input as its inverse, since this is the only form in which it is used in the neuron. $C_{m,inv}$, V_{Na}, V_k, and V_L have 8 integer bits and 8 fractional bits. g_{Na} has 3 integer bits and 13 fractional bits, while g_k and g_L have 1 integer bit and 15 fractional bits. Again, these radix points were selected based on the reported ranges of these parameters which were derived from experimental observations for the neuron model [4, 57].

Given the selected iteration numbers for the CORDIC algorithm implementations of the non-linear terms, the input operands are bit extended before a given operation to prevent overflow and underflow during shifts, and the result is returned to the 22-bit representation. The extension methodology is used to reduce the required number of LUTs and flip-flops in the FPGA implementation as data is stored with a lower word length while temporary registers are used for operations other than addition and subtraction.

The structure of the implemented CORDIC exponential, multiplication, and division units are shown in Figures 3.5 - 3.7. In Figure 3.5 the arctangent of 2^{-k} is implemented using multiplexed shift and add operations that are conditionally performed. 2^{-i} is generated for each iteration using a right shift register that is

initialized to 2^{-1} for the first iteration and shifted before each subsequent iteration. When k, the desired number of iterations is reached (in this case 10 iterations), $i(4)$, the fourth bit of register i will be 1, moving the unit to the third state in which conditional multiplication or division by e is performed based on the integer part of input x. These multiplications and divisions are performed using shift and add operations for a low hardware cost.

As can be observed in Figures 3.6 and 3.7, the structures of the CORDIC multiplication and division units are quite similar. The most notable difference is that the division unit requires correction for the signs of the operands. They are state machines with two states, where the first state performs addition or subtraction based on the sign bit of x (the multiplicand or the dividend) and updates the iteration count. In the second state, the iteration counter's most significant bit is checked. If it is 1, the process is complete and a return signal is generated to return the result of the operation, otherwise right shifts are performed and the unit returns to the first state.

Exponential terms are computed consecutively using a single CORDIC exponential unit. This was found to be the most effective scheduling as the CORDIC exponential unit represents the critical path of the system and is the largest CORDIC evaluation unit in terms of resource utilization. Therefore, the most resource-efficient scheduling was the use of a single CORDIC exponential unit. The CORDIC multiplication operations were processed in batches of 3 such that the resource usage is reasonable, and the throughput of the system is not adversely affected. Division operations were run sequentially. Other scheduling options were evaluated, including sequential multiplication and division operations. Although these options reduce hardware requirements, the reduction is not significant in comparison to the additional delay these options introduced since the CORDIC multiplication and division units as shown in Figures 3.6 and 3.7 are relatively simple. Figure 3.8 shows scheduling diagrams for the proposed hardware implementation summarizing the system's scheduling in sections. As can be seen, the successional

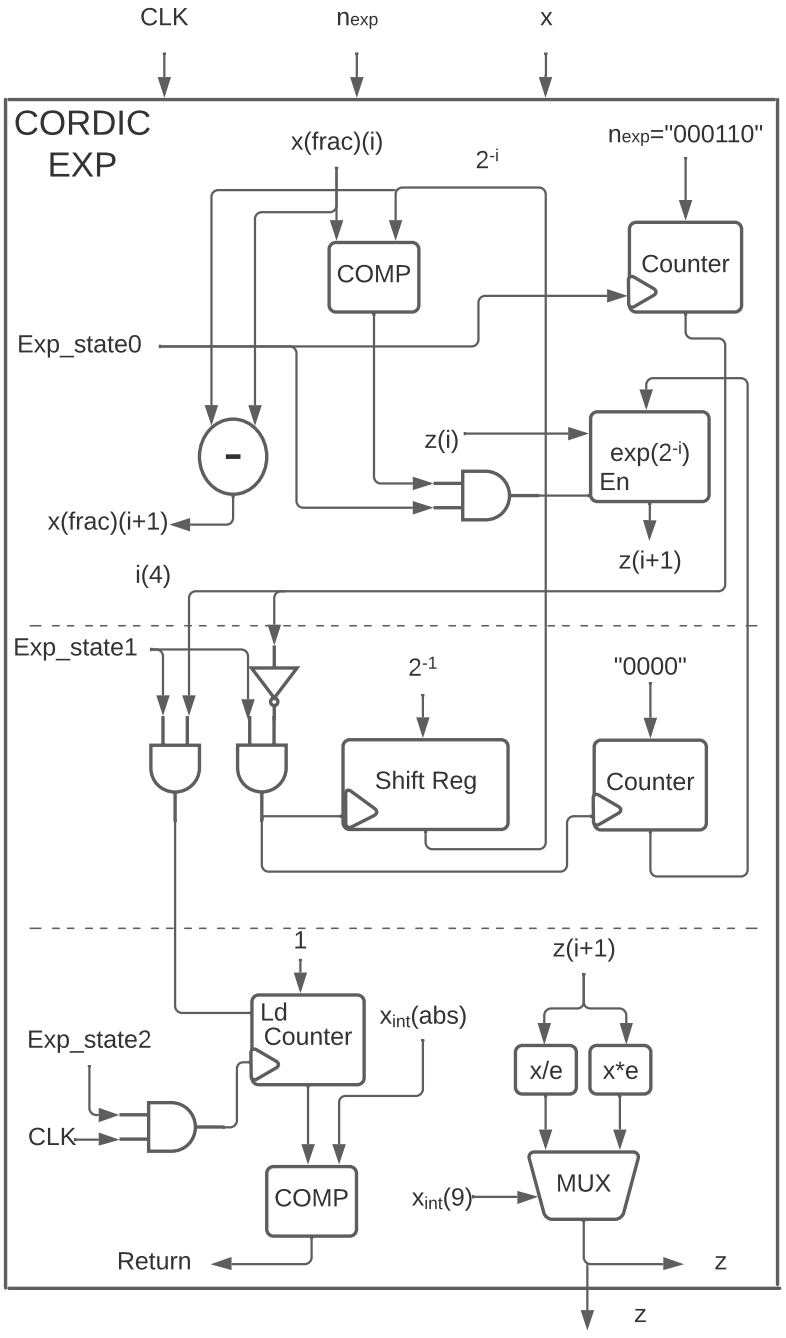


FIGURE 3.5: A block diagram of the CORDIC exponential unit. $e^{2^{-i}}$ is implemented using multiplexed shift and add operations. z is initially set to one and conditionally updated based on the control signal generated by the comparison of $x(frac)$ (the fractional part of x) and 2^{-i}.

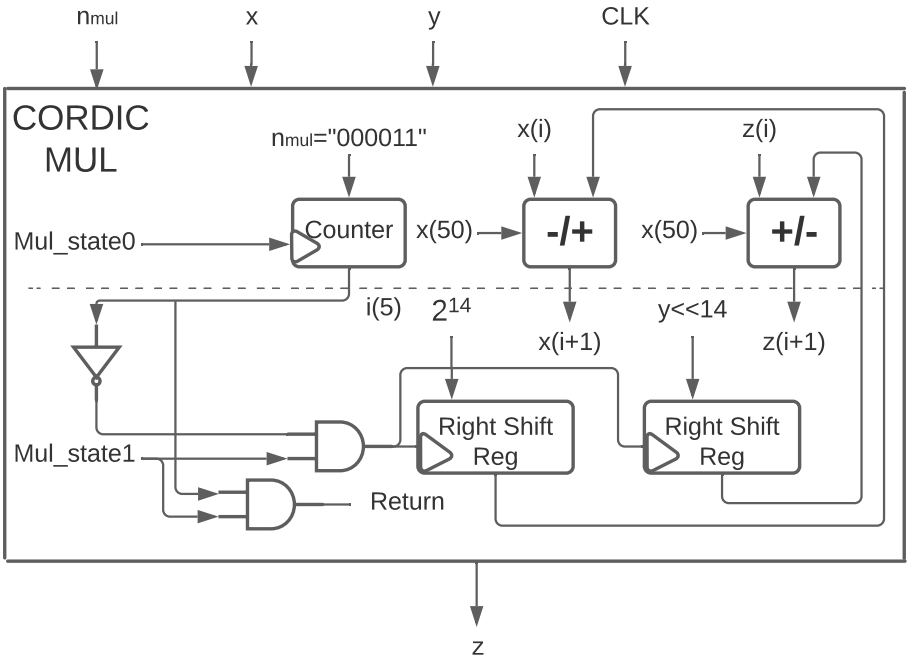


FIGURE 3.6: A block diagram of the CORDIC multiplication unit.

exponential and division operations are run concurrently with the multiplication operations such that the use of single exponential and division units does not increase the latency of the system. Handshaking protocols are used to coordinate the operations into distinct stages as shown in Figure 3.8.

The CORDIC multiplication and division algorithms require multiplication of the operand by 2^{-n} to 2^{n}, were n is the number of iterations. These multiplications by powers of 2 translate into right and left shift operations in digital hardware. Therefore, for 8 iterations in the division operation, the 22-bit input operands were extended to 39 bits by padding with 8 fractional bits and 9 integer bits for an additional sign bit to handle possible overflow for division operations. By the same logic, for CORDIC multiplication with 14 iterations the input operands were extended to 51 bits. The results were then reduced to 22-bit outputs for further use in the system.

Figure 3.9 shows oscilloscope images of the membrane potential of the implemented CORDIC Hodgkin-Huxley neuron. The membrane potential was converted into

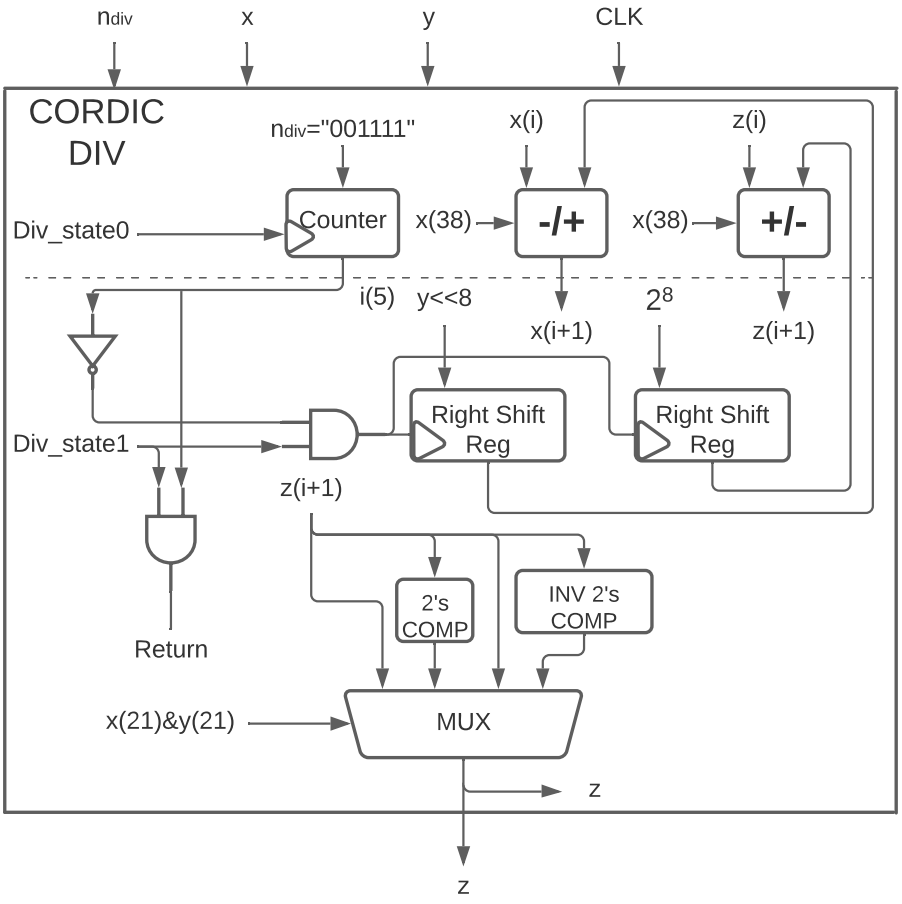


FIGURE 3.7: A block diagram of the CORDIC division unit.

an analog signal to view on the oscilloscope using a 12-bit resistive ladder Digital-to-Analog Converter (DAC), which explains any observed noise. To physically verify the design, the proposed design was realized on an FPGA to obtain the results shown in Figure 3.9.

3.5 Discussion

In [45, 46] the concept of neuron cores is used, where each core implements 512 neurons in real time. Although this is feasible for simulation, in reality each core is a single neuron with the membrane potential and state information of each neuron stored and sequentially loaded. The figure of 512 neurons refers to the

