University of Genoa

Master's Program in Bioengineering

Thesis submitted in partial fulfillment of the requirements for the title of Master of Bioengineering

PQN model for Silicon Neuronal Network on FPGA: two designs for close-to-biology applications

Giuseppe Leo

December 2023

Thesis advisor: Prof. Michela Chiappalone

Thesis advisor: Prof. Takashi Kohno

Thesis advisor: Dr. Takuya Nanami

A nonno Raffaele

Summary

Even in the tiniest animal brains, there's a fascinating complexity at work, governing their bodies and actions. Despite being simple, these brains showcase impressive computational skills that are being unraveled. These processing capabilities operate with significantly lower energy consumption compared to computers. The study of the brain and its characteristics offers valuable insights into a novel and highly efficient form of computation that, from the biological world, can be exported to the artificial one.
By adopting this approach, the last decades saw the development of diverse brain-inspired systems. These technologies aim to achieve the flexibility and computational efficiency of biological neural processing systems, characterised by the implementation of neural architectures that tightly integrate processing and memory and the use of a spike-based encoding of the information.
Borrowing these innovative features from biology, neuromorphic engineering represents a transformative approach to computing. Its goal is to create novel, energy-efficient technologies while actively seeking to comprehend, through practical implementation, the remarkable capabilities of the brain.

In 2023, the Laboratory for Neuromimetics Systems at the Institute of Industrial Science, University of Tokyo, introduced the Piecewise Quadratic Neuron (PQN) model. Based on a qualitative modelling approach, it outperforms tradi-

tional conductance-based models in terms of resource utilisation and power consumption while mimicking their capability of accurately modelling different neural classes. Designed for an efficient digital implementation on a Field Programmable Gate Array (FPGA), it represents a valid tool for constructing large-scale Silicon Neural Network (SiNN).
These SiNNs are useful for cognitive computing and computational neuroscience applications. Furthermore, their capability of mimicking the brain functioning at a cell level, utilising low power, made them a promising instrument for the development of innovative neuroprosthetic devices for the treatment of brain disorders.

The objective of this thesis is to expand the PQN hardware design, which presents eight different configurations, each one reproducing a distinct neuronal class, with two innovative designs. The first new configuration consists of integrating some of these designs to create a population composed of four electrophysiological classes of cortical and thalamic neurons. In the next one, cell-to-cell variability within each class is implemented by providing each neuron with a heterogeneous set of parameters.
Both projects share the goal of enhancing the resemblance of the SiNN to their biological counterpart. In particular, the modelling of two important biological features, such as diversity in the types of neuronal classes and intraclass variability, represents a step forward in the realisation of a thalamocortical microcircuit.

A Xilinx Artix-7 XC7A35T FPGA, integrated within a Digilent cmod A7 board, was used for this project. All the source codes are openly accessible on GitHub[1].

[1] `https://github.com/gle0/PQN_thesis`

Contents

List of Figures

Chapter 1

Introduction

The field of neuroscience is relatively new. Unlike other subjects that have been studied for centuries, the exploration of the brain began less than two centuries ago [1]. In fact, the brain remains shrouded in mystery, particularly regarding the information processing mechanisms. Modelling and studying the brain can aid in unravelling these mysteries and offer insights into neural architecture and information processing. Comprehending these mechanisms not only enables the harnessing of their potential for the creation of innovative bio-mimetic technologies, but also aids in gaining a deeper understanding through modelling.

In the late 1960s, it was clear that certain limitations would impede the progress of electronic device technology [2], and Moore's Law [3] was anticipated to reach its limits. Unfortunately, standard integrated circuit fabrication scaling processes are approaching fundamental physical constraints [4]. Thus, there is the need for an entirely new class of devices free from these limitations [5]. One promising avenue involves exploring brain-inspired computing architectures with the aim of replicating the flexibility and computational efficiency observed in biological neural processing systems [6].

Moreover, the costs and energy consumption of Artificial Intelligence (AI) technologies are escalating exponentially [7]. Currently, AI technologies are characterised by an inefficient structure that demands significant power. This form of machine learning with a loosely brain-inspired architecture, might benefit from a shift towards a neuromorphic structure to address the power consumption issue [8].

The term "neuromorphic" was coined by Caltech Professor Carver Mead, who observed that the MOS transistor operating in the subthreshold regime could easily replicate the dynamics of brain ion channels [9]. He was convinced that "we have something fundamental to learn from the brain about a new and much more effective form of computation. Even the simplest brains of the simplest animals are awesome computational instruments. They do computations we do not know how to do, in ways we do not understand" [5].
For example, although the honeybee's brain consists of about one million neurons with a power budget of only 10 μW, it can perform real-time navigation and complex pattern recognition, adapting to an environment in continuous evolution [10].

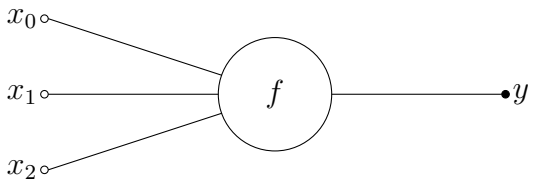
Neuromorphic engineering uses notions from the fields of neuroscience, electrical engineering, and informatics to study biomimetic systems. From a biological point of view, vision takes place when the retina converts photons into spikes. The smell is just volatilised molecules converted into spikes, like the sense of touch is the conversion of tactile pressure. All sensory inputs are thus encoded into spikes. This allows for event-driven processing and low-precision computation.
Another attribute of biological neural processing systems includes the tight collocation of computation and memory. Therefore, the aim of neuromorphic technologies is the realisation of two paramount innovations in their systems. Contrary to traditional von Neumann processor architectures (computers) that separate processing

(Central Processing Unit (CPU)) and memory (RAM), the brain's organising principles are based on distributed computation, aligning processing and memory with neurons and synapses [11]. This new disposition has the potential to lower wiring, data transfer costs and energy supplied. Secondly, conventional processors encode data as multi-bit words processed sequentially under a global clock. Time is thus a by-product of computation. Conversely, the brain processes information by encoding data both in space and time with binary (all-or-none) events, where "time represents itself".

Adopting a similar approach is advantageous for neuromorphic hardware for several reasons. A spike can be effectively approximated using a single bit (0 or 1), simplifying hardware representation compared to high-precision values. Another significant benefit is the inherent sparsity of neural activity; neurons are mostly silent. Leveraging this property proves cost-effective, as storing 0 values requires fewer resources. Additionally, the event-driven processing paradigm involves handling information only when there is new data, avoiding the need for processing at every time step. This not only enhances power efficiency but also results in faster processing speeds, enabling operation in biological or even accelerated time.

The building block for realising this paradigm shift is thus the Spiking Neural Network (SNN), that will be soon described. It is also called the third generation of neural network [12].
In fact, neural networks have a long history, started in 1943 [13]. The term neural network is referred to graphs with a hierarchical structure of the nodes that resemble the cortical layers organization. Each node receives a series of inputs that are first integrated and then fed to a function that computes an output.
The first generation of neural network is the McCulloch-Pitts Neuron [14]. In this

model, the binary inputs are added and then passed to a function that outputs a binary value depending on whether the result of the sum is greater than a certain threshold. This model can accept both excitatory and inhibitory (negative) inputs, and it was mainly used to model logic gates. Figure 1.1 shows a 3-inputs AND gate.



$$if\ x_0 + x_1 + x_2 \geq 3,\ y = 1$$

Figure 1.1: *Description of a three inputs AND gate implemented using the McCulloch-Pitts model. All the inputs and the outputs are binary (0 or 1). The output is 1 only when the sum of the inputs is equal to 3. Otherwise, the output is 0.*

The second generation [15] marked the advent of Deep Neural Network (DNN), featuring the Artificial Neuron (AN) (Figure 1.2) as its fundamental component. Diverging from the McCulloch-Pitts model, these networks comprise new layers different from the input and output ones, called hidden layers. The AN comprises weighted continuous inputs that are aggregated and processed through an activation function, typically nonlinear. The resulting output from this function is continuous.