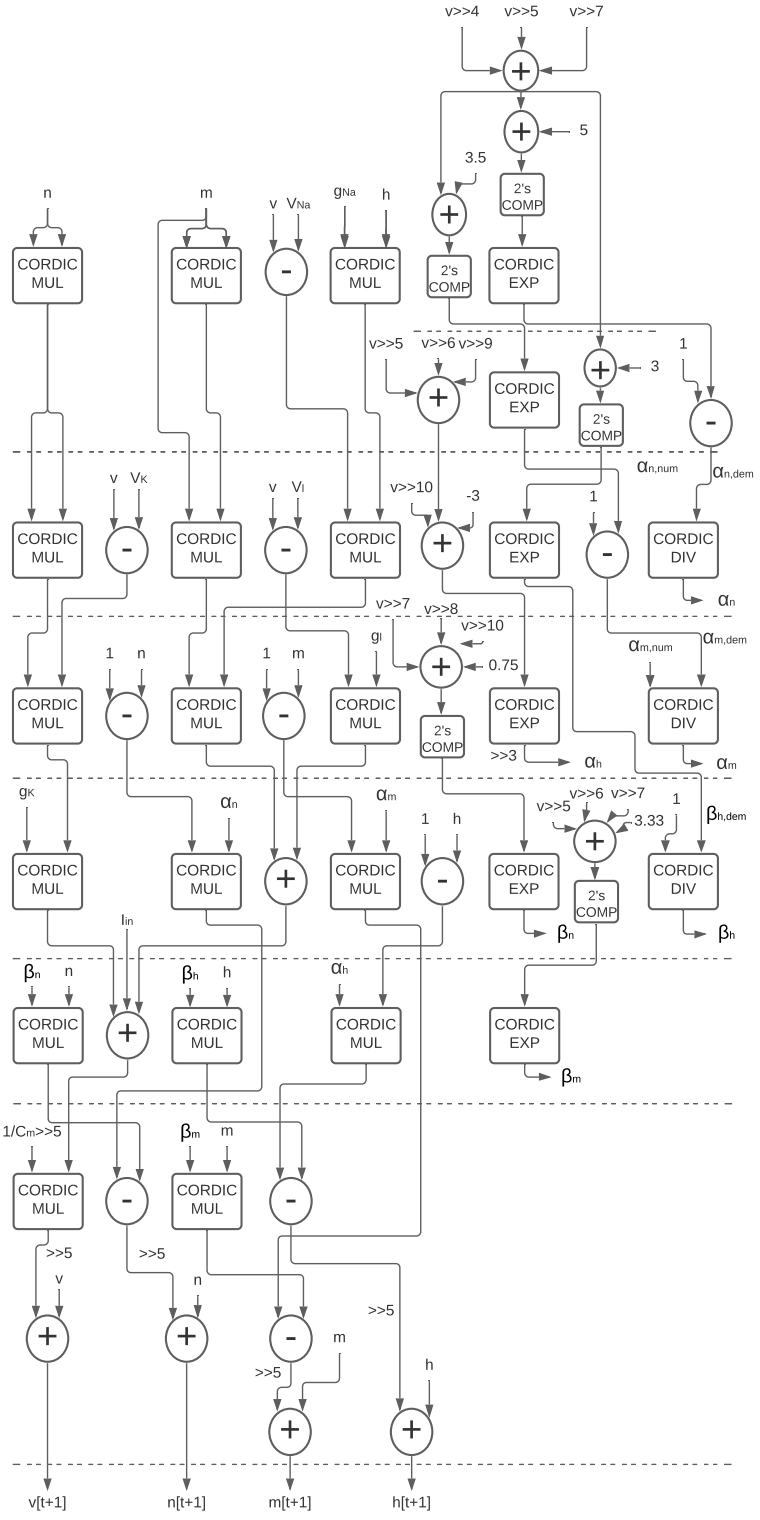
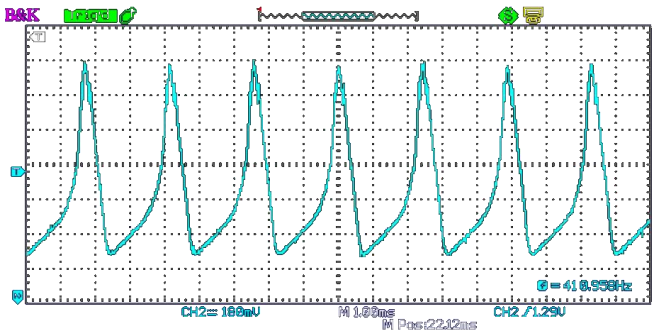
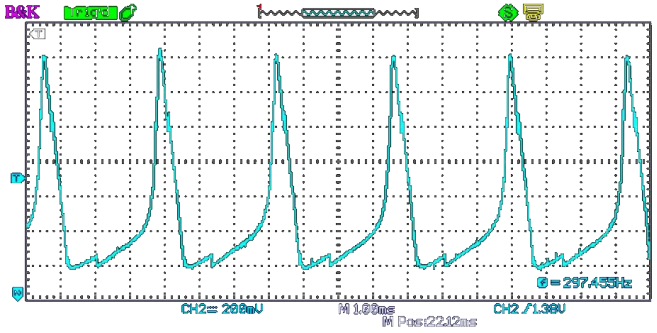


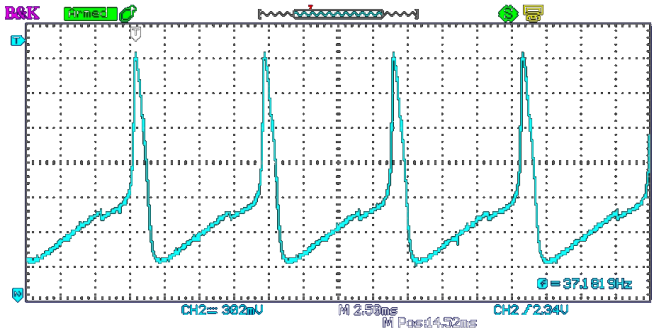
FIGURE 3.8: An overall scheduling diagram of the proposed neuron.



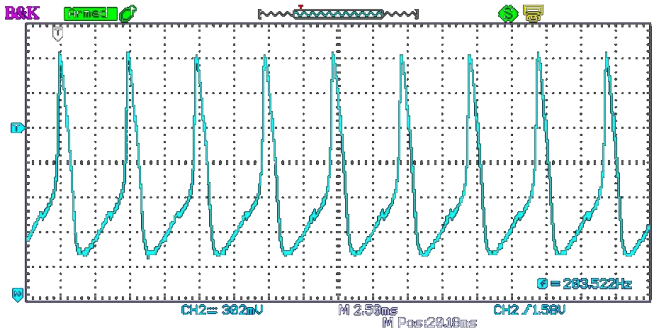
(A) Parameter Set 1, I=500μA



(B) Parameter Set 1, I=200μA



(C) Parameter Set 2, I=20μA



(D) Parameter Set 2, I=62.5μA

FIGURE 3.9: Oscilloscope images showing the membrane potential of the implemented neuron for different input synaptic currents and input parameters.

TABLE 3.5: Post implementation FPGA resource usage information for the proposed CORDIC Hodgkin Huxley Neuron and previously proposed FPGA implementations of the Hodgkin-Huxley Neuron. *Note: Modelling RMSE for [44], [43], and the proposed implementation was computed by evaluating the error in the membrane potential for the proposed implementation approximations and methods to the original Hodgkin-Huxley neuron model for various input currents. See the Error Analysis subsection for detailed information on the error analysis.

Resource	**CORDIC HH** Kintex-7	**Haghiri et al.** [44] Virtex-4	**CORDIC HH** Kintex-7	**Shama et al.** [43] Virtex-2
Flip Flops	1534	480	1534	2840
4-input LUTs	2380	2344	2380	5660
DSPs	0	0	0	0
Max Freq.	204.7MHz	200MHz	204.7MHz	85MHz
Model RMSE*	0.255	7.4	0.255	0.27
Resource	**CORDIC HH** Kintex-7	**Khoyratee et al.** [46] Kintex-7	**CORDIC HH** Artix-7	**Akbarzadeh et al.** [45] Artix-7
Flip Flops	1534	1552	1534	25430
4-input LUTs	2380	4735	2386	29130
DSPs	0	16	0	280
Max Freq.	202.5MHz	100MHz	163.64MHz	71.4MHz
Resource	**CORDIC HH** Spartan 7	**Bonabi et al.** [47] Spartan 3	**CORDIC HH** Zinq	**Yi et al.** [59] Zinq
Flip Flops	1534	7231	1534	7571
4-input LUTs	2388	23514	2394	8457
DSPs	0	99	0	45
Max Freq.	152.05MHz	37.563MHz	133.87MHz	-

ability of the system to compute at on biological time scale for this number of neurons. Additionally the model implemented in [46] is a simplification of the original Hodgkin-Huxley model and is therefore not an even comparison. Furthermore, [45–47, 59] all use DSP blocks, which if implemented using logic on the FPGA would add dramatically to the number of LUTs.

The proposed implementation in [43] is of very similar accuracy to the proposed CORDIC implementation. However, it requires far greater amounts of digital resources from the FPGA and operates at a lower clock frequency. Additionally, the proposed neuron in [43] does not allow for variable model parameter input. For applications such as disease modelling or the simulation of the effects of new medications, these input parameters are important as they directly model a physical characteristic of a real neuron. Some neurological diseases, such as Fabry Disease, are believed to include signature changes to ionic conductance in neurons [60].

Thus the proposed implementation in [43] may not be suitable to this task since the ionic conductance values are fixed, whereas they are externally changeable in the current proposed CORDIC implementation.

In [50], the authors implement the Hodgkin-Huxley neuron with floating point arithmetic and fixed physiological parameters. Although the authors show that the floating point arithmetic of their design has a better accuracy of a fixed point implementation, the design is substantially larger than fixed point implementations and fixed physiological parameters impose the same limitations on biomedical applications previously described. Additionally, to modify the floating point architecture to include the additional multiplication and division terms would substantially increase the size of the neuron.

Although the proposed design in [44] is similar in size to the proposed CORDIC implementation, it is important to note that the CORDIC implementation is distinct because it is far more accurate and accepts the model parameters as input, meaning they are variable rather than fixed. The potential significance of having variable physical neuron parameters in the design was described above for biomedical applications. This opens many more applications to the proposed CORDIC implementation in neuronal simulation acceleration, disease modelling, and artificial SNN implementations, among other applications since the parameters of the model depend on the type of neuron being modeled [56, 57]. Furthermore, in [44] the exponential functions in the Hodgkin-Huxley neuron are approximated by base 2 functions. Although this is an effective method for hardware simplification, it imposes a limitation on the input current range as the error in the approximation grows dramatically with increasing input. The CORDIC algorithm does not suffer from this problem [29]. Moreover, although the flip-flop usage is higher in the case of the proposed CORDIC implementation, the flip-flop utilization on the Kintex-7 board used for implementation is only 1.87% compared to 5.80% for Look-Up Table (LUT) utilization. This implies that flip-flop usage is not the limiting factor as to the number of neurons that is implementable on a single FPGA and thus

the number of flip-flops is less significant than the number of LUTs, which is very similar for both implementations.

3.6 Conclusion

A Hodgkin-Huxley neuron hardware system was designed and implemented on FPGA using the CORDIC algorithm for all nonlinear terms. The resulting system has low digital hardware resource requirements, close matching to the target neuron model, and accepts different neuron parameters as input. The flexibility of the physiological parameters of the model in the proposed design distinguishes it from other designs as a strong candidate for diverse biomedical applications.

Chapter 4

The Input-Dependent Variable Sampling (I-DEVS) Digital Neuron Implementation Method

4.1 Introduction

Parallelism with real biological systems is central to neuromorphic systems, and one of the most characteristic properties of real neural systems is their remarkably low power consumption in comparison to electrical hardware systems [61–63]. Given the large networks of parallel neurons found in real biological systems, low power consumption is paramount to an effective and feasible electrical hardware replication.

Many proposed spiking neuron and neuroprocessor implementations offer substantial power savings compared to implementations prior to their proposal [15, 31, 42, 64–72], however in each case limitations or restrictions are imposed on the implementation such as the use of a specific technology node, neuron model, and/or hardware implementation technique are required. Additionally, among the diverse

models of varying biological detail of spiking neuron behaviour [4, 5, 7, 8, 10], implementations concerned with power consumption often use the simplest and least biologically detailed models available.

References [42, 64–67] use neuron models of low biological detail to achieve power savings. These neuron models cannot replicate all behaviours of real biologial neurons and the power savings are not replicable with a higher level of biological detail in these propsed implementations.

Among the vast digital spiking neuron implementations, few works prioritize power optimization, and among these works a small portion report power consumption for their implemented neurons. Most proposed implementations focus on minimizing hardware resource requirements. Although this is important, the neuron's power consumption is a critical factor for biological analogy [62]. Among the few digital implementations of biologically detailed neurons that report power consumption, specific hardware implementation techniques such as the CORDIC algorithm have been employed and shown to offer power savings [69]. Although the CORDIC algorithm is effective in reducing hardware resource and power consumption, it is an iterative algorithm which lowers the system throughput and this approach to power reduction limits the flexibility in the hardware design to the use of a single technique.

In designing SNNs, one of the most important considerations is the design of the spiking neurons used in the network. Many mathematical neuron models have been proposed with varying levels of biological detail and description [4, 5, 7, 8, 10]. Since neurons are dynamical systems, meaning they are neither linear nor time-invariant, implementing spiking neuron hardware systems presents a specific and substantial set of design challenges and considerations. Firstly, the implemented neuron must faithfully follow the behaviour of the target neuron model, exhibiting similar behavioural characteristics and bifurcations. Simultaneously, biological

neurons are highly efficient, exhibiting low power consumption, and are additionally physically very small. The power efficiency is due in part to the fact that biological neurons are active only when receiving stimulus [61]. Thus, low power consumption and hardware resource usage are critical as these factors are parallel with biology and permissive of larger SNNs.

To remedy this need for reduced power consumption in digital neurons, it was noted that in real biological neural systems, neurons are inactive most of the time [61, 62]. This inactivity is how biological systems achieve their low-power operation. In this work, a digital hardware implementation method by which inactivity is included in the digital neuron is proposed that can be applied to greatly reduce its power consumption. This implementation method is parallel to the inactivity observed in real biological neurons for low stimulus [63].

The proposed novel approach to digital hardware implementations of spiking neurons evaluates the neuron's differential equations at a frequency dependent on input current using a sampling-based approach. The variability in the frequency of the sampling reduces unnecessary switching activity for low-stimulus states. This novel Input-DEpendent Variable Sampling (I-DEVS) digital realization method to spiking neuron implementation results in neurons with minimal additional hardware resource usage in exchange for dramatic dynamic power savings as the switching activity is greatly reduced compared to traditional neuron implementations. Furthermore, the behaviour of the neuron is unaltered using this approach.

To verify the effectiveness of the I-DEVS digital neuron realization method, the Adaptive-Exponential Integrate-and-Fire (AdEx) model [8] and the Izhikevich model [7], were designed and implemented on an FPGA using the proposed method. The novel I-DEVS sampling-based method limits the number of times the differential equation is evaluated in the hardware system, meaning idle clock cycles are permissible when possible in the system which greatly reduces the power consumption of the neuron. The proposed implementation's inclusion of idle time

for low stimulus is analogous to the behaviour of real biological neurons under low-stimulus operational conditions, which allows for the low-power operation of the human nervous system [61, 62]. The unique approach allows for a large range of input current for which the neuron remains stable, while also allowing for a reduction in power consumption and uninhibited neuron performance.

4.2 Neuron Modelling

Two neuron models were selected for implementation, namely the Izhikevich Model [7] and the AdEx Model [8]. Both are characterized by a pair of differential equations that describe neuronal behaviour. The discretized form of the Izhikevich Model describes neuronal behaviour with the following two equations and reset condition:

$$v[n+1] = v[n] + dt(0.04v^2 + 5v + 140 - u + I) \tag{4.1}$$

$$u[n+1] = u[n] + dt(a(bv - u)) \tag{4.2}$$

$$if \quad v[n] \geq 30mV, then \begin{cases} v \rightarrow c \\ u \rightarrow u + d \end{cases} \tag{4.3}$$

Where v is the membrane potential, I is the input synaptic current, u is an auxiliary variable, c is the reset potential, a, b, and d are model parameters, and dt is the time step.

After discretization by the Euler method, the AdEx neuron model is given by:

$$V[n+1] = V[n] + dt\frac{1}{C}(-g_L(V - E_L) + g_L\Delta_T e^{\frac{V-V_T}{\Delta_T}} - w + I) \tag{4.4}$$

$$w[n+1] = w[n] + dt\frac{1}{\tau_w}(a(V - E_L) - w) \tag{4.5}$$

$$if \quad V[n] \geq 20mV, then \begin{cases} V \rightarrow V_r \\ w \rightarrow w + b \end{cases} \tag{4.6}$$

Where C is the membrane capacitance, g_L is the leak conductance, E_L is the leak reversal potential, V_T is the spike threshold, Δ_T is the slope factor, τ_w is the adaptation time constant, a is the subthreshold adaptation, b is the spike-triggered adaptation, V_r is the reset potential, and dt is the time step.

It is important to note that many proposed digital implementations of the AdEx and Izhikevich neuron models focus primarily on hardware design techniques that reduce resource requirements. Many acknowledge the significance of power consumption, but do not include detailed power analysis [73,74]. Given the significant advantages of digital neuron hardware implementations, namely their accuracy and tolerance for noise, the power consumption is a significant topic to address to ameliorate digital neuron designs and make them more accurate to real biology.

4.3 Proposed Power Reduction Methodology

4.3.1 Methodology

Analogous behaviour to real biological neurons was central to the development of the I-DEVS neuron implementation method. The stimulus-dependent activity of real neurons allows for the low-power operation of biological nervous systems [61–63], and in homology the I-DEVS method allows for inactivity to be included in the neuron's behaviour at a proportion determined by the input stimulus. Figure 4.1 shows a block diagram of the I-DEVS method. The input current and clock input to a comparator and timer block, where the input is compared to a threshold determined by the neuron model. Based on the input current, the timer and time step used in the neuron discretization are set. Once the timer reaches a value that is set dependent on the input current, the current, clock, and time step of

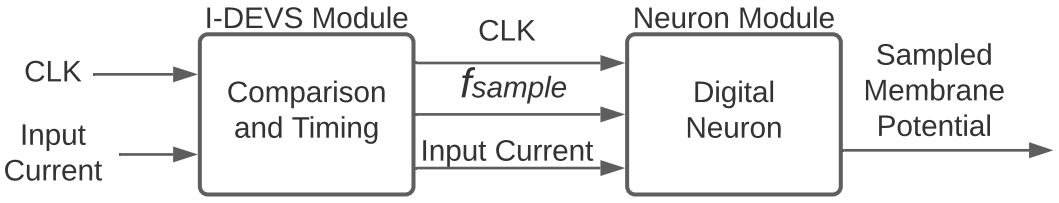


FIGURE 4.1: A high-level block diagram of the proposed I-DEVS neuron method.

discretization are passed to the neuron for the period of time necessary for the neuron to perform one evaluation, then, upon the receipt of a handshake signal, the neuron's input and clock are disabled again. This effectively samples the neuron's output membrane potential at a variable sampling frequency.

The variability in the sampling is critical to allowing the neuron to produce valid, uncorrupted output for a wide domain of input current while avoiding unnecessary switching in the digital neuron for low input stimulus. The sampling-based approach allows for large amounts of inactivity in the neuron circuit, which greatly reduces the digital switching activity, leading to a reduced dynamic power consumption in the system. An important novelty of the proposed I-DEVS method is that the neuron model and digital hardware architecture have no restriction and do not require modification to successfully utilize the method.

4.3.2 Simulations for Validation and Design Formulation

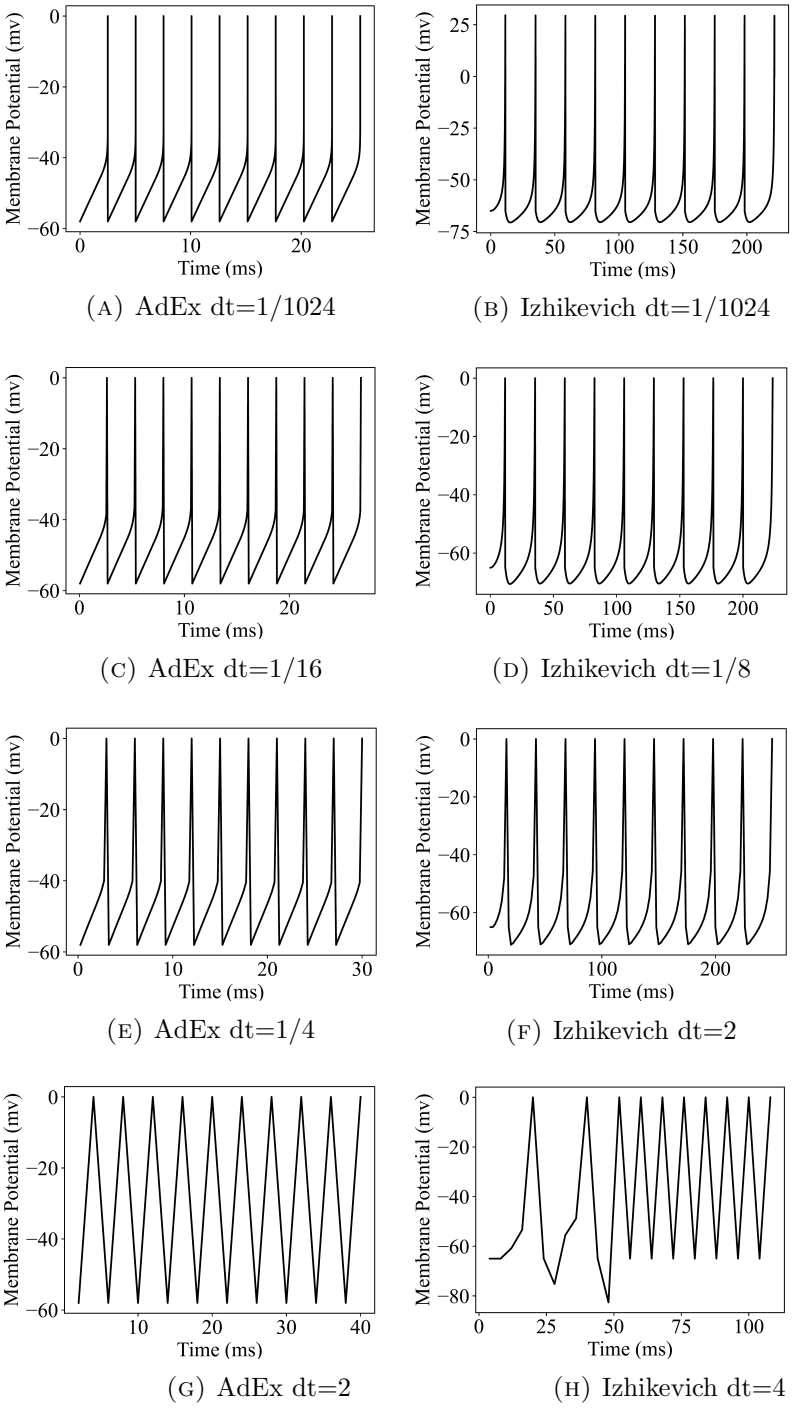
Software simulations were conducted to evaluate the stability and validity of the behaviours of both the AdEx [8] and Izhikevich [7] neuron models for varying input currents for a given sampling frequency. To successfully realize a neuron with a variable time step, it was first necessary to assess the effects of the size of the time step on the valid input current range as well as the power consumption associated with a given time step.

Figure 4.2 shows simulations of both neuron models for a constant input current with varying time step. It is evident from Figure 4.2 that a large time step has an adverse effect on the accuracy of the neuron's membrane potential. However, its is also clear that the smallest possible time step is not always necessary for high accuracy, meaning a time step below a given size for a given input current will result in redundant and unnecessary additional calculations of the neuron's differential equations. There is no visible qualitative change between Figure 4.2 (A) and (C) (AdEx neuron for $dt = 1/1024$ and $dt = 1/16$) or between (B) and (D) (Izhikevich neuron for $dt = 1/1024$ and $dt = 1/8$), which means for those two given input currents time steps smaller than 1/16 and 1/8 respectively are not necessary to achieve desired behaviour.

Figure 4.3 shows the value of the time step and current for which each neuron becomes unstable, which directly correlates to stability at a given sampling frequency in digital hardware. As expected, higher current input requires a smaller time step to maintain stability in the neuron's output. Figure 4.3 provides highly important information for the design of a variable time step system as the points at which the time step should change can be inferred from the curve. It logically follows that a lower time step results in a greater valid input current domain.

Using the information gathered from simulation, a system was designed in which the time step of the neuron discretization is variable. The time step assumes one of three different values depending on the input current applied to the neuron. The threshold time steps found from Figure 4.3 were used to determine the proper transition points for the time step in the domain of the input current.

Figures 4.4 and 4.5 show simulations of the proposed neurons with variable time step. It is evident that the stability of the neuron is maintained for a large range of current. However, the unnecessarily high computation time and power consumption associated with a high time step is avoided for input current values for which it is not necessary. In digital hardware, the use of a high time step for low input



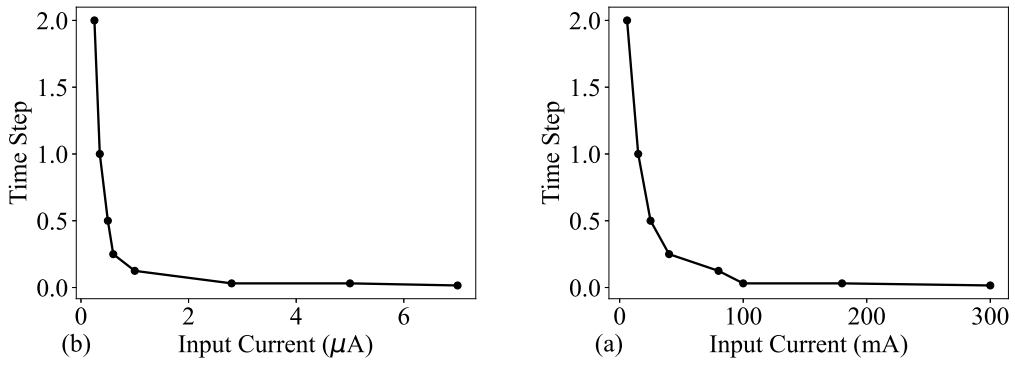
(A) AdEx dt=1/1024 (B) Izhikevich dt=1/1024

(C) AdEx dt=1/16 (D) Izhikevich dt=1/8

(E) AdEx dt=1/4 (F) Izhikevich dt=2

(G) AdEx dt=2 (H) Izhikevich dt=4

FIGURE 4.2: Membrane potential waveform of (A), (C), (E), (G) the AdEx (B), (D), (F), (H) the Izhikevich neurons for different time steps.



(A) AdEx (B) Izhikevich

FIGURE 4.3: Threshold time steps at which the (A) AdEx and (B) Izhikevich neurons become unstable as function of input current.

currents to avoid unnecessary calculation translates to lower switching activity, which implies lowered dynamic power consumption. All of the above-mentioned effects of time step variation were characterized in [75].

4.4 Validation in Digital Hardware

4.4.1 Hardware Design

The proposed neuron systems were implemented on FPGA. Consistent implementations for both the AdEx and Izhikevich neuron were used between the I-DEVS and the traditional implementations. The AdEx neuron was implemented in Verilog using a 37-bit signed fixed-point implementation with 1 sign bit, 13 integer bits, and 23 fractional bits. The digital word length was selected to accommodate the CORDIC algorithm used for the implementation of the exponential term in the AdEx neuron model. The Izhikevich neuron was implemented in VHDL using a 33-bit signed fixed point with 1 sign bit, 18 integer bits, and 14 fractional bits. The v^2 in the Izhikevich neuron model was implemented using a DSP multiplier.

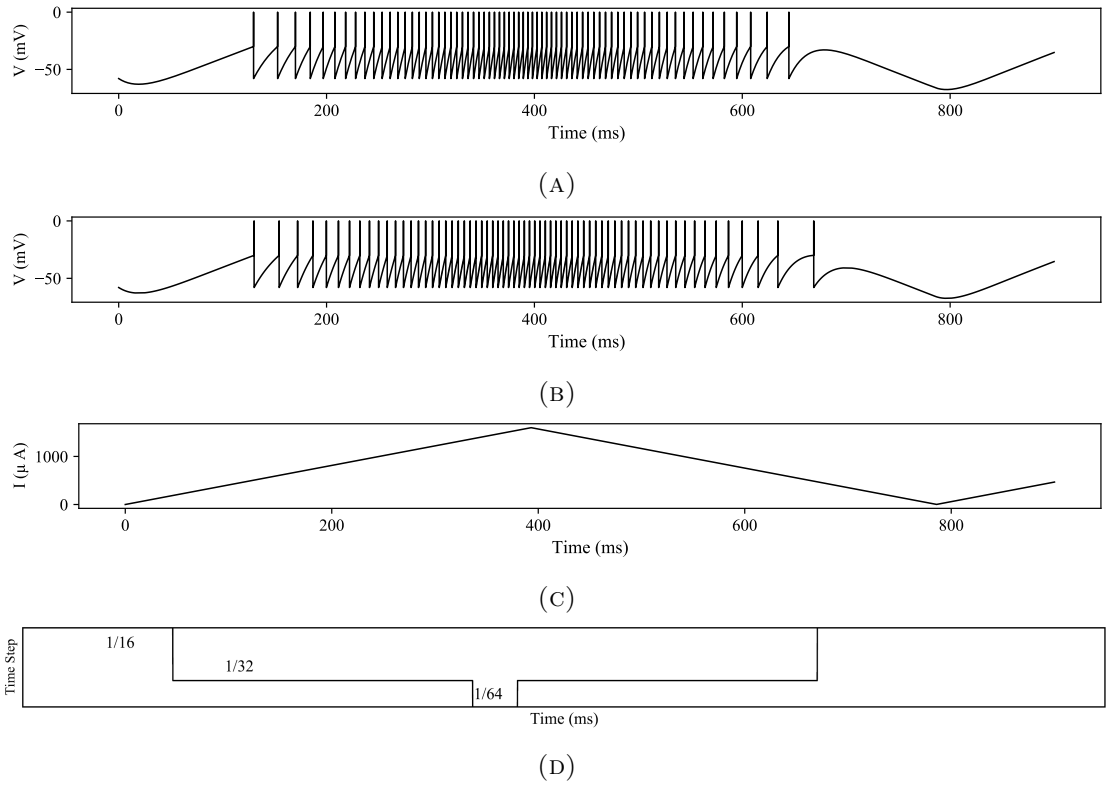


FIGURE 4.4: Simulations of the AdEx Neuron with and without a variable time step for a triangular wave current. (A) shows the membrane potential of an AdEx neuron with a fixed timestep of $dt = 1/512$ (B) shows the membrane potential of an AdEx neuron with the I-DEVS method, (C) shows the input current, and (D) shows the value of dt as a function of time.