An essential aspect of these models lies in the weights associated with each node in the network. Among various methods for adjusting these weights (training the network) to make the network able to perform a particular task, the widely employed algorithm is backpropagation. Exploiting this model has led to diverse network types, such as Convolutional Neural Network (CNN), used for image recognition applications like Google Lens, and Recurrent Neural Network (RNN), applied in fields such as speech recognition and language modelling, exemplified by systems

like ChatGPT and Gemini.

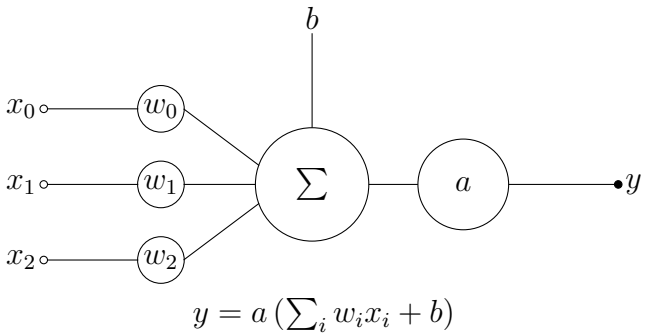


Figure 1.2: *An artificial neuron has continuous inputs and outputs. Each input is associated with a weight that determines the relevance of the input in the computation. The weighted inputs are summed together with a bias ($\sum$) and the result feed the activation function (a) that determines the output.*

In the end, the Spiking Neuron (SN) (Figure 1.3) is the building block of the third generation of neural networks. Here, every neuron receives different synaptic binary inputs, and it outputs spikes based on its internal dynamic. This dynamic can be modelled in different ways and with different levels of detail and will be described in Chapter 3. The response of an SN depends on its past inputs, while an AN responds only to its instantaneous inputs. This key feature allows for the temporal encoding of information.

The PQN model [16] is one way of modelling the internal dynamics of the neurons using a qualitative approach. It ensures higher biological accuracy while maintaining more compact and lower power consumption than more bio-plausible alternatives, like conductance-based models. Remarkably versatile, the PQN model can represent diverse neuron classes and prioritises efficient implementation through digital arithmetic circuits. These characteristics make it a low-power tool for implementing SiNN on FPGA chips.

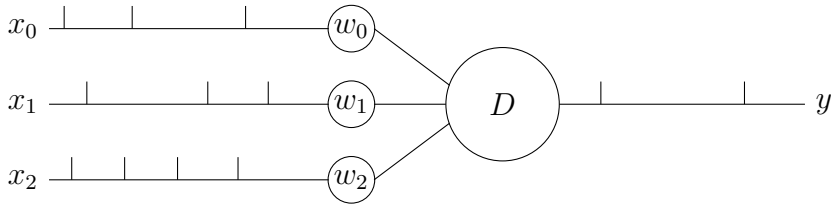


Figure 1.3: *The spiking neuron consists in a series of binary inputs (spikes) each one associated with a weight. These synaptic inputs alter the neural dynamic (D) that produces a binary output. The neural dynamic is a set of continuous equations that simulate the behaviour of the membrane potentialm of a neuron. The detailed discussion on this topic is delayed to Chapter*

Indeed, the PQN model aims to achieve exceptional AI information processing performance with minimal power consumption, allowing it to mimic biological systems closely [17].

The applications of these devices are incredibly broad. A multitude of event-based sensors has already been developed. The event-based paradigm also applies to control. Indeed, various methods have been proposed to achieve robot locomotion.

Furthermore, the recent availability of extensive anatomical and physiological data has led to the development of data-driven SNN models [18] with the ambitious goal of replicating the mammalian brain.

Increasing the level of biological accuracy, these models are thus suited for artificially recreating brain regions, also with a high level of detail. Thanks to these attributes and the real-time simulation speed of SiNN, they can also play a crucial role in the treatment of neurological disorders.

Currently, one of the most significant challenges in neuroscience revolves around finding effective treatments for numerous chronic and incurable brain conditions.

Worldwide, over one billion people have diseases and injuries affecting the brain, and existing treatments fail to address all symptoms while providing no guarantee of complete functional recovery. Among the myriad brain issues, stroke and Traumatic Brain Injury (TBI) stand out as they disrupt connectivity within the brain. Importantly, many of the neurochemical processes leading to cell death and disconnection also contribute to heightened susceptibility to plastic changes. Current therapeutic approaches attempt to leverage this plasticity through pharmacological treatments, which are still not highly effective, and robotic aids, which are both expensive and inadequately accessible [19]. To advance plasticity recovery, innovative biological and engineering approaches geared towards brain rewiring are under exploration. This encompasses novel forms of neuroprosthetics [20] capable of artificially reconnecting disconnected neuronal modules and substituting the activity of one module with real-time neuromorphic hardware [21], [22].

The work of [23] established interactions between artificial and living neurons for the first time. Subsequently, the field witnessed the emergence of "hybrid neuromorphic engineering", a concept that integrates technological devices with biological neurons [24]. This interdisciplinary approach enables real-time interaction between artificial and biological neurons [25] facilitating learning adaptation, miniaturisation, and improved power consumption efficiency. SiNN-based neuroprostheses have demonstrated remarkable effectiveness in experiments focused on restoring locomotion in isolated spinal cords of rats subjected to complete transection [26].

The objective of these new devices is to establish a closed-loop system [27], connecting the brain portion with neuromorphic hardware. Following the recording of neural activity with a Microelectrode Array (MEA), a signal processing com-

ponent endeavours to identify patterns within the recorded activity and, based on these patterns, deliver stimulation to the cells. This process can be executed in two ways, as illustrated in figure 1.4. A critical aspect of this procedure is the signal processing phase, during which the recorded signal from the MEA is analysed to extract spike events. Spike detection algorithms are typically employed for this task, but their lack of a universal method and absence of a definitive criterion pose challenges. In addressing these limitations, SiNN have been integrated into the signal processing stage instead of traditional spike detection algorithms. This choice is motivated by the SiNNs' capability to operate in real-time, implement bio-inspired processing techniques, and meet the growing demand for low-power and low-latency platforms. Notably, in the study by [28], the SpiNNaker platform was utilised to implement a novel method for burst detection, yielding highly favourable results.