Although this is arguably not ideal for neuron implementations, the primary focus of this work is validation of the independence of the I-DEVS method on the architecture of the neuron, so diverse neuron implementation architectures were explored. In both neurons, the I-DEVS module was implemented using a state machine to control a comparator to the input current, and an 8-bit timer with three settings based on the input current range. The timer is used to determine when to pass the clock and input current to the neuron to acquire another sample. Figure 4.6 shows a block diagram of the I-DEVS hardware module.

The effectiveness of the I-DEVS method was assessed firstly in simulations to verify that the implemented hardware systems exhibit the targeted functionality. Functional simulations were successful. It was noted that the spike frequency of

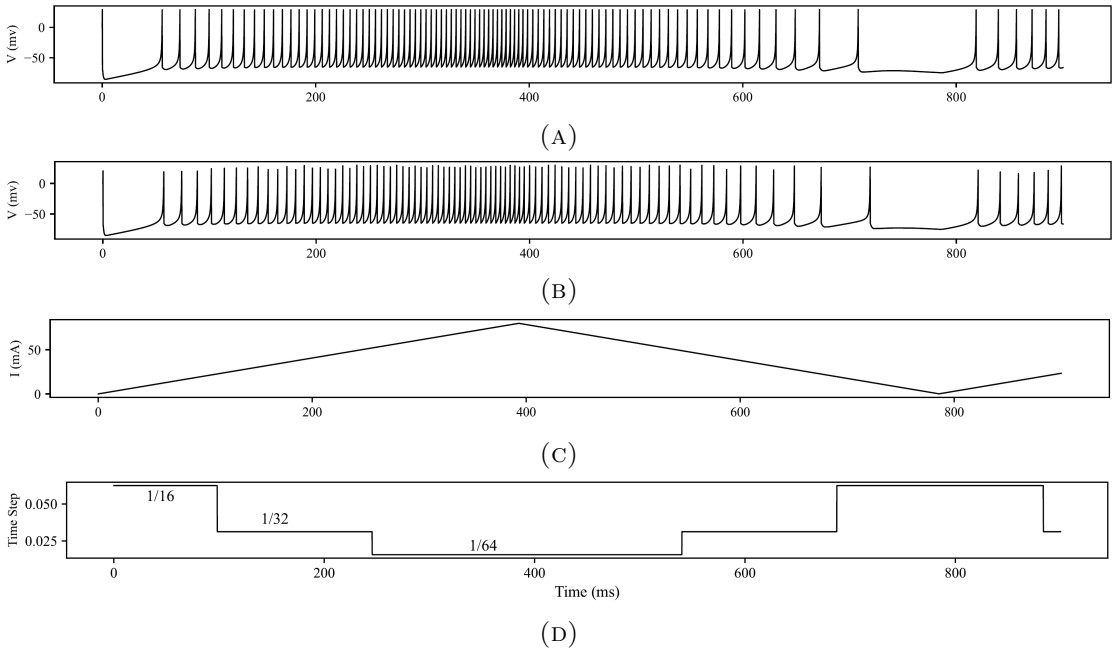


FIGURE 4.5: Simulations of the Izhikevich Neuron with and without a variable time step for a triangular wave current. (A) shows the membrane potential of an Izhikevich neuron with a fixed timestep of $dt = 1/512$ (B) shows the membrane potential of an Izhikevich neuron with the I-DEVS method, (C) shows the input current, and (D) shows the value of dt as a function of time

the neurons implemented using the I-DEVS method was lower than those of the traditional hardware implementations. This was observed in software simulations as well and was expected as more points of the difference equations are computed per unit time. This does not effect the bifurcative behaviours of the neurons since bifurcation analysis is performed independently of the discretized neuron, meaning that their network performance will be uninhibited as the final synaptic weights of a trained network will be different in compensation for the different spike frequencies [3].

Furthermore, the projected savings in circuit switching activity were assessed. To validate theoretical expectations, the number of times each implementation evaluates the neuron's difference equations per average time was evaluated through functional simulation of the implemented neurons for different ranges of input current. Figure 4.7 shows the savings in evaluations per time as well as the savings in

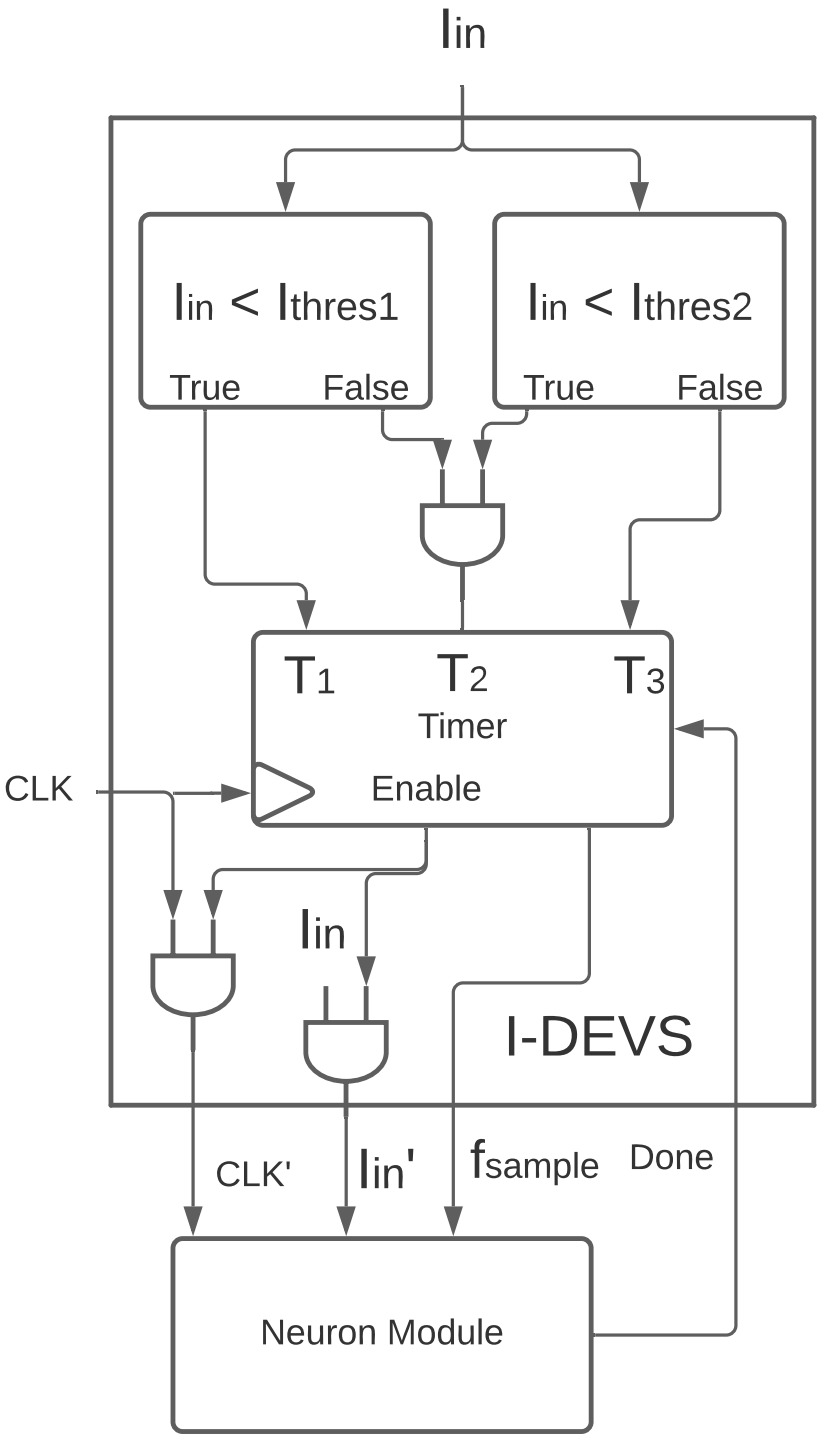
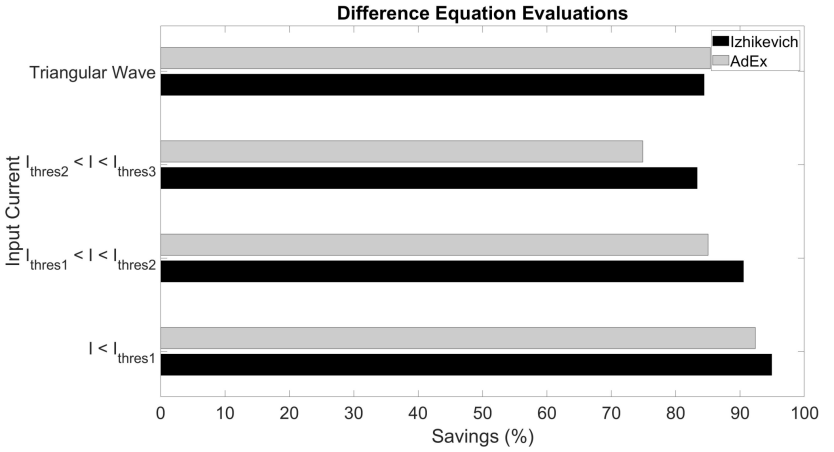
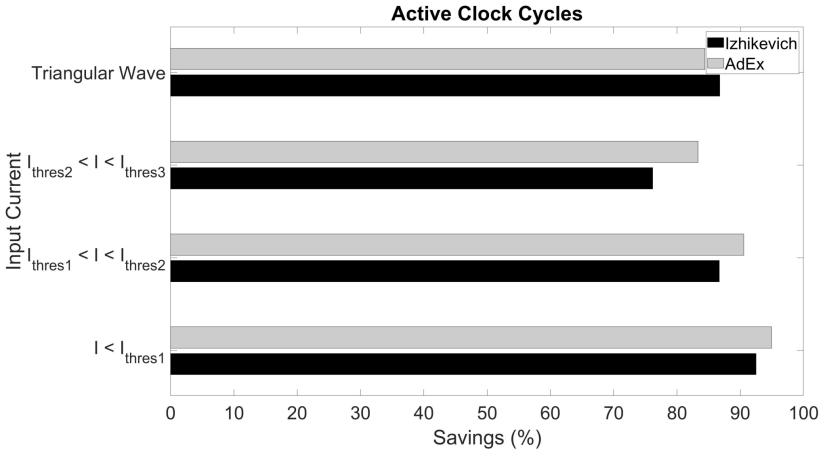


FIGURE 4.6: A block diagram of the digital hardware implementation I-DEVS module.



(A)



(B)

FIGURE 4.7: Computational savings per time from the I-DEVS method compared to traditional implementations for the AdEx and Izhikevich neurons. (A) shows the savings in difference equation evaluations and (B) shows the savings in clock cycles for which the neuron is active.

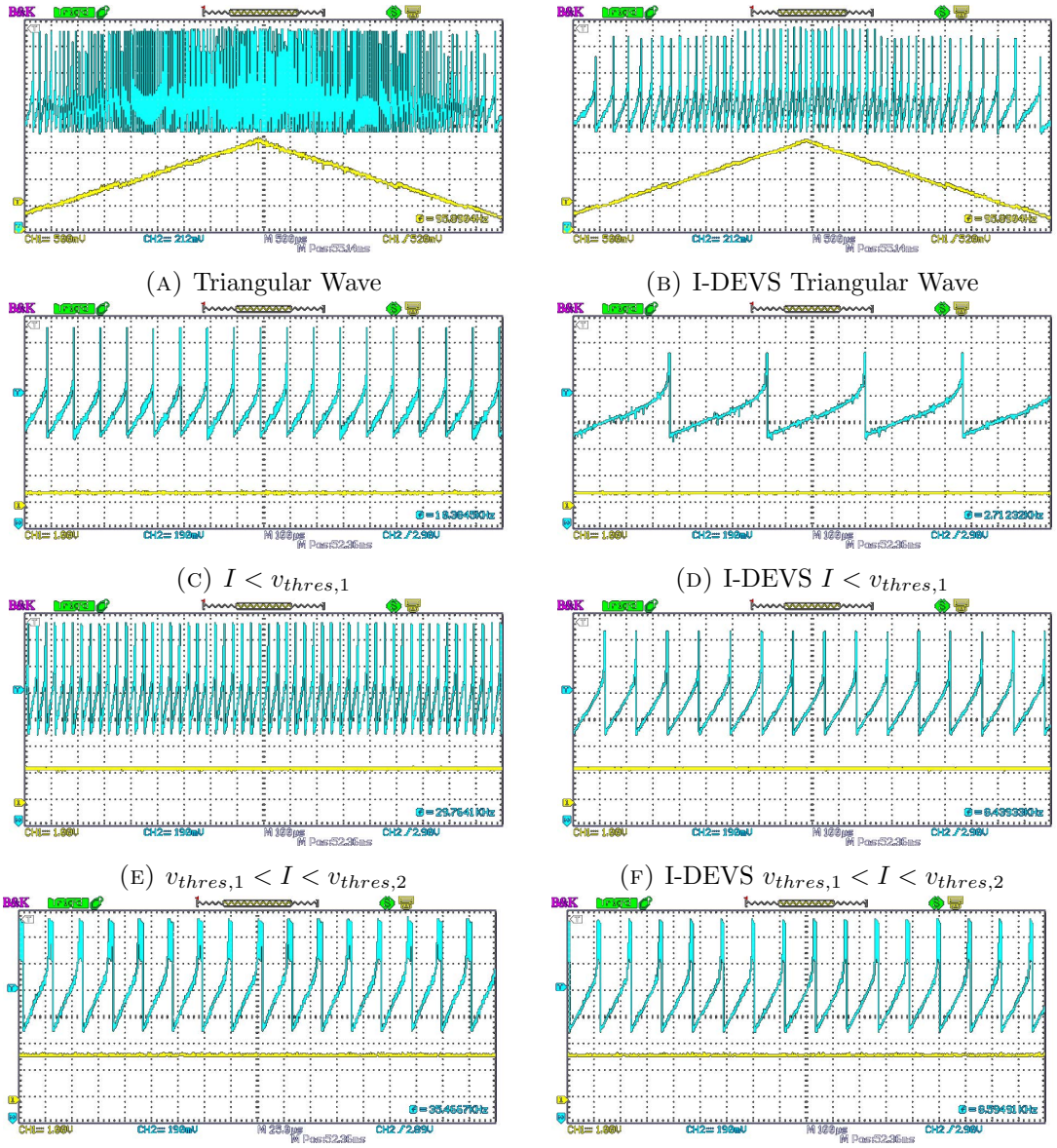
the number of clock cycles for which the neuron is actively performing calculations. The substantial savings in activity implies reductions in switching activity in the system, which leads to reductions in power consumption. The third threshold current, I_{thres3}, shown in Figure 4.7 denotes the input current at which the output becomes corrupt for the smallest time step used in the systems of $dt = 1/128$. It is important to note that this threshold was not implemented since input current in this range does not contribute significant information to the behaviour of the neuron models [7, 8].

TABLE 4.1: FPGA Resource Usage Information on the Xilinx Spartan 7 for neuron implementations with and without the I-DEVS method.

Neuron Model	**LUT**	**FF**	**DSP**	**I/O**	**Max. Clock (MHz)**
AdEx	1338	479	0	108	120.31
AdEx I-DEVS	1582	510	0	115	119.42
Izhikevich	397	194	4	67	249.25
Izhikevich I-DEVS	451	217	4	67	249.25

The proposed system was implemented and validated on an Altera DE0 development board. For comparison, traditional implementation methods were implemented and tested as well alongside the I-DEVS neurons. Figures 4.8 and 4.9 show oscilloscope images for both the implemented AdEx and Izhikevich models. The digital membrane voltage was converted to an analog signal using a 12-bit resistive ladder for digital-to-analog conversion to view the waveform on the oscilloscope. It is evident that the behaviour of the implemented neuron is consistent with expectations. Although the I-DEVS neurons exhibit lower spiking frequencies than their fixed time step counterparts, the I-DEVS and traditional implementations exhibit consistent spiking behaviour for a given input current.

The prominence of the spike frequency difference was observed to be greater in the digital hardware implementations compared to the software simulations. This can be explained by the number of clock cycles required to evaluate the neuron's difference equations in hardware. Since both neurons were implemented using state machines (and in the case of the AdEx neuron, with an iterative algorithm), more than one clock cycle is required to evaluate the difference equations, which lowers the effective sampling frequency compared to simulation. Adjustments to the I-DEVS module's timer values could compensate if a higher spike frequency is necessary for a given application. However, it is important to note again that the I-DEVS method does not change the bifurcative behaviour of the neuron, meaning traditional neuron implementations and I-DEVS neuron implementations exhibit the same behaviour.



(A) Triangular Wave

(B) I-DEVS Triangular Wave

(C) $I < v_{thres,1}$

(D) I-DEVS $I < v_{thres,1}$

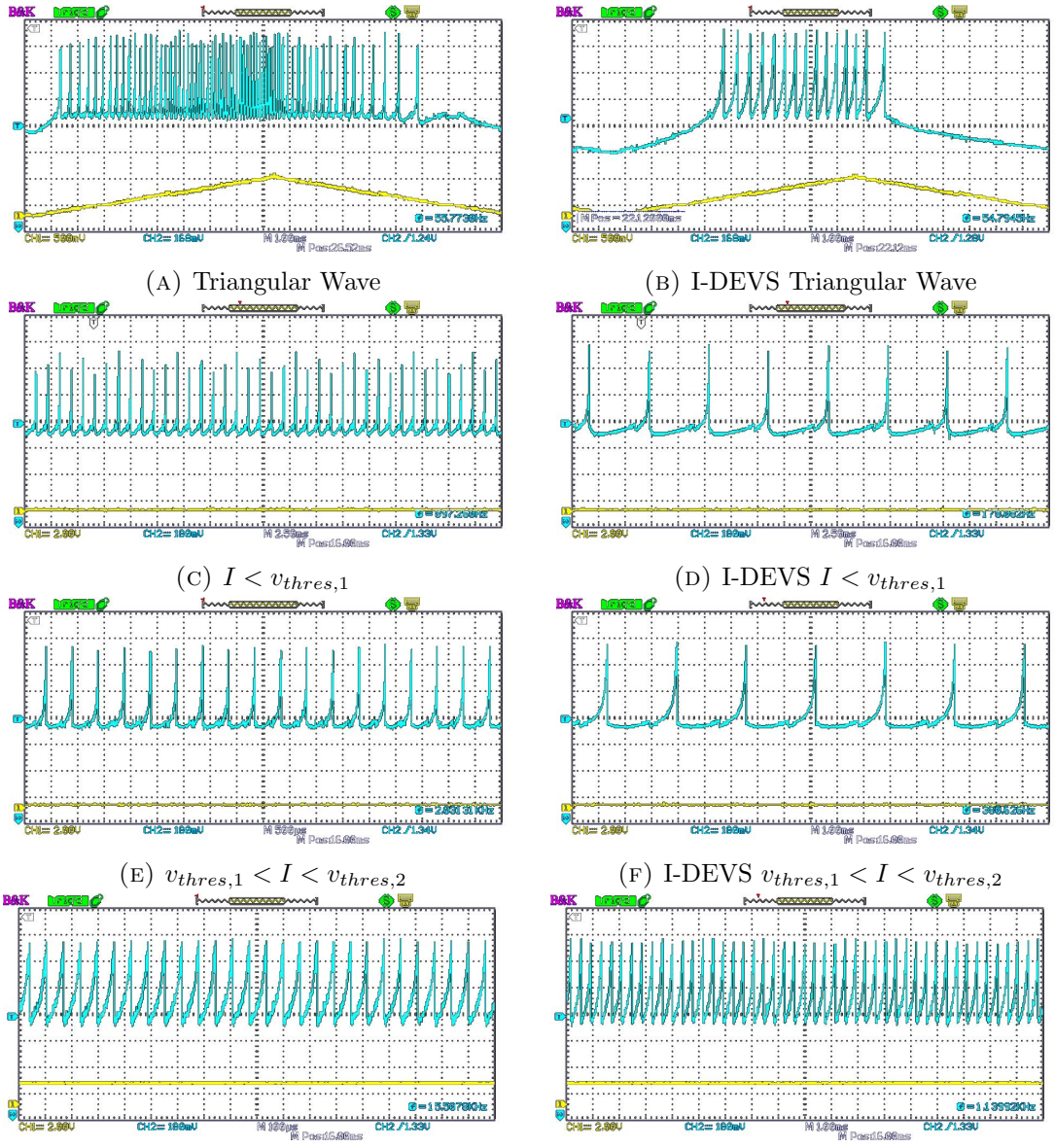
(E) $v_{thres,1} < I < v_{thres,2}$

(F) I-DEVS $v_{thres,1} < I < v_{thres,2}$

(G) $I > v_{thres,2}$

(H) I-DEVS $I > v_{thres,2}$

FIGURE 4.8: Oscilloscope images for (A), (C), (E), (G) an AdEx Neuron implemented using traditional hardware implementation, and (B), (D), (F), (H) the same neuron implemented using the I-DEVS method. Membrane potential is shown in blue and input current is shown in yellow. Horizontally paired images show the same input current.



(A) Triangular Wave

(B) I-DEVS Triangular Wave

(C) $I < v_{thres,1}$

(D) I-DEVS $I < v_{thres,1}$

(E) $v_{thres,1} < I < v_{thres,2}$

(F) I-DEVS $v_{thres,1} < I < v_{thres,2}$

(G) $I > v_{thres,2}$

(H) I-DEVS $I > v_{thres,2}$

FIGURE 4.9: Oscilloscope images for (A), (C), (E), (G) an Izhikevich Neuron implemented using traditional hardware implementation, and (B), (D), (F), (H) the same digital neuron implemented using the I-DEVS method. Membrane potential is shown in blue and input current is shown in yellow. Horizontally paired images show the same input current.

Furthermore, Table 4.1 shows the FPGA hardware resource usage information for each neuron implemented both with a fixed time step and with the I-DEVS method on the Xilinx Spartan 7 FPGA. It is obvious that the I-DEVS method used to greatly reduce the neuron's power consumption presents a very minor trade-off in hardware usage.

4.4.2 Power Analysis

The power consumption of the variable time step neurons was analyzed using Vivado's Power Estimator and compared with the power consumption of fixed time step implementations. Using Vivado's Power Analysis tool, the FPGA implementations showed a 16.67% best case power reduction. Although this is inconsistent with the much larger anticipated power savings, it is noted that FPGA implementations are limited in their power savings by the architecture of the FPGA [76]. To better capture the power savings offered by the I-DEVS method, the neurons were synthesized in Synopsys and the power consumption was estimated for a digital ASIC design. Table 4.2 shows power consumption information for the original and I-DEVS neurons. The substantial power savings shown are consistent with the expectations associated with the savings in switching activity at a clock frequency of 50MHz. As expected, the power savings are lower than the computation savings due to parasitic elements in the ASIC implementation that are prominent with circuitry on fast-switching lines such as the clock line in digital systems [77]. Furthermore, the power consumption is much lower in ASIC implementation estimations for all neurons as expected due to the architecture of the FPGA.

Since the power consumption of a digital Izhikevich neuron is reported for a clock frequency of 9.1MHz in [69], Synopsys power estimates were also collected for a clock frequency of 10MHz for the Izhikevich implementation to create a fair

TABLE 4.2: Dynamic Power Estimates for a 50MHz clock for ASIC and FPGA digital implementations of the AdEx and Izhikevich neurons using the I-DEVS method. Note that the I-DEVS column for power reports from [69] is used to report the power consumption of the CORDIC implementation.

Clock		**Neuron Model**	**Original (mW)**	**I-DEVS (mW)**	**Savings (%)**
50MHz	ASIC	AdEx	4.4733	1.4662	67.22
		Izhikevich	2.0344	0.69369	65.90
	FPGA	AdEx	18	15	16.67
		Izhikevich	7	7	0
10MHz	ASIC	Izhikevich	0.4081	0.1391	65.91
9.1MHz		Izhikevich [69]	1.06	0.33	68.87

comparison. As shown below, the I-DEVS method offers very similar power savings without imposing restrictions on the neuron's implementation architecture.

It is also important to note again that the AdEx neuron used in this study was implemented using the CORDIC algorithm for both the I-DEVS and traditional neuron, and the Izhikevich neuron was implemented using DSP multipliers in both cases. Different design approaches, digital word lengths, and neuron models were used to support the assertation that the I-DEVS method can reduce the power consumption for neurons of higher biological detail independent of the architecture of the neuron.

Although the relationship of the switching activity savings shown in Figure 4.7 to the power savings reported in Table 4.2 is not directly proportional, there is a clear and evident relationship between the switching activity and the dynamic power consumption of the neuron. Thus, the I-DEVS method is highly effective in reducing dynamic power consumption through the inclusion of inactivity in the neuron.

4.5 Conclusion

Two spiking neurons were implemented in digital hardware via FPGA with the novel I-DEVS sampling-based digital neuron implementation method. The valid input current domain is substantially large, and simultaneously the implementation exhibits a lower power consumption compared to traditional implementations with fixed small time step as unnecessary active computation time for lower input currents is avoided. The I-DEVS method showed consistent reductions in switching activity and thus reductions in dynamic power consumption in the neuron independent of the neuron model and implementation method of the neuron module.

The application of the I-DEVS method offers substantial advantages for neurons to be used in SNNs as the power consumption is a key metric and an important consideration for SNN design. The reduced dynamic power consumption of neurons implemented using the I-DEVS method offers potential for larger networks of biologically detailed neurons.

Chapter 5

Selective Input Sparsity for Spiking Neural Network Size Reduction

5.1 Introduction

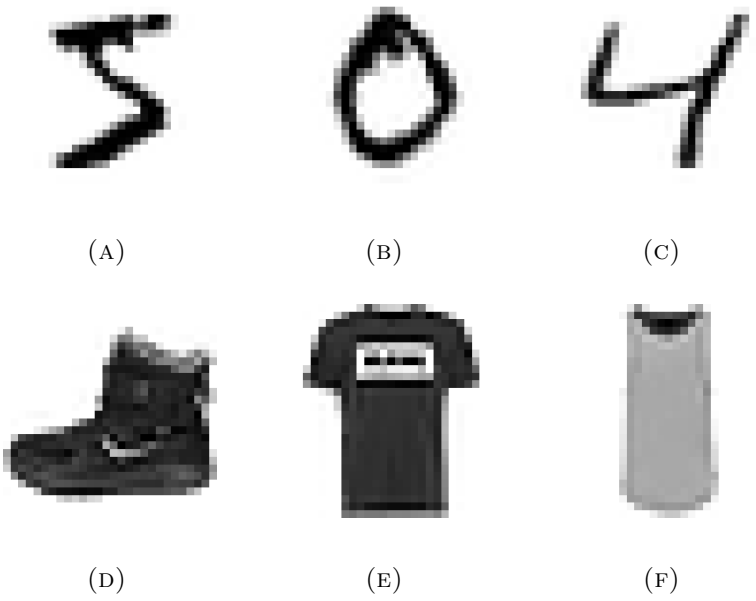
As explained in Chapter 1, SNNs come with potential benefits over traditional ANNs, most notably temporal sparsity [12, 21], which may lead to lower switching activity and thus lower power consumption in hardware when implemented effectively [77]. Again considering the ever-increasing inclusion of AI systems in commercial applications and the environmental impacts of high-power computing associated with data processing [78], SNNs are an important option to fully excavate for the future of AI.

Hardware implementations of SNNs have substantial potential for the above-mentioned reasons. However, in hardware the size of the network is a significant consideration. Large, fully parallel networks consume substantial silicon area and

require a great deal of interconnection which leads to routing challenges. Convolutional SNNs [79–82] and feedfoward architectures [83–85] have been proposed as an effective solution for many pattern recognition problems and have achieved exceptional accuracy. However, the hardware resource requirements for implementing such networks is extraordinarily high. Furthermore, methods such as downsampling input data [15] have been deployed to simplify hardware requirements and increase accuracy. Although effective, that requires substantial data pre-processing which translates to delay and additional hardware in edge applications.

Sparsity in neural networks is a fascinating and exciting concept because it offers the potential to reduce the network size and has been shown to have limited adverse effect on classification accuracy in pattern recognition applications [86,87]. Sparsity in the interconnections of network layers is a highly active topic of interest for these reasons.