The primary aim of this thesis is to extend the functionalities of the PQN model through the development of two new hardware designs to progress toward the realisation of an artificial thalamocortical system in a single hardware.
The first design encompasses the four predominant electrophysiological classes of cortical and thalamic neurons — namely, Fast Spiking (FS), Regular Spiking (RS), Intrinsically Bursting (IB), and Low Threshold Spiking (LTS) cells — integrated into a single chip.
In constructing thalamocortical networks, it is crucial to incorporate not only different classes of intrinsic properties but also the heterogeneity within each class [29]. Notably, even neurons of the same type exhibit substantial variances in their properties. This variability contributes to the richness of neuronal processes, with the reliable functional behaviour of neural systems that depends critically on such diversity [30]. For this reason, the second design introduces cell-to-cell variabil-

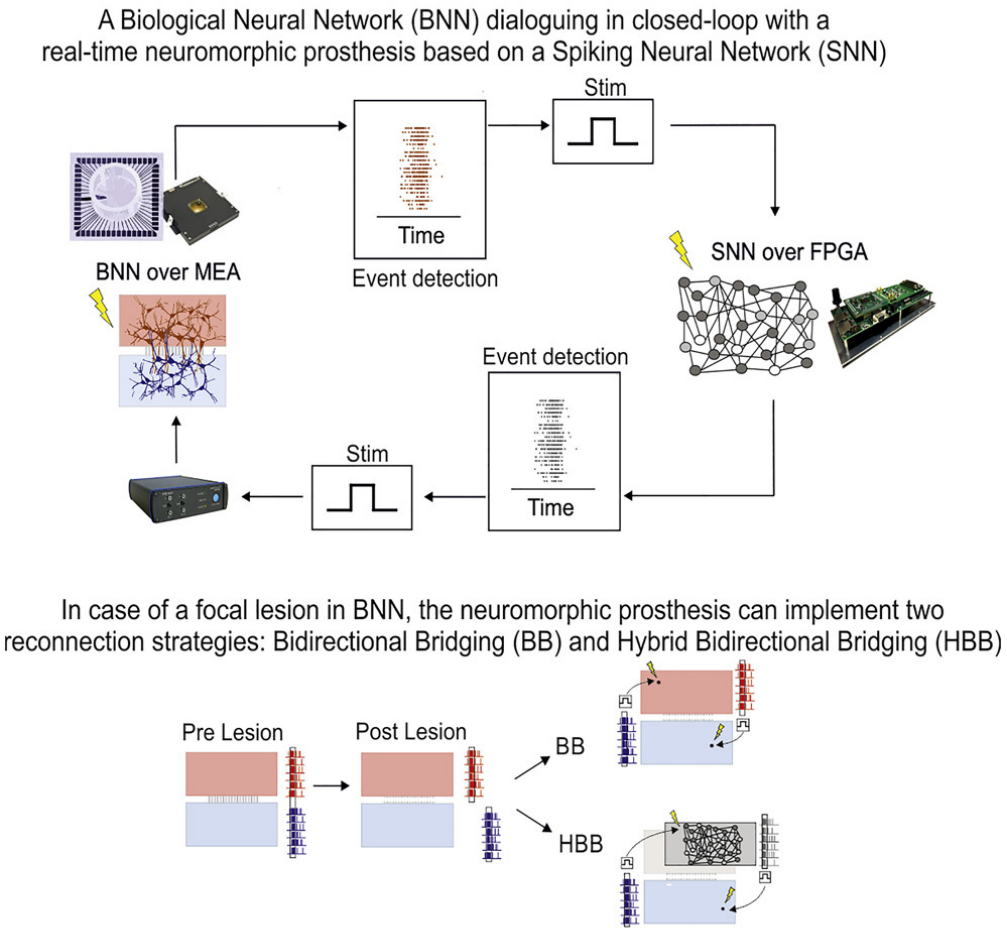


Figure 1.4: *Figure taken from [22]. (Top) The closed-loop system is depicted, wherein a BNN on a MEA and a SNN on a FPGA interact. The system initiates by recording activity from the biological population and extracting spikes through an event detection procedure. These extracted spikes then serve as input to the SNN, and the resulting output from the FPGA is transmitted back to the BNN. (Bottom) Two potential applications of this protocol are illustrated. In the first scenario, the SNN receives input spikes from the initial biological population and generates output spikes on a second population. In the second case, the SNN interfaces directly with a single population.*

ity within each class, achieved by assigning specific parameters to each individual neuron.

The thesis is structured as follows:

- Chapter 2 provides a concise overview of digital logic systems, with a focus on programmable logic and FPGA.
- Chapter 3 introduces the computational neuron models most used in neuromorphic hardware, specifically focusing on the PQN model.
- Chapter 4 elucidates the characteristics of various neuromorphic hardware, emphasising the one proposed by [16], which serves as the foundational design for this project.
- Chapter 5 explains the two new hardware designs developed within this project.
- Chapter 6 summarises applications and future perspectives.

Chapter 2

Digital Logic and FPGA

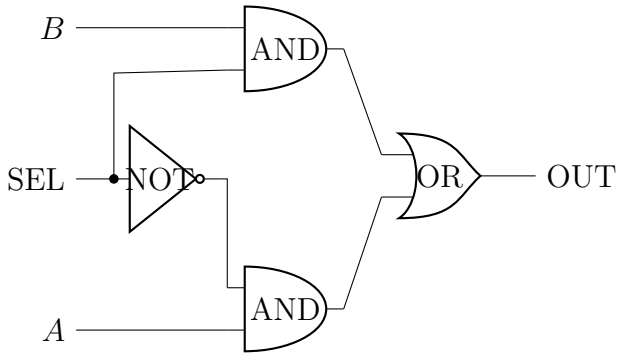
Although information is a disembodied thing, in order to process it, it's necessary to give it a physical form [31]. Any physical quantity that can be easily manipulated can be used to represent information. Electrical current, air or fluid pressure, and physical position have been used to build processing systems. However, electric signals became universally predominant due to the ease and cost-effectiveness of fabricating intricate systems through Complementary Metal-Oxide-Semiconductor (CMOS) integrated circuits.

Electric signals can be analog or digital, both are pivotal in transmitting the information. Analog circuits translate information into electric pulses with varying amplitudes, embodying a continuous spectrum, whereas digital technology employs a binary format (zero or one), with each bit symbolising two discrete amplitudes (minimum and maximum). The prevalence of digital systems can be attributed to their ability to process, transport, and store information without succumbing to distortion induced by noise. This is made possible by the discrete nature inherent to digital data.

2.1 Logic Gates

To process and manipulate binary information, logical primitives are essential — very simple operations on one or more binary inputs. AND, OR, and NOT are the elementary ones. By combining these primitives, we can create fundamental modules such as decoders, multiplexers, encoders, etc., which serve as the foundational components for every digital circuit.

A 2-bit multiplexer is just two AND gates, one NOT gate, and one OR gate and a series of wires. This module (Figure 2.1) has two 1-bit inputs and, based on the value of the selector (SEL) bit, outputs one of the two inputs.



(a)

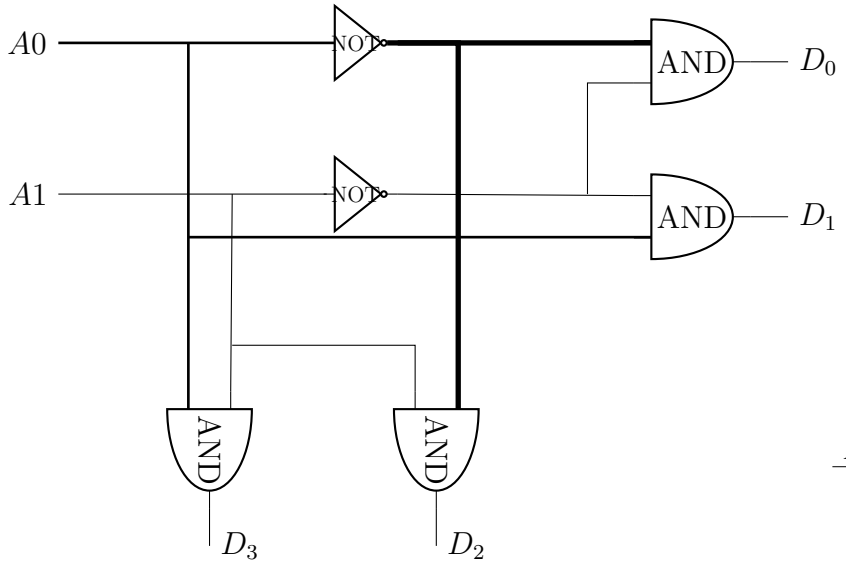
SEL	A	B	OUT
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(b)

Figure 2.1: *(a) One possible multiplexer's schematics; (b) multiplexer's truth table: the output is A when SEL is 0 and B when SEL is 1.*

A 2-bit decoder (Figure 2.2) can be constructed by employing two NOT gates and four AND gates. It takes a binary input and transforms it into a one-hot vector. It's a vector with all its bits set to 0 except for one.

Although these components can appear too simple, they work as the foundation of the whole digital logic. Indeed, a decoder, in conjunction with multiple OR gates, enables the creation of Read-Only Memory (ROM), a module which



A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

(b)

Figure 2.2: *(a) One possible decoder's schematics; (b) decoder's truth table: a binary n-bit input i translated into a one-hot 2^n-bit vector.*

allows permanent information storage. A ROM can implement any arbitrary logic function by storing the corresponding truth table within its memory. By accepting an address as input, the ROM outputs the corresponding value stored in its table at that address. The term 'read-only' signifies that the values within the table are predetermined and hard-wired during the ROM's manufacturing process, rendering them immutable.

The previous elements are all part of the category of combinational logic, where the current state of the inputs solely determines the output of a module. In contrast, a circuit whose output relies on preceding input states is called a sequential circuit. For instance, in a majority circuit — a logic circuit that takes n inputs and outputs a 1 if at least half of the inputs are 1 — the output is only determined on the count of 1s in the current input state. Conversely, a circuit that yields 1 when the count of 1s in the inputs surpasses the count in the preceding input state

is classified as sequential.

It's clear that the sequential logic needs to account for time changes in the circuit. To do so, it's necessary to store bits of information dynamically. The latch (Figure 2.3) represents the simplest represents the simplest form of memory in sequential circuits. It's built using just two NOR gates (an OR gate with its input inverted). It is an asynchronous circuit, i.e., its output can change any time its input changes.

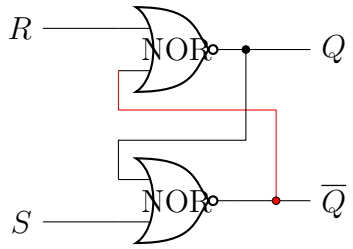


Figure 2.3: *The latch is composed of two inputs, Set (S) and Reset (R), one output (Q) and its inverted ($\overline{Q}$). The NOR gate outputs a 0 if at least one of its inputs is 1. So, starting with $S = 1$ and $R = 0$, $\overline{Q}$ becames 0 and Q 1, because the inputs of the top gate are both 0. Switching the Set to 0, the output Q remains 1, because it is an input for the bottom gate. In this way, the bit is successfully stored. To change the output Q it's necessary to change the R input to 1. In this way, the Q turns back to 0.*

The Flip Flop (FF) is a synchronous latch. The synchronous element is introduced through a clock control signal that allows changes in the output only at specific times. In fact, it is also known as a gated or clocked SR latch. This makes the FF the foundational component of sequential logic circuits. It's built by adding two AND gates to the latch.

A Delay Flip Flop (DFF) (or data flip-flop) (Figure 2.4) is more functional than the FF because it has only one input called data that works as the set input, while

its inverted value is the reset. Introducing an enable signal to the output of the DFF transforms creates a register. It works by coping its input to the output only when its enabling bit is high; otherwise, the output maintains its prior value.

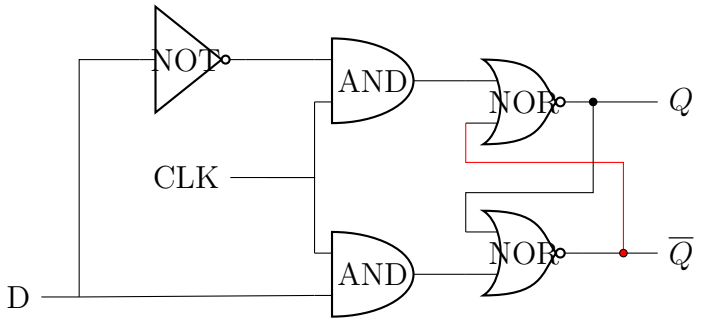


Figure 2.4: *The output Q reflects the D input only when the clock transitions from 0 to 1. From one rising edge to the other the module simply store the current value even if the data input change. When clock input is high, the data is transferred to the output of the FF and when the clock input is low, the output of the FF is held in its previous state.*

The essential storage element for any computing system, the Read-Write Memory (RWM), is built by exploiting a demultiplexer, a multiplexer, and several registers. Often referred to as Random Access Memory (RAM) for historical reasons, read–write memories share similarities with ROMs, and it has an additional feature consisting of the modification (writing) to the stored contents of the table. Unlike ROMs, where the data at each location is constant, in RAM, the stored data is retrieved from a register, making it dynamic and changeable [32].
Similarly to ROMs, RAMs need an address that specifies which storage element one wants to operate with. This address simply determines which register needs to change value. This is done by activating only its enable signal while inactivating all the others.

2.2 History of Programmable Logic Devices

The chips commonly encountered in daily interactions with technology usually operate on simple digital logic. These hardware components are designed to perform specific functions, with their wires and gates configured in a way that remains unalterable. However, there exists another category of devices known as Programmable Digital Logic.

A Programmable Logic Device (PLD) is an electronic component used for constructing reconfigurable digital circuits. Unlike digital logic built using fixed-function logic gates, a PLD has a more general organisation. It lacks a predefined function at the time of manufacturing, and it must be programmed before being integrated into a circuit.

The evolution of PLDs traces back to Simple PLDs, including Programmable Read-Only Memory (PROM) and Programmable Array Logic (PAL), progressing to Complex Programmable Logic Device (CPLD), culminating in the development of FPGA.

In 1956, the PROM was invented. Figure 2.5 shows the different notations between conventional and programmable gates.

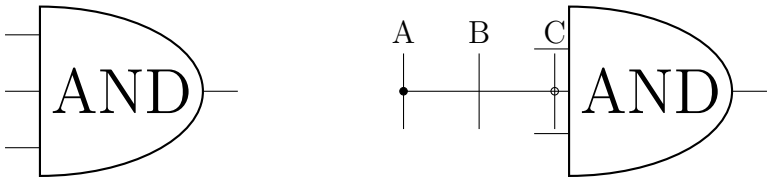


Figure 2.5: *(a) Standard logic notation; With a programmable logic notation (b) there can be three possible link types: hardwired (A), not connected (B) and programmable (C).*