Here the novel concept of input sparsity in image classification problems is introduced and explored. Input sparsity refers to the removal of connections to input pixels without concern for the preservation of image integrity. The primary motivation goal of the introduction of input sparsity to SNNs is network size reduction, decreased inference and training time, and lowered hardware resource requirements and power consumption in hardware implementations. Following initial exploration, a methodology by which input sparsity can be selectively introduced based on the training image set of a dataset is proposed. The proposed Selective Input Sparsity (SIS) methodology not only reduces network size but increases the network's inference accuracy in the datasets to which it was applied. The inference time and accuracy of SNNs created with the SIS architecture are compared to the performance of SNNs with random input sparsity and it is shown that the SIS method introduces substantial sparsity in the network while maintaining comparable accuracy.



(A) (B) (C)

(D) (E) (F)

FIGURE 5.1: Sample images from (a) - (c) the MNIST handwritten digit database [88] and (d) - (f) the Fashion MNIST database [89].

5.2 Background Simulation

Firstly, a study of the effects of pixel deletion in the input image was conducted. Two data sets were used for exploration; the Modified National Institute of Standards and Technology (MNIST) handwritten digits dataset containing handwritten digits from 0 to 9 [88], and the "Fashion MNIST" dataset that contains images of fashion items belonging to ten different classes [89]. Sample images from both datasets are shown in Figure 5.1. Both datasets contain 60000 training images and 10000 test images that are 784 pixels and each pixel is an 8 bit greyscale value from 0 to 255. In the case of the digit MNIST dataset, the images were deskewed following the approach in [90]. Since the main consideration of the proposed networks is hardware feasibility, network size is critical as it relates to power consumption, silicon area, and speed which are all main hardware performance metrics. Deskewing operations are common in smaller networks when approaching pattern classification problems [88].

To develop baseline results and examine the possible benefits of input sparsity, simulations were conducted for varying levels of input sparsity for random pixel deletion. For sparsity ranging from 0 (a fully connected network) to 0.9 (90% of all input pixels removed), the accuracy of a two-layer network was assessed for the stated sparsity range in increments of 0.1. 100 trials were conducted for levels of input sparsity ranging from 0 to 0.9 in increments of 0.1. The number of deleted pixels for a given trial corresponded to the level of sparsity. For example, a sparsity of 0.1 means that the number of deleted pixels is the rounded product of 0.1 and 784, the total number of pixels. Thus, at a sparsity of 0.1, 78 pixels were randomly deleted for each of the 100 trials. Since for each trial a different set of pixels were removed at random, 100 trials were used to observe variation in the network's classification accuracy for different configurations of input sparsity. A sparsity of 1.0 implies that the input is not connected to the network and is therefore meaningless.

The network, as stated above, was a simple two-layer network comprised of an input layer of n pixels, where n is the number of pixels in the input layer, and ten output neurons with Rectified Linear Unit (ReLU) activation functions. No biases were used in addition to the network weights for simplicity. Many recent finds have found that the trained weights from a simple network of ReLU non-spiking neurons are mappable to an analogous structure as an SNN with relatively low accuracy loss [12,13]. The weights were then applied to an SNN, input pixels were converted to spike trains using non-leaky unsigned 8-bit integer Integrate-and-Fire (IF) neurons, and IF neurons were used in place of ReLU activation functions. Although the network after pixel deletion is fully connected, this method is fundamentally different from downsampling as features in the image are not necessarily preserved since pixels are deleted at random.

In both datasets, the pixel intensities were normalized into an 8-bit fractional range through division by 256 for training a traditional ANN. Input normalization is a helpful step for weight convergence in ANNs. Tensorflow with Keras was used

to train a non-spiking network using Sparse Categorical Cross Entropy as the loss function and Stochastic Gradient Descent with the Adam optimizer [91]. Pixels are deleted from the input image at random according to the sparsity level, and the remaining pixels are connected to ten output neurons, one for each class of input, with ReLU activation functions in a fully connected manner.

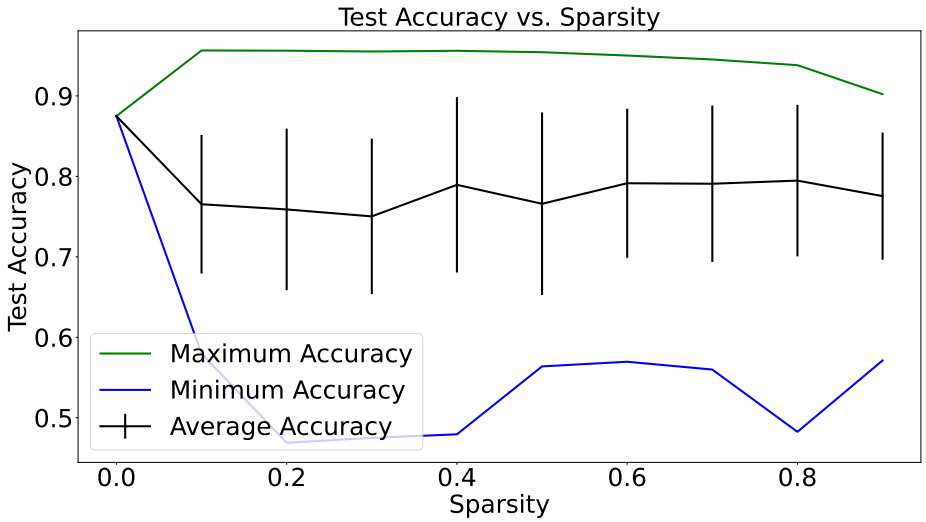
Figure 5.2 shows the average classification accuracy and minimum and maximum accuracy observed for each tested level of sparsity. From the experimental results, it is not apparent that increasing sparsity results in lower classification accuracy. The results do not present an obvious trend. An interesting finding is that the maximum accuracy of networks with input sparsity exceeds that of the fully connected network. This may imply that input sparsity in SNNs, much like sparse connections, can meet or exceed baseline performance when properly realized [87].

However, it is interesting to observe that the maximum accuracy of the network remains high for high levels of sparsity. Conversely, some sparse inputs yield very low accuracy. A clear implication of this trend is that some pixels are far more significant in class distinction than others.

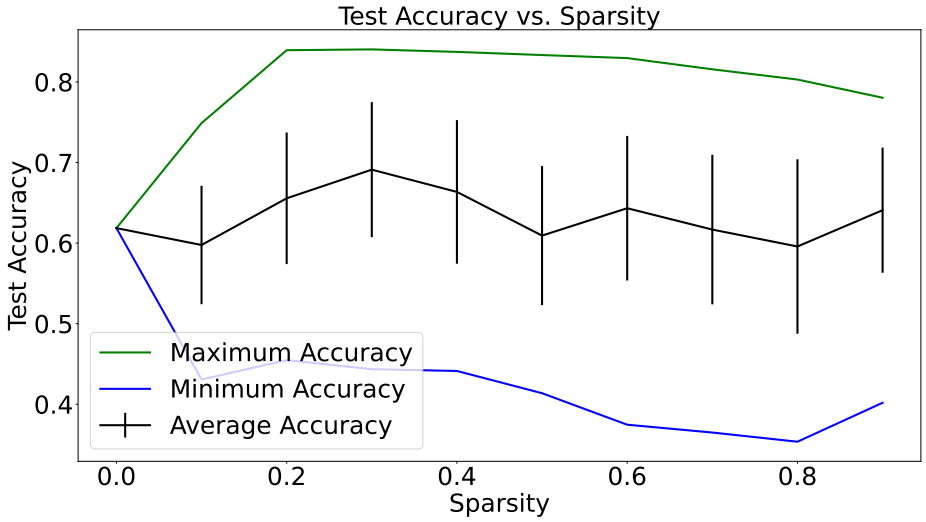
5.2.1 Other Explored Reduction Methods

Another interesting method of hardware reduction that was explored was miniature, independent SNNs that make inferences based on localized regions of the input image. The miniature networks then voted based on how they classified their local region of the input image to develop an overall classification.

Although this method showed itself to be effective in reducing inference time and reducing prospective hardware resource requirements, the accuracy of this method suffered substantially compared to baseline methods. Although the concept is interesting, it may require further exploration to fully excavate the possible savings while preserving classification accuracy.



(A) MNIST



(B) Fashion MNIST

FIGURE 5.2: Average, maximum, and minimum classification accuracy observed from 100 trials at each level of input sparsity. It is interesting to observe that the proposed experimentation does not present a clear relationship between the input sparsity and classification accuracy.

5.3 Proposed Selective Input Sparsity

5.3.1 The Selective Input Sparsity (SIS) Method

In the background simulations, it was noted that no obvious trend between input sparsity and inference accuracy was observed in the experimental results. Naturally, this leads to curiosity and further investigation to evaluate why some sparse configurations showed substantially higher classification accuracy than others. Given the nearly linear decrease in inference time with input sparsity and the linear decrease in the number of synapses, it would be highly advantageous to derive a method by which the significance of input pixels can be evaluated, and deletions can be made based on the training set to create sparsity in the network before training.

The proposed novel method of Selective Input Sparsity (SIS) involves inferring information from the training set to reduce the network size. In experimentation, the intensity of the pixels in the first n_1 training set images were averaged to determine a threshold pixel value, t. Then, a tally of the number of times a pixel in each index exceeds the threshold value in the next n_2 images was conducted. a_U and a_L are both less than 1 as they are used to assess the a given pixel's activity with respect to the number of images in group n_2. If the tally for a given pixel exceeded a_L of the n_2 images, which was 10% in the experiment, this implies that at least one of the image classes requires that pixel for distinction, so it was kept. Conversely, if the tally for a given pixel exceeded $a_U = 50\%$ of the n_2 images, this implies that the pixel may not help in distinguishing classes, so the pixel was discarded. In the cases of the digit MNIST and fashion MNIST datasets in the proposed experimentation, $n_1 = n_2 = 10000$. This selection method can be summarized by the following algorithm:

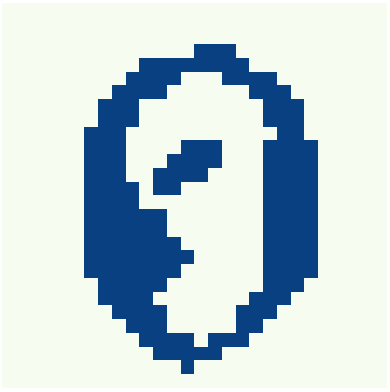
For the two datasets, the threshold pixel intensity of the normalized data was found to be 0.1317 for the digit MNIST dataset and 0.2852 for the fashion MNIST

Algorithm 4: The proposed SIS method. p is the number of pixels per image, n_1 and n_2 are partitions of the training images, t is the threshold pixel intensity, count is the tally of instances where a pixel exceeds the threshold intensity, x represents the training set images, a_U and a_L are the upper and lower limits of pixel activity, and index is the list of indices of retained pixels.

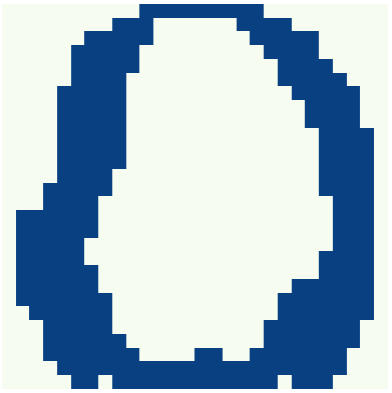
```
for i ← 1 to n_1 do
    y[i] = average of pixels of x[i]
end
t = average of y
for i ← 1 to n_1 + n_2 + 1 do
    for j ← 1 to p do
        if x[i,j] ≥ t then
            count[i] = count[i] + 1
        end
    end
end
for i ← 1 to p do
    if count[i] ≥ a_L n_2 AND count[i] ≤ a_U n_2 then
        Add pixel j to index
    end
end
```

dataset. This variation in the threshold pixel intensity corresponds to different levels of input space usership by the two datasets. This difference is helpful in assessing the effectiveness of the proposed SIS method for network implementation for distinct classification problems. Figure 5.3 shows input space receptive field maps for both datasets after network implementation using the SIS method. For the digit MNIST dataset 597 input pixels were removed while 489 pixels were removed in the fashion MNIST dataset, corresponding to sparsity values of 76.15% and 62.37% respectively. Additionally, the SIS method was effective in finding pixels that gave the maximum observed accuracy in the random experiments for a given level of input sparsity.

It is important to note that Figure 5.3 shows retained and omitted pixels for the structure of the SNN. Although Algorithm 4 may resemble edge detection, it is not a pre-processing step applied to input data for inference, but rather is used to



(A) Digit MNIST



(B) Fashion MNIST

(C) Digit MNIST

(D) Fashion MNIST

FIGURE 5.3: (A) and (B) A receptive field map of the retained and discarded pixels in the SIS networks for both datasets. Blue pixels were retained. (C) and (D) show sample images applied to the receptive field maps.

select retained sparse connections for the whole network.

5.3.2 Software Validation

ANNs were trained for both a baseline Fully-Connected (FC) network and an SIS network for both datasets. 100 trials were conducted for each network to

find the random weight initialization that converged to the maximum observed classification accuracy for the network.

Subsequently, the trained ANNs were quantized to 8-bit signed integer representation for all network weights using Keras quantization-aware training and the TensorFlowLite tool. This was done to simplify hardware implementation requirements and the accuracy change was negligible in most cases since the networks are small with two layers.

SNNs with pre-trained weights were simulated to make inferences on all images in the test set images for each dataset using the quantized weights. In both cases, the test images are different from the training images. The images were converted to rate codes using simple 8-bit unsigned IF neurons.

Since SNNs are time dynamic, two important considerations in the conversion from traditional ANN to SNN is the time step of discretization and the exposure time of each image. As both these parameters increase, the accuracy of the SNN approaches that the of the pre-trained ANN with the same weights. However, longer exposure times require longer simulations, and higher time steps remove the advantage of temporal sparsity presented by SNNs [12]. After investigation, a balance between accuracy and inference time was found for $dt = 0.25$ for 16 time steps.

In the case of the MNIST handwritten digits, a selective sparsity of 76.15% obtained an accuracy of 90.87% compared to a baseline accuracy of 95.1% for all 10000 test images. Table 5.1 summarizes this information as well as spike thresholds for input and output layer neurons. Spiking thresholds were selected as powers of 2 so they can be treated as flag bits in hardware. It is important to note that the goal was not to solve the MNIST handwritten digits with a record accuracy, as exceptional accuracies have already been achieved [83–85, 92, 93]. Instead, the goal was to determine if the pre-training network can be created with targeted sparse

TABLE 5.1: Python simulation results for all 10000 test images for baseline Fully-Connected (FC) SNNs and SNNs with Selective Input Sparsity (SIS). $dt = 0.25$ and each test image was exposed for 16 time steps.

	FC Digits	**SIS Digits**	**FC Fashion**	**SIS Fashion**
Input Pixels	784	187	784	295
Synapses	7840	1870	7840	2950
Sparsity	0	76.15%	0	62.37%
Inference Accuracy	95.1%	90.87%	80.28%	80.22%
Inference Time (m:s)	4:10.58	1:25.39	07:49.32	2:31.33
$v_{th,in}$	128	128	128	128
$v_{th,out}$	64	32	16	16

input connections and evaluate the efficacy of the SIS method when applied to a baseline architecture.

The input sparsity introduced by the SIS method is lesser in the case of the Fashion MNIST dataset as there are more pixels that are frequently part of the image foreground. However, the introduced sparsity is still substantial at 62.36%. Furthermore, it was noted that the accuracy of the networks in both the baseline and selective sparsity cases was relatively low. This is likely due to the higher complexity of the Fashion MNIST dataset compared to the MNIST handwritten digits dataset [89]. Since both the baseline and selectively sparse-input networks have depressed accuracy, a different network architecture may be required to obtain a better accuracy. The architecture used is small as hardware implementation is the primary objective. Nevertheless, the introduction of selective sparsity reduces the number of required synapses and yields a very similar classification accuracy. An important note is that the SIS methodology exceeds the average accuracy of random input sparsity and approaches the maximum observed accuracy at a given level of sparsity in both cases.

5.4 Conclusion

The concept of input sparsity in SNNs for image classification problems has been introduced, and a Selective Input Sparsity (SIS) method has been proposed that can be used to potentially increase classification accuracy while reducing the size of the network before training. SNNs employing SIS is evidently well-suited to implementation in digital hardware.

One important note to make for this current chapter is that the goal of the work was not to propose a record accuracy solution to the discussed datasets, but rather to implement efficient hardware and maintain an accuracy that is similar to a baseline network. The adoption of various hidden layer sizes, further grid search for optimal random network weight initialization, and more training epochs could be used to improve the presented results potentially further.

Chapter 6

Spiking Neural Network Hardware Implementations

The information collected in the previous chapter was highly important for hardware development. The trained network weights and selected input and output layer spiking thresholds were applied to the digital hardware.

All implementations used a time step of discretization of $dt = 0.25$ and signed 8-bit integer datapaths. The 8-bit integer datapath is both sufficiently wide for accuracy in the implementation and small enough to maintain low hardware resource requirements.

Since there are ten output neurons, one corresponding to each input class, the output layer neuron with the highest spiking activity determines the image classification.

6.1 Hardware Blocks

6.1.1 Input Image Loading

Image input to the system is loaded one pixel at a time and each 8-bit pixel is loaded in parallel. The network then loads the presented pixel into a RAM. The loading stops when an end of data signal is triggered, indicating the end of the current image.

The loading process uses a handshaking protocol. The image loading controller gives a signal *valid in* to the network to indicate that valid data is ready to be presented to the network. When the network has completed an inference and is waiting for new data it gives the controller a signal called *new data* to indicate that it is ready to receive new data for inference. Following the first clock cycle where both signals are asserted high, the controller starts to present data to the network. The address of the current input pixel is also passed to the network as a redundant method of insuring synchronization.

In the case of the SIS network, a Look-Up Table (LUT) contains the indices of the retained pixels after the application of the SIS algorithm and points to the 0th element initially. As input pixels are presented to the network, the input loading mechanism compares the input pixel address to the address in the element to which the LUT points. If the input pixel address matches the current element in the SIS index, the pixel is added to the input RAM and the LUT pointer is incremented to the next element. A signal to indicate the end of the required data is triggered after pixel corresponding to the last element in the LUT has been loaded.

6.1.2 Output Layer Integrate and Fire Neurons

The network implementation uses signed 8-bit integer IF neurons for the output layer. The neuron design is simple and is essentially an accumulator with a spiking condition. Since spiking thresholds were selected as powers of 2, the spike condition can be checked through the logical AND of the inverse of the sign bit and the corresponding threshold bit. A synchronous reset was also implemented to reset the membrane potential to zero when asserted. This reset simplifies resetting the network to a resting state between input images.

Additionally, a clamp was introduced for when the membrane potential is less than or equal to -65. -65 was selected for the clamp because it can be implemented as the logical AND of the sign bit and inverted sixth bit. Additionally, since the time step was selected as $dt = 0.25$, the maximum magnitude of negative input current to this neuron, -63, after a shift right by two operation will not cause integer rollover. Therefore, this clamp was implemented to avoid integer rollover without the need for a larger word length in the system.

The behaviour of the output neuron can be described by:

$$v[n+1] = v[n] + dt * I[n] \tag{6.1}$$

$$if v[n] \geq v_{thres}, then \begin{cases} v[n+1] = 0 \\ spike \rightarrow 1 \end{cases} \tag{6.2}$$

$$if v[n] \leq v_{clamp}, then \begin{cases} v[n+1] = v_{clamp} \\ spike \rightarrow 0 \end{cases} \tag{6.3}$$

where v is the membrane potential of the neuron, I is the input current to the neuron, v_{thres} is the spiking threshold, and v_{clamp} is the membrane potential clamp

value described above. The input layer neurons are designed in the same way, except with an unsigned 8-bit datapath and therefore no negative clamp condition.

6.1.3 Weight Application at the Time of Spike Events

When an input spike event occurs, the event is attributed to the address of the pixel that caused the event. This address is then passed to read-only memory blocks containing the weights for each connection to the output neurons. A *weight enable* signal is then asserted high, and the corresponding weight is then applied as input current to the output layer neurons. Once the update has been completed, *weight enable* is set low so no unneeded updates occur.

6.1.4 Output Spike Counting

The activity of each output neuron is measured by the total number of spikes the neuron emits during exposure to an image. The spikes are counted using a simple unsigned synchronous counter with reset to zero where the count enable is the spike from the corresponding output neuron. The width of the counters was chosen to be 5-bit based on activity observations to prevent overflow. Once exposure to an image has ended, the spike tallies are compared using a simple comparison tree algorithm where comparisons occur in pairs until the counter with the highest value is selected as the network's classification.

6.2 Network Operation

Both a baseline Fully-Connected (FC) SNN and an SIS SNN were implemented so that an even comparison can be made between the proposed methodology and

TABLE 6.1: FPGA hardware implementation resource utilization and performance for 1000 test images compared to expectations from Python simulations for the MNIST digits classifcation network. Both hardware designs were implemented on the Kintex-7 FPGA.

	FC Digits Hardware	**SIS Digits Hardware**	**FC Digits Python**	**SIS Digits Python**
Inference Accuracy	93.4%	87.7%	95.1%	89.2%
Inference Time (ms)	109.96	36.26	24015.7	8120.3
Max. Clock Freq.	349.16MHz	267.38MHz	N/A	N/A
LUTs	1405	345		
LUTRAM	182	0		
FF	333	263		
BRAM	0	3.5		

baseline methods. The operation mechanism and network building blocks are consistent for both designs.

After the input image data has been loaded as described in subsection 6.1.1, the network cycles through the input pixels stored in RAM sequentially. During a cycle through the input pixels, the input pixel stored in RAM is loaded as well as a corresponding membrane potential for the pixel and sent as input to an input layer neuron. If a spike occurs, the neuron is reset, and the spike is sent to the output layer and the membrane potentials of the parallel output neurons are updated accordingly based on the pre-trained weights stored in ROM.

When the last pixel in the image has been reached, an "end of data" flag is asserted, and the pixel address counter is reset. Since, as mentioned in Table 5.1, the images were exposed for 16 time steps, this cycle occurs 16 times until the fourth bit of a counter is asserted high (corresponding to 16 cycles), meaning the exposure time has concluded. The spike counting procedure described in Subsection 6.1.4 then returns the network's classification of the input image.

6.3 Results

Table 6.1 shows FPGA hardware implementation results for the baseline and proposed SIS SNNs for the MNIST digits dataset. It is important to again note that the goal of the proposed hardware designs was not record accuracy on the presented datasets but rather to show the efficacy of the proposed SIS method against a baseline implementation, and to create efficient hardware for edge inference applications.

As is apparent from Table 6.1, the FPGA hardware implementations for both the FC network and SIS network outperform their corresponding Python SNN realizations in inference time by dramatic margins while maintaining consistent accuracy. The reported inference times also includes the required loading time to sequentially receive input image pixels from the loading mechanism, so this time metric is more inclusive and better models a real edge application.

Furthermore, the expected speed advantage is notable in the SIS network compared to the FC network. To increase the inference speed of the FC network, the Block RAM (BRAM) was converted to LUTRAM so that the maximum clock frequency could increase. Even with a clock frequency that is 29.5% higher than the SIS network, the SIS network still can perform inferences in less than one third of the time required by the FC network in the case of the MNIST digits dataset. Although the classification accuracy suffers to varying degrees depending on the dataset, given the speedup observed using the SIS method it is a valid contender for applications where speed is prioritized.

Chapter 7

Conclusion

7.1 A Summary of Conclusions

The overall goal of these works was novel contributions in neuromorphic engineering as well as in SNN-based AI applications.

Given the variety of the work proposed between chapters and sections, conclusions have been summarized for each section.

7.1.1 Spiking Neuron Implementations

Chapter 2 describes novel modifications to the Izhikevich neuron model through which the hardware resource requirements of a digital hardware implementation of the neuron are reduced and the spiking behaviour of the neuron can be manipulated using a single parameter to behaviourally reproduce all excitatory cortical neuron behaviours. The novel proposed HOMIN model was also shown to be effective in simple SNNs with synapses exhibiting STDP for associative learning. The HOMIN model would also be a good candidate for networks in which an Izhikevich neuron would be effective as the HOMIN neuron would require lower silicon

area per unit and less interconnection should the spiking behaviour of the neuron require flexibility.

Chapter 3 presents an FPGA implementation of the Hodgkin-Huxley neuron using the CORDIC algorithm for all non-linear terms, resulting in simplified hardware requirements compared to most other proposed implementations, very high accuracy, and flexibility in the physiological parameters of the neuron. The latter two successes make this design an excellent candidate for hardware acceleration of neuronal disease modeling and other biomedical applications where high parallelism to biological neurons is required.

Chapter 4 proposes a novel power-reduction methodology for biologically detailed digital spiking neurons that is highly inspired by the operation of real neurons. The sampling-based method introduces low-activity states to the system, which in turn reduces circuit switching activity by avoiding unnecessary computations and thus reduces power consumption. This method could be paired with neuromorphic chips for energy-efficient solutions in neuromorphic computing.

7.1.2 Spiking Neural Networks

Chapter 5 proposes a novel algorithm by which an SNN's size can be reduced before training based on properties of the training set images. This method dramatically reduces inference time (67% reduction in inference time for the MNIST digit dataset) with only minor accuracy impediments compared to baseline networks. Chapter 6 describes corresponding hardware implementations for the proposed SNNs. The hardware shows excellent speed improvements compared to software simulation.

7.2 Possible Future Works and Extensions

The proposed works have many possible extensions in various applications.

The work in Chapters 3 and 4 could be coupled with biologically detailed digital astrocytes and tripartite synapses to create an energy-efficient neuromorphic system for hardware-accelerated nervous system modeling. Work such as this could have many positive implications in biomedical applications.

The work in Chapters 5 and 6 could be extended to testing in edge applications and could be applied to many diverse image classification problems.

Bibliography

[1] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesus, *Neural Network Design, 2nd Ed.* Campus Pub. Service, University of Colorado Bookstore, 2002.

[2] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathem. Biophysics*, vol. 5, pp. 115–133, Dec. 1943.

[3] V. Gauck and D. Jaeger, "The control of rate and timing of spikes in the depp cerebellar nuclei by inhibition," *Journal of Neuroscience*, vol. 20, pp. 3006–3016, Apr. 2000.

[4] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J Physiol*, vol. 117, pp. 500–544, Aug. 1952.

[5] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophys. J.*, vol. 35, pp. 193–213, July 1981.

[6] E. M. Izhikevich, "Which model to use for cortical spiking neurons?," *IEEE Trans. Neural Netw.*, vol. 15, pp. 1063–1070, Sept. 2004.

[7] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, pp. 1569–1572, Nov. 2003.

[8] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *J Neurophysiol*, pp. 3637–3642, July 2005.

[9] H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophys. J.*, vol. 12, pp. 1–24, July 1972.

[10] L. Lapicque, "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation," *Journal de Physiologie et de Pathologie Générale*, vol. 9, pp. 620–635, 1907.

[11] Y. Dan and M.-M. Poo, "Spike timing-dependent plasticity: From synapse to perception," *Physiological Reviews*, vol. 86, pp. 1033–1048, July 2006.

[12] S. Lu and A. Sengupta, "Exploring the connection between binary and spiking neural networks," *Frontiers in Neuroscience*, vol. 14, p. 535, 2020.

[13] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[14] Y.-L. Chen, C.-C. Lu, K.-C. Juang, and K.-T. Tang, "Conversion of artificial neural network to spiking neural network for hardware implementation," in *2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, pp. 1–2, 2019.

[15] C. Frenkel, M. Lefebvre, J. D. Legat, and D. Bol, "A 0.086-mm^2 12.7-pJ/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 145–158, 2019.

[16] E. D'Angelo, S. Solinas, J. Garrido, C. Casellato, A. Pedrocchi, J. Mapelli, D. Gandolfi, and F. Prestori, "Realistic modeling of neurons and networks: towards brain simulation," *Functional Neurology*, vol. 3, pp. 153–166, 2013.

[17] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, vol. 8, pp. 225134–225180, 2020.

[18] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, "Fast classification using sparsely active spiking networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2017.

[19] D. Lisitsa and A. A. Zhilenkov, "Prospects for the development and application of spiking neural networks," in *Proc. 2017 IEEE Conf. Russian Young Researchers in Electrical and Electronics Engineering*, (St. Petersburg, Russia), pp. 926–929, Feb. 2017.

[20] T. Levi, F. Khoyratee, S. Saïghi, and Y. Ikeuchi, "Digital implementation of Hodgkin-Huxley neuron model for neurological diseases studies," *Artif Life Robotics*, vol. 23, pp. 10–14, Mar. 2018.

[21] S. Davidson and S. B. Furber, "Comparison of artificial and spiking neural networks on digital hardware," *Frontiers in Neuroscience*, vol. 15, p. 345, 2021.

[22] A. Cassidy, S. Denham, P. Kanold, and A. Andreou, "FPGA based silicon spiking neural array," in *2007 IEEE Biomedical Circuits and Systems Conference*, (Montreal, Quebec, Canada), pp. 75–78, Nov. 2007.