As shown in 2.6, the PROM comprises a fixed AND plane, or *product terms*,

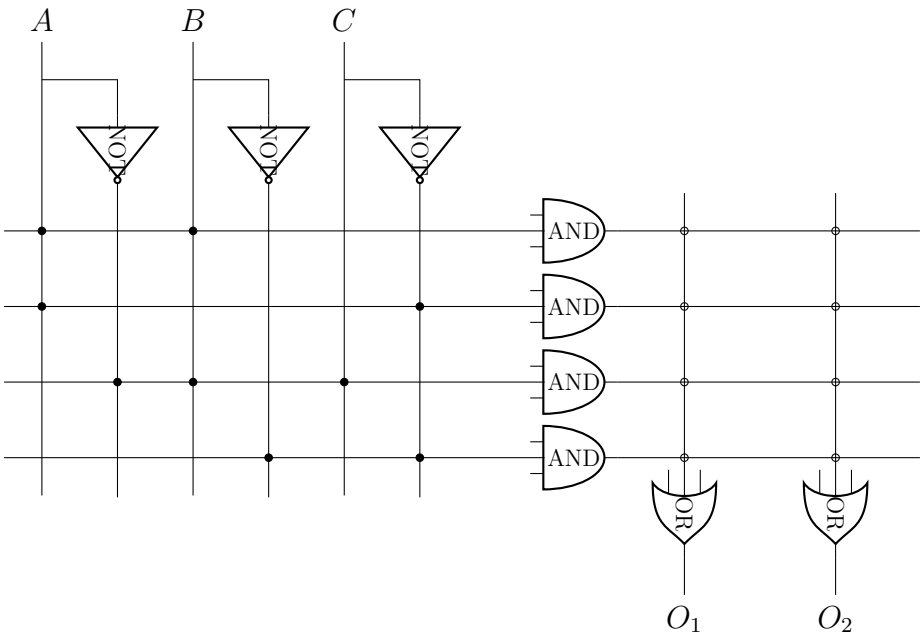


Figure 2.6: *The PROM structure has a fixed AND plane (left) and a programmable OR plane (right). This difference is expressed in the notation used for the links: hardwired or not connected in the first case and programmable in the second.*

followed by a programmable OR plane, where *sum terms* can be altered by modifying the memory contents. In the AND plane, links are either hardwired or left unconnected.

The working principle is straightforward, programming the device means determining which link will be activated in each column. To do so, there's the need to store an array of bits for any column of the OR plane.

For example, when aiming for the first output (O_1) to embody the function (A AND B) OR (NOT C AND A), the binary array 1100 should be stored in the relative column. This activates only the two first programmable links in the column, while the last two remain not connected.

After key technological advancements, the Erasable Programmable Read-Only Memory (EPROM) emerged in 1971, followed by the introduction of the Programmable Logic Array (PLA) in 1975 and the PAL in 1978. The PAL, resembling the PROM, differs in having a fixed OR plane and a programmable AND plane. This architecture demonstrates increased efficiency, as most of the logic functions of interest generally exhibit limited product terms [33].

The PAL is the foundational component of the CPLD. These new devices feature a distinct structure; internally, CPLDs typically incorporate multiple instances of a fundamental programmable Logic Element (LE) or macrocell, each roughly equivalent to a PAL. Each macrocell can implement a network of various logic gates, which feed into 1 or 2 acrshortffs. These LEs are organised in a column or matrix format on the chip. To facilitate more intricate operations, LE can be automatically interconnected with other LEs on the chip through a programmable interconnection network.
However, for designs requiring a substantial number of registers, data transfers, and bus interfaces, these logic-intensive but FF-limited devices proved less scalability. This limitation prompted the evolution of another PLD architecture, the FPGA [34].

2.3 Field Programmable Gate Array

The FPGA was conceived to offer a versatile digital logic device with significant flexibility and functionality. Rather than relying on transistor gates for logic implementation, FPGAs often utilise memory fashioned as lookup tables. Sharing a similar architecture with CPLDs, FPGAs feature smaller logic elements known as

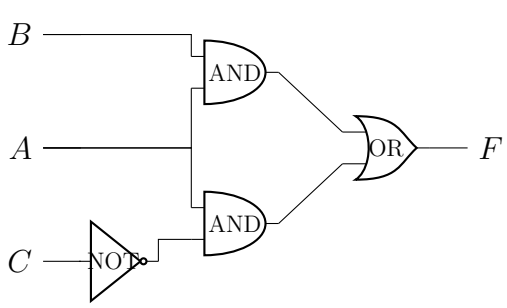
Complex Logic Block (CLB). These CLBs contain Look-Up Table (LUT) that can be programmed to execute specific logic functions [35].

An example showing how an LUT can model any gate network is shown in figure 2.7. First, the gate network is converted into a truth table. Given three inputs and one output, an 8-row truth table is generated. Subsequently, this truth table is loaded into the LUT during device programming. To do so, the 8 bits representing the output of the F function are stored in a particular type of RAM. The three inputs—A, B, and C—serve as address lines for the multiplexers, and the output F, is thus dependent on the values of these inputs. This process enables the LUT to execute the gate network through a RAM, eliminating the need for conventional logic gates.

FPGA compete with Application Specific Integrated Circuit (ASIC) and Application Specific Standard Product (ASSP). An ASSP is an integrated circuit tailored for a specific application market and is standardised for sale to multiple users. Instead, an ASIC is designed and manufactured for sale to a single company.

Due to their tightly controlled configurations, both exhibit compactness, cost-effectiveness, speed, and low power consumption. However, the fixed functionality at the time of manufacture limits their adaptability, as even a small section of the circuit cannot have its functionality altered.

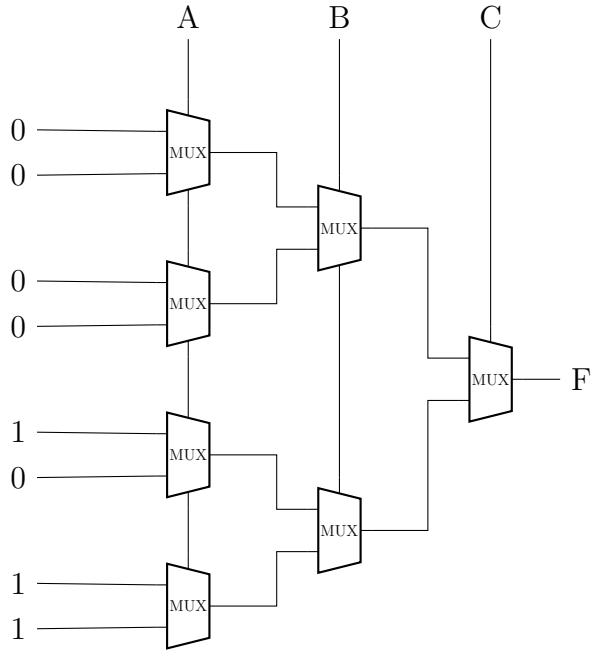
In contrast, FPGAs provide a dynamic solution by allowing adaptation to new standards and enabling hardware reconfiguration for specific applications even after being installed in the field—hence the term "field-programmable". This flexibility renders FPGAs a more versatile and cost-effective alternative to the rigid configurations of ASICs and ASSPs.



(a)

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

(b)



(c)

Figure 2.7: *(a) logic gate function; (b) truth table of the function; (c) LUT implementation of the function*

2.3.1 Programming FPGA: VHDL

VHDL and Verilog are the primary HDL for programming FPGAs. Originating in the 1980s under the U.S. Department of Defence, the Very High-Speed Integrated

Circuit (VHSIC) Hardware Description Language (VHDL) has evolved into an industry standard for delineating the behaviour and structure of digital systems. VHDL operates as an HDL, facilitating the modelling and simulation of digital systems across various levels of abstraction, from system-level down to logic gates [36].

While conventional computer languages describe algorithms (sequential execution), VHDL is specifically tailored to describe hardware (parallel execution). It's thus a concurrent language: the majority of VHDL instructions are executed all at the same time.

A VHDL design comprises a hierarchy of modules, encapsulating design structures that communicate through declared input, output, and bidirectional ports. Different modules are called *entities*, and their functions are described in the *architecture*. The *entity* construct in VHDL abstracts circuit functionality to a higher level, promoting modularity and abstraction.

Internally, an *architecture* can encompass signal/variable declarations, concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed only within a block called *process*.

VHDL primarily serves as a tool for simulation, allowing designers to build models that facilitate testing and verification before hardware implementation. Once the simulation yields the expected results, the subsequent step involves compiling the HDL design. In FPGA programming, a synthesis tool takes the HDL design as input and transforms it into a network of gates, registers, and wires configured to embody the desired functions. The implementation processes then adapt this network of components to the chosen FPGA model. Lastly, this information is enclosed in a programming (bitstream) file that is uploaded to the FPGA.

2.3.2 UART communication

The communication between the FPGA and the computer in this project is facilitated through the Universal Asynchronous Receiver-Transmitter (UART), a widely utilised device-to-device communication protocol with configurable data formats and transmission speeds. In UART, data bits are transmitted serially, one at a time, from the least significant to the most significant, encapsulated by start and stop bits. A second UART reconstructs the bits into complete bytes at the receiving end.

The UART protocol is employed within the project scope to transmit two data between the FPGA and the Personal Computer (PC), specifically the membrane potential (V) and input current (I). Both signals are represented by 18 bits and encoded into 3-byte data through UART communication. Packets containing this information are transmitted every 0.1 ms from the FPGA to the computer and every 0.1 s from the computer to the FPGA.

The Cmod A7-35T board, housing the Artix-7 FPGA, used in this project supports a maximum baud rate of 4 Mbits/s.

Chapter 3

Computational Neuron Models

In biology, the nervous system is the central processing and communication hub within an organism, facilitating the transmission of signals to and from various body parts to control, regulate, and coordinate its functions. It comprises two types of cells: neurons and glial cells. Glial cells play a crucial role in stabilisation, nutrition, and protection [37].
On the other hand, neurons are highly specialised cells responsible for generating and transmitting electrical signals to other cells. They consist of a cell body, or soma, which gives rise to two distinct ramifications: dendrites and axons. The dendrites receive stimuli from other neurons and convey them towards the cell body. At the same time, the axons transport the neural signal from the cell body to the synapses, sites specialised in transferring the signals to other cells [38].

For transmitting information, neurons utilise Action Potential (AP) (Figure 3.1), which are rapid fluctuations in their membrane potential that propagate along the neuronal membrane. They are generated by specific types of voltage-gated ion channels.
When a neuron receives a synaptic input, the membrane depolarises, resulting in

a rise in its potential. This permits an influx of sodium ions in the cell, which leads to a strong increase in the potential. This induces a reversal in the voltage polarity, followed by the closure of the sodium channels and the activation of the potassium channels, initiating an outward flow of potassium ions. This outward current restores the electrochemical gradient, returning the membrane potential to its resting state [38].

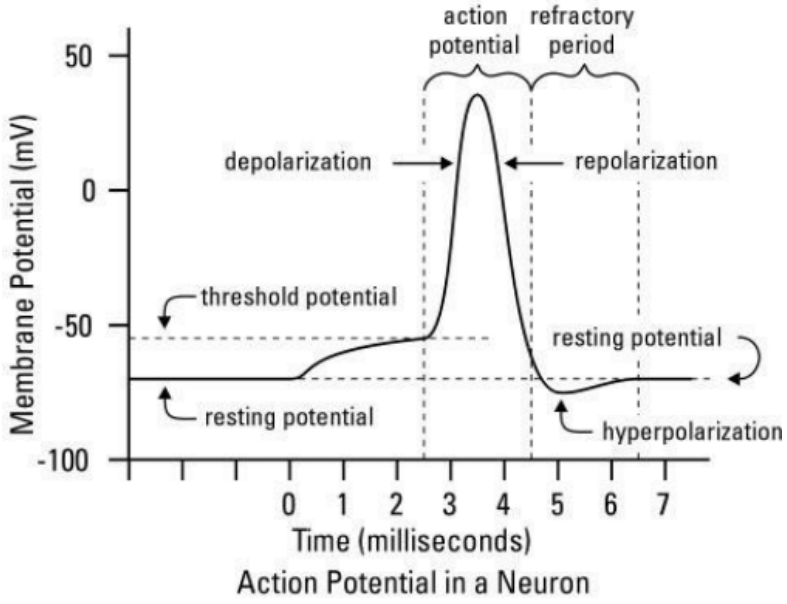


Figure 3.1: *Picture taken from [39]. With the arriving of input current, there's the accumulation of the inputs that increase the membrane potential. When it reaches a threshold, all the channels open, bringing to the actual depolarization, followed by the repolarisation. A spike happens. After that, there's a hyperpolarisation, and the neuron enters a resting period, in which it can't fire. The AP shape can be modelled with an exponential function.*

From a computational perspective, neurons, the key computational elements in biological systems, perform nonlinear transformations on their inputs in both spatial and temporal dimensions. As illustrated in figure 3.2, dendrites function as the input stage, the soma operates the computational processes, and the resulting

output is transmitted along the axon. It, in turn, connects to the dendrites of other neurons through synapses. In many neuromorphic systems, the soma is commonly referred to simply as the entire neuron.

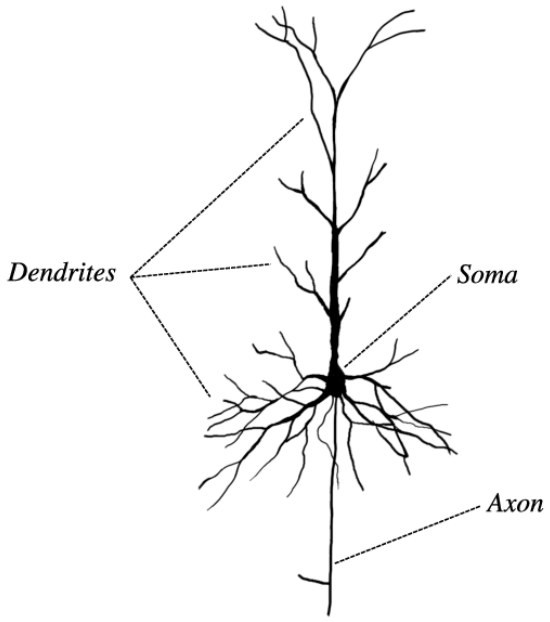


Figure 3.2: *Image taken from [40]. Neurons can have diverse morphologies. Here, a pyramidal neuron, the most plentiful type of neuron in the cortex, is illustrated. While presenting morphological differences, all the neurons share the same functional structures. The dendrites are short extensions that originate in different parts of the cell. They receive input stimuli from other neurons' synapses. These arrive at the soma, which is the cellular main body. Its form varies with the cell type and its dimensions range from 4-140 μm. It processes the inputs and produces an output that travels along the axon, a long branch that has many synapses at its end.*

Models of neural computation try to elucidate, in an abstract and mathematical manner, the fundamental principles that govern information processing in biological nervous systems [41], [42]. Neuron models are mathematical descriptions of how the voltage potential across the neuron membrane changes over time, with

the first such model dated 1907 [43].

Today, based on biological realism, neuron models can be broadly categorised into two classes: biophysical and abstracted neural models. Biophysical models aim for a detailed and realistic representation of the biological processes within neurons, while abstracted models simplify the representation to emphasise the essential aspects.

In the context of neuromorphic applications, the following lists the most used neuron models.

3.1 Neuron models for neuromorphic applications

3.1.1 Integrate-and-Fire models

In 1907, Louis Lapique introduced the Integrate-and-Fire (IF) model [44], [45], which belongs to the category of abstracted models. This simple model represents neurons using an electric circuit consisting solely of a capacitor. The capacitor models the neuronal membrane. When it is charged to a specific threshold potential, an AP is generated, with the subsequent capacitor discharge.

One of the most used IF models is the Leaky Integrate-and-Fire (LIF) model [46], which incorporates a leakage term (resistance) into the IF circuit (Figure 3.3). This accounts for voltage-independent channels that are responsible for the leakage of ions into and out of the cell, giving rise to a net leakage current I_{leak}. From a modelling perspective, the introduction of this new term enables the incorporation of time-dependent memory into the system.

The equations of the LIF model are 3.1 and 3.2

$$C\frac{dV}{dt} = -g_L\left(v - E_L\right) + I(t) \tag{3.1}$$

$$v = v_{reset} \quad if\ v > v_{th} \tag{3.2}$$

In 3.1 $I(t)$ is the synaptic or input current at time t, C the membrane capacitance, g_L the membrane conductance and E_L the resting membrane potential.

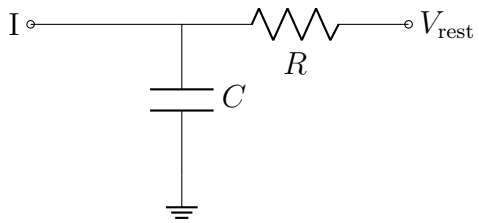


Figure 3.3: *The subthreshold regime of the model is described by the RC circuit dynamics. When a current is applied across the circuit, the capacitor starts to charge. During the charging phase, the voltage across the capacitor (V_c) increases over time. The rate at which the capacitor charges is determined by the time constant ($\tau = RC$) of the circuit. The charging of the capacitor follows an exponential function given by the formula $V_c(t) = V_{max}(1 - e^{-t/\tau})$, where V_{max} is the maximum value that can reach the voltage*
When there is no input current, the capacitor starts to discharge through the resistor, following an exponential decay given by $V_c(t) = V_0 e^{-t/\tau}$, where V_0 is the initial value of the voltage.

Figure 3.4 explains the model working principle: neurons integrate incoming inputs from external signals or synapses and generate an output, firing an action potential. A key feature shared by the IF models is the reset of the state variable v when it reaches a predetermined threshold. When this happens, a spike is registered at time t_0. In these models, the exact shape of the action potential is not crucial; what matters is the time of its occurrence. Consequently, the output of the simulation is a spike train, with t_0 values corresponding the times at which

action potentials are elicited.

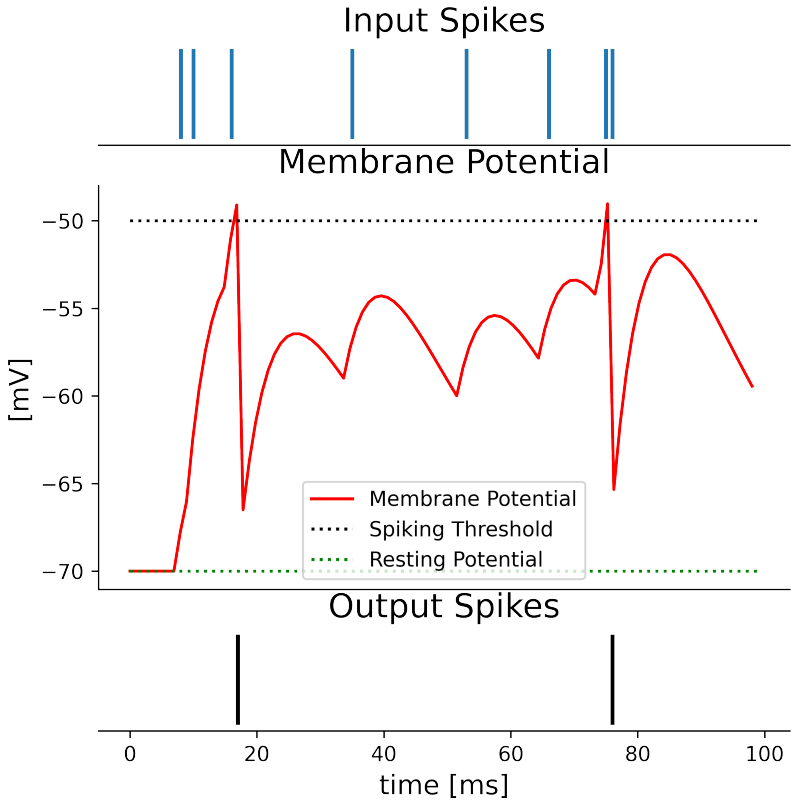


Figure 3.4: *On the top, there are the spikes in inputs to the model; the central plot shows the dynamics of the membrane potential that change according to the inputs; on the bottom, there are output spikes of the model, produced when the membrane potential exceeds the membrane threshold.*

The LIF model, while simple, has limitations in reproducing various neuronal dynamics, such as firing rate adaptation, and burst. To overcome these limitations, researchers have developed new IF models with enhanced capabilities. Some of these include the Quadratic Integrate-and-Fire (QIF) [47], the Exponential Integrate-and-Fire (EIF) [48], the Integrate-and-Fire-or-Burst (IFB) [49], the

Adaptive Exponential Integrate-and-Fire (AdEX) [50], and the Izhikevich (IZH) [51].

3.1.2 Conductance-based models

Conductance-based models [52] offer a straightforward biophysical representation of a neuron, where protein molecule ion channels are represented by conductances and the lipid bilayer by a capacitor. These models are grounded in an equivalent circuit representation of a cell membrane as well. This framework is characterised by a single isopotential electrical compartment, accounting for ion movements between the interior and exterior of the cell. The membrane potential v is influenced by the flow of ions, (such as sodium, potassium, and calcium) through various ion channels across the neuronal membrane.
The Hodgkin-Huxley (HH) model, introduced in 1952 [53] and based on the giant squid axon, stands as the pioneering conductance-based model. In this instance, the included ion channels are those for sodium (Na) and potassium (K) (figure 3.5).

Therefore, the model incorporates a sodium current I_{Na} with activation (m) and inactivation (h) variables, a potassium current I_K with only an activation (n) variable. This allows to accurately capture the dynamics of action potential generation.
The model's behaviour is described with a system of differential equations, where each equation represents the dynamics of a specific variable, such as membrane potential or ion conductance's activation and inactivation variables.

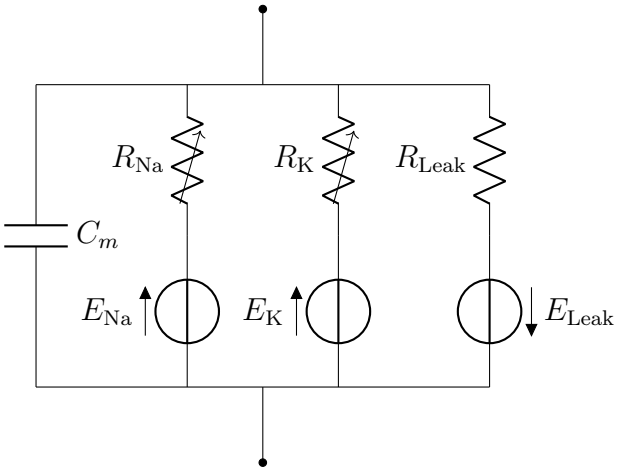


Figure 3.5: *The HH model accurately simulates the generation and propagation of action potentials in neurons. Its whole dynamics are modelled through an RC circuit. The most relevant parts of this circuit are R_{Na} and R_K. These resistances vary over time depending on the voltage. This introduces two new currents to the model that represent the flow of the sodium and potassium ions.*

$$\begin{aligned} C\frac{dv}{dt} &= -I_{Na} - I_K - I_{leak} + I \\ &= -\overline{g}_{Na} m^3 h \left(v - E_{Na}\right) - \overline{g}_K n^4 \left(v - E_K\right) - g_L \left(v - E_L\right) + I \end{aligned} \tag{3.3}$$

$$\frac{dn}{dt} = \alpha_n \left(v\right)\left(1 - n\right) - \beta_n \left(v\right) n \tag{3.4}$$

$$\frac{dm}{dt} = \alpha_m \left(v\right)\left(1 - m\right) - \beta_m \left(v\right) m \tag{3.5}$$

$$\frac{dh}{dt} = \alpha_h \left(v\right)\left(1 - h\right) - \beta_h \left(v\right) h \tag{3.6}$$

In equations 3.3 $\overline{g}_X$ is the maximum value for the ionic conductances, *alpha* and *beta* are voltage dependent parameters and E_X the ion equilibrium potential.

It is the membrane potential at which there is no net flow of ions.

3.1.3 Qualitative models

Qualitative models are simplified representations of conductance-based models, aiming to replicate neural dynamics using reduced equations. This simplification is achieved by focusing on the mathematical structures in ionic conductance models rather than delving into the details of ion charge dynamics.
The history of qualitative modelling traces back to the FitzHugh–Nagumo model [54], [55] and finds a theoretical basis in the theory of nonlinear dynamics [56]–[59].

The Fitzhugh-Nagumo model is well-suited for describing the dynamics of excitable cells, this model captures only the essential features of action potential generation, using:

1. A voltage-like variable v with cubic nonlinearity that allows regenerative self-excitation via positive feedback.

$$\frac{dv}{dt} = v - \frac{v^3}{3} - w + I \tag{3.7}$$

2. A recovery variable w with linear dynamics, providing slower negative feedback and associated with the activation and inactivation of ion channels.

$$\frac{dw}{dt} = \frac{1}{\tau}(v + \alpha - bw) \tag{3.8}$$

In equations 3.7 and 3.8 τ is time constant of the recovery variable, a, b are model constant parameters.

3.2 Piecewise Quadratic Neuron model

3.2.1 Reason behind the PQN model

A diverse range of neuron models has been employed in both large-scale SNN models and SiNNs [6]. Within this context, achieving a balance between the reproducibility of neuronal activities and computational efficiency is crucial.

The nervous system comprises various neuronal classes, each characterised by distinct electrophysiological properties, known to play an essential role in information processing [60]–[62]. Therefore, the chosen model should allow for the modelling of different classes. While ionic-conductance models offer this capability, their implementation in SiNNs is challenging due to resource-intensive circuit requirements. On the other hand, LIF models are computationally inexpensive. The IZH model and the AdEX model are extensions of the LIF that share the ability to cover multiple neuronal classes, with a slightly higher computational cost compared to the LIF model. They seem to be the best option but resetting the state variable in these IF-based models compromises the reproducibility of some features of neuronal activities.

For instance, in certain hippocampal cells, the amplitude of the spikes depends on the received stimulus [63]. This characteristic called graded response is particularly evident [57] in the Class II cells of the Hodgkin classification [64]. Notably, the assumption of spikes as stereotyped events in IF-based models prevents the faithful reproduction of this graded response.
Another example is the phase-resetting curve (PRC) [57], which delineates how the phase of a neuron in an oscillatory spiking state changes based on the timing of a pulse stimulus. Interestingly, the PRC shape for Class II modes in IZH and AdEX

models deviates from the standard Type II. However, neurons exhibiting Type II behaviour are recognized for their role in promoting synchronous firing [65]. Consequently, these neurons may play a crucial role in modelling synchronous activity in the brain. These pieces of evidence suggest that an IF-based network may have a limited capacity to faithfully replicate certain aspects of information processing in the nervous system.

Qualitative neuron models offer a lower computational cost than ioni conductance models while faithfully reproducing the dynamics of the spiking process without requiring state variable resets. The FitzHugh–Nagumo model and the Hindmarsh–Rose model [66] represent a single neuron class and are characterised by equations containing third-degree polynomials.
In 2007, a Digital Spiking Silicon Neuron (DSSN) [67], [68] model was introduced, following a qualitative approach. This model utilises quadratic terms and piecewise and step functions, enabling efficient software and digital arithmetic circuit implementation. Moreover, it is designed to represent a broad range of neuron classes [69].

3.2.2 PQN model formulation

The PQN model [16] is an enhanced version of the DSSN model that can replicate graded responses and Type II PRCs. Additionally, the PQN model can emulate the activity of ionic-conductance models of six neuronal classes (RS, FS, IB [70], LTS [71], Elliptic Bursting (EB) [72], and Elliptic Bursting (EB) [73]) in response to step inputs of various magnitudes. Notably, its FPGA implementation consumes significantly fewer circuit resources than the ionic-conductance model, thanks to its simple equations tailored for digital arithmetic circuits [74]. The PQN model aims

to offer a tool for constructing large-scale and biologically accurate SNN models and SiNNs.

To efficiently capture the distinct dynamics of its neuronal classes, the PQN model is presented in four variations:

1. The PQN model supports the Class II. v and n represent the membrane potential and recovery variable, respectively.

$$\frac{dv}{dt} = \frac{\phi}{\tau}\left(f\left(v\right) - n + I_0 + kI_{stim}\right) \tag{3.9}$$

$$\frac{dn}{dt} = \frac{1}{\tau}(g\left(v\right) - n) \tag{3.10}$$

In equations 3.9 and 3.10 $f(v)$, $g(v)$ are quadratic terms, I_0 is a bias constant, I_{stim} is the stimulus input, k is a scaling parameter, τ, ϕ are time constants of the variables.

2. The three-variable PQN model introduces the slow variable q. It models the RS, FS, and EB classes.

$$\frac{dv}{dt} = \frac{\phi}{\tau}\left(f\left(v\right) - n - q + I_0 + kI_{stim}\right) \tag{3.11}$$

$$\frac{dn}{dt} = \frac{1}{\tau}(g\left(v\right) - n) \tag{3.12}$$

$$\frac{dq}{dt} = \frac{\epsilon_q}{\tau}(h\left(v\right) - q) \tag{3.13}$$

In equation 3.13 ϵ_q is the time constant of the slow variable, $h(v)$ is a quadratic term.

3. The four-variable PQN model adds the slow variable u to replicate the PB class. Here the slow subsystem consisting of u and q is capable of producing oscillations.

$$\frac{dv}{dt} = \frac{\phi}{\tau}\left(f\left(v\right) - n - q + u + I_0 + kI_{stim}\right) \tag{3.14}$$

$$\frac{dn}{dt} = \frac{1}{\tau}(g\left(v\right) - n) \tag{3.15}$$

$$\frac{dq}{dt} = \frac{\epsilon_q}{\tau}(h\left(v\right) - q) \tag{3.16}$$

$$\frac{du}{dt} = \frac{\epsilon_u}{\tau}(v + v_0 - \alpha_u u) \tag{3.17}$$

In equation 3.17 ϵ_u is the time constant of u. v_0, $\alpha_u u$ are model parameters.

4. The extended four-variable supports the LTS and IB classes. The only difference to the previous is that the time constant of n is dependent on u.

$$\frac{dv}{dt} = \frac{\phi}{\tau}\left(f\left(v\right) - n - q + u + I_0 + kI_{stim}\right) \tag