[23] J. Han, Z. Li, W. Zheng, and Y. Zhang, "Hardware implementation of spiking neural networks on FPGA," *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479–486, 2020.

[24] E. Z. Farsa, A. Ahmadi, M. A. Maleki, M. Gholami, and H. N. Rad, "A low-cost high-speed neuromorphic hardware based on spiking neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 9, pp. 1582–1586, 2019.

[25] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. Rahimi Azghadi, "CORDIC-SNN: On-FPGA STDP learning with izhikevich neurons," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2651–2661, 2019.

[26] G. C. Qiao, S. G. Hu, J. J. Wang, C. M. Zhang, T. P. Chen, N. Ning, Q. Yu, and Y. Liu, "A neuromorphic-hardware oriented bio-plausible online-learning spiking neural network model," *IEEE Access*, vol. 7, pp. 71730–71740, 2019.

[27] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, p. 99, 2015.

[28] D. Saunders, D. Patel, H. Hazan, H. Siegelmann, and R. Kozma, "Locally connected spiking neural networks for unsupervised feature learning," *Neural Networks*, vol. 119, 08 2019.

[29] J. Volder, "The CORDIC computing technique," in *Proc. Western Joint Computer Conf.*, (San Francisco, CA, USA), pp. 257–261, Mar. 1959.

[30] E. M. Izhikevich, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. Cambridge, Massachusetts, America: The MIT Press, 2007.

[31] G. Indiveri, E. Chicca, and R. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE Trans. Neural Netw.*, vol. 17, pp. 211–221, Jan. 2006.

[32] G. Indiveri and S. Fusi, "Spike-based learning in VLSI networks of integrate-and-fire neurons," in *Proc. IEEE Int. Symp. Circuits Syst., 2007*, New Orleans, LA, USA, May 2007, pp. 3371–3374.

[33] J. Shamsi, K. Mohammadi, and S. B. Shokouhi, "A low power circuit of a leaky integrate and fire neuron with global reset," in *Proc. 25th Iranian Conf. on Elect. Engineering, 2017*, Tehran, Iran, May 2017, pp. 366–369.

[34] S. Haghiri, A. Zahedi, A. Naderi, and A. Ahmadi, "Multiplierless implementation of noisy izhikevich neuron with low cost digital design," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, pp. 1422–1430, Dec. 2018.

[35] T. Hishiki and H. Torikai, "A novel rotate-and-fire digital spiking neuron and its neuron-like bifurcations and responses," *IEEE Trans. Neural Netw.*, vol. 22, pp. 752–767, May 2011.

[36] S. Gomar and A. Ahmadi, "Digital multiplierless implementation of biological adaptive-exponential neuron model," *IEEE Trans. Circuits Syst. I*, vol. 61, pp. 1206–1219, Apr. 2014.

[37] M. Liu, L. R. Everson, and C. H. Kim, "A scalable time-based integrate-and-fire neuromorphic core wth brain-inspired leak and local lateral inhibition capabilities," in *Proc. IEEE Custom Integrated Circuits Conf., 2017*, Austin, TX, USA, Apr. 2017.

[38] S. Sato, H. Akima, K. Nakajima, and M. Sakuraba, "Izhikevich neuron circuit using stochastic logic," *Electron. Lett.*, vol. 50, pp. 1795–1797, Nov. 2014.

[39] E. Chicca, G. Indiveri, and R. J. Douglas, "An event-based VLSI network of integrate-and-fire neurons," in *Proc. IEEE Int. Symp. Circuits Syst., 2004*, Vancouver, BC, Canada, May 2004, pp. 357–360.

[40] X. Wu, V. Saxena, K. Zhu, and S. Balagopal, "A CMOS spiking neuron for brain-inspired neural networks with resistive synapses and in situ learning," *IEEE Trans. Circuits Syst. II*, vol. 62, pp. 1088–1092, Nov. 2015.

[41] H. Amin, "Spiking neural networks learning, applications, and analysis," Ph.D. dissertation, 09 2006.

[42] B. U. Pedroni, S. Das, J. V. Arthur, P. A. Merolla, B. L. Jackson, D. S. Modha, K. Kreutz-Delgado, and G. Cauwenberghs, "Mapping generative models onto

a network of digital spiking neurons," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 4, pp. 837–854, 2016.

[43] F. Shama, S. Haghiri, and M. A. Imani, "FPGA realization of Hodgkin-Huxley neuronal model," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 15, no. 5, pp. 1059–1068, Mar. 2020.

[44] S. Haghiri, A. Naderi, B. Ghanbari, and A. Ahmadi, "High speed and low digital resources implementation of Hodgkin-Huxley neuronal model using base-2 functions," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 275–287, 2021.

[45] K. Akbarzadeh-Sherbaf, B. Abdoli, S. Safari, and A.-H. Vahabie, "A scalable FPGA architecture for randomly connected networks of Hodgkin-Huxley neurons," *Frontiers in Neuroscience*, vol. 12, p. 698, 2018. [Online]. Available: `https://www.frontiersin.org/article/10.3389/fnins.2018.00698`

[46] F. Khoyratee, F. Grassia, S. Saïghi, and T. Levi, "Optimized real-time biomimetic neural network on FPGA for bio-hybridization," *Frontiers in Neuroscience*, vol. 13, p. 377, 2019. [Online]. Available: `https://www.frontiersin.org/article/10.3389/fnins.2019.00377`

[47] S. Y. Bonabi, H. Asgharian, S. Safari, and M. N. Ahmadabadi, "FPGA implementation of the Hodgkin-Huxley neuron model," in *Proc. 4th Int. Joint Conf. Comp. Intelligence*, Barcelona, Spain, Oct. 2012, pp. 522–528.

[48] T. Yu and G. Cauwenberghs, "Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 4, no. 3, pp. 139–148, 2010.

[49] A. Natarajan and J. Hasler, "Hodgkin–Huxley neuron and fpaa dynamics," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 4, pp. 918–926, 2018.

[50] Y. Zhang, J. Nunez-Yanez, J. McGeehan, E. Regan, and S. Kelly, "A biophysically accurate floating point somatic neuroprocessor," in *2009 International Conference on Field Programmable Logic and Applications*, 2009, pp. 26–31.

[51] S. Postnova, C. Finke, M. T. Huber, K. Voigt, and H. A. Braun, *Conductance-Based Models for the Evaluation of Brain Functions, Disorders, and Drug Effects*. Vienna: Springer Vienna, 2012, pp. 97–132.

[52] N. Shimada and H. Torikai, "A novel asynchronous cellular automaton multicompartment neuron model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 776–780, Aug 2015.

[53] J. Wu, Y. Zhan, Z. Peng, X. Ji, G. Yu, R. Zhao, and C. Wang, "Efficient design of spiking neural network with STDP learning based on fast CORDIC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 6, pp. 2522–2534, 2021.

[54] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. Rahimi Azghadi, "CORDIC-SNN: On-FPGA STDP learning with izhikevich neurons," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2651–2661, July 2019.

[55] M. Heidarpur, A. Ahmadi, and N. Kandalaft, "A digital implementation of 2d hindmarsh–rose neuron," *Nonlinear Dynamics*, vol. 89, no. 3, pp. 2259–2272, Aug 2017.

[56] J. R. Clay, D. Paydarfar, and D. B. Forger, "A simple modification of the Hodgkin and Huxley equations explains type 3 excitability in squid giant axons." *Journal of the Royal Society Interface*, vol. 5(29), pp. 1421–1428, Dec. 2008.

[57] R. B. Wells, *Introduction to Biological Signal Processing and Computational Neuroscience*, Idaho, USA, 2010.

[58] S. Kirigeeganage, D. Jackson, J. M. Zurada, and J. Naber, "Modeling the bursting behavior of the Hodgkin-Huxley neurons using genetic algorithm based parameter search," in *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2018, pp. 470–475.

[59] Z. Yi, W. Fu, and L. Cao, "Implementation of Hodgkin-Huxley spiking neuron model using FPGA," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 2019, pp. 1535–1539.

[60] B. Namer, K. Ørstavik, R. Schmidt, N. Mair, I. P. Kleggetveit, M. Zeidler, T. Martha, E. Jorum, M. Schmelz, T. Kalpachidou, M. Kress, and M. Langeslag, "Changes in ionic conductance signature of nociceptive neurons underlying fabry disease phenotype," *Frontiers in Neurology*, vol. 8, p. 335, 2017. [Online]. Available: `https://www.frontiersin.org/article/10.3389/fneur.2017.00335`

[61] S. Shoham, D. O'Connor, and R. Segev, "How silent is the brain: is tthere a 'dark matter' problem in neuroscience," *J Comp Physiol A*, vol. 192(8), pp. 777–784, Aug. 2006.

[62] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010, sensory systems. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0959438810000450`

[63] R. C. Vergara, S. Jaramillo-Riveri, A. Luarte, C. Moënne-Loccoz, R. Fuentes, A. Couve, and P. E. Maladonado, "The energy homeostasis principle: Neuronal energy regulation drives local network dynamics generating behavior," *Frontiers in Computational Neuroscience*, vol. 13, 49, pp. 777–784, Jul. 2019.

[64] Y. Kuang, X. Cui, Y. Zhong, K. Liu, C. Zou, Z. Dai, Y. Wang, D. Yu, and R. Huang, "A 64k-neuron 64m-1b-synapse 2.64pJ/sop neuromorphic chip

with all memory on chip for spike-based models in 65nm CMOS," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.

[65] S. Y. Woo, D. Kwon, N. Choi, W. M. Kang, Y. T. Seo, M. K. Park, J. H. Bae, B. G. Park, and J. H. Lee, "Low-power and high-density neuron device for simultaneous processing of excitatory and inhibitory signals in neuromorphic systems," *IEEE Access*, vol. 8, pp. 202 639–202 647, 2020.

[66] J. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, 2011, pp. 1–4.

[67] M. Bensimon, S. Greenberg, Y. Ben-Shimol, and M. Haiut, "A new digital low power spiking neuron," *International Journal of Future Computer and Communications*, vol. 8, no. 1, Mar. 2019.

[68] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy, "A 4096-neuron 1m-synapse 3.8-pJ/sop spiking neural network with on-chip STDP learning and sparse weights in 10-nm finfet CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 992–1002, 2019.

[69] A. Elnabawy, H. Abdelmohsen, M. Moustafa, M. Elbediwy, A. Helmy, and H. Mostafa, "A low power CORDIC-based hardware implementation of izhikevich neuron model," in *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*, 2018, pp. 130–133.

[70] A. Rubino, M. Payvand, and G. Indiveri, "Ultra-low power silicon neuron circuit for extreme-edge neuromorphic intelligence," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 458–461.

[71] D. Chatterjee and A. Kottantharayil, "A CMOS compatible bulk finfet-based ultra low energy leaky integrate and fire neuron for spiking neural networks," *IEEE Electron Device Letters*, vol. 40, no. 8, pp. 1301–1304, 2019.

[72] A. S. Cassidy, J. Georgiou, and A. G. Andreou, "Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization," *Neural Networks*, vol. 45, pp. 4–26, 2013, neuromorphic Engineering: From Neural Systems to Brain-Like Engineered Systems. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0893608013001597`

[73] S. Xiao, W. Liu, Y. Guo, and Z. Yu, "Low-cost adaptive exponential integrate-and-fire neuron using stochastic computing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 5, pp. 942–950, 2020.

[74] S. Haghiri and A. Ahmadi, "A novel digital realization of adex neuron model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 8, pp. 1444–1448, 2020.

[75] M. Heidarpur, A. Ahmadi, and M. Ahmadi, "Time step impact on performance and accuracy of izhikevich neuron: Software simulation and hardware implementation," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[76] *Vivado Design Suite User Guide: Power Analysis and Optimization*, UG907 (v2020.1), Xilinx, Jun. 2020.

[77] V. L. T. I. Inc., "Technology and design challenges of mos VLSI," *IEEE J. Solid-State Circuits*, vol. SC-17, no. 3, pp. 442–448, Jun. 1982.

[78] J. Ferreira, G. Callou, A. Josua, and P. Maciel, "Estimating the environmental impact of data centers," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018, pp. 1–4.

[79] L. A. Camuñas-Mesa, B. Linares-Barranco, and T. Serrano-Gotarredona, "Low-power hardware implementation of SNN with decision block for recognition tasks," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 73–76.

[80] C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 3, pp. 384–394, 2019.

[81] V. Saxena, "A mixed-signal convolutional neural network using hybrid CMOS-rram circuits," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[82] A. Zhang, X. Li, Y. Gao, and Y. Niu, "Event-driven intrinsic plasticity for spiking convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2021.

[83] J. Han, Z. Li, W. Zheng, and Y. Zhang, "Hardware implementation of spiking neural networks on FPGA," *Tsinghua Science and Technology*, vol. 25, no. 4, pp. 479–486, 2020.

[84] M. Liang, J. Zhang, and H. Chen, "A 1.13μj/classification spiking neural network accelerator with a single-spike neuron model and sparse weights," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[85] Z. Zhao, Y. Wang, C. Li, X. Cui, and R. Huang, "A sparse event-driven unsupervised learning network with adaptive exponential integrate-and-fire model," in *2019 International Conference on IC Design and Technology (ICICDT)*, 2019, pp. 1–4.

[86] V. Kůrková and M. Sanguineti, "Classification by sparse neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2746–2754, 2019.

[87] W. Luo, "Improving neural network with uniform sparse connectivity," *IEEE Access*, vol. 8, pp. 215 705–215 715, 2020.

[88] Y. LeCun, C. Cortes, and C. J. Burges. (1998) The mnist database of handwritten digits. [Online]. Available: `http://yann.lecun.com/exdb/mnist/`

[89] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[90] P. Simard, D. Steinkraus, and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis." 01 2003, pp. 958–962.

[91] "Tf.keras.losses.sparsecategoricalcrossentropy: tensorflow core v2.6.0." [Online]. Available: `https://www.tensorflow.org/apidocs/python/tf/keras/losses/SparseCategoricalCrossentropy`

[92] P. Panda and K. Roy, "Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 299–306.

[93] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Networks*, vol. 111, pp. 47–63, 2019. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0893608018303332`

Vita Auctoris

Alexander Leigh was born in Windsor, Ontario in 1996. He graduated from Holy Names High School in 2014 and subsequently obtained a Bachelor of Applied Science in Electrical Engineering at the University of Windsor in 2018. He is currently a PhD candidate at the University of Windsor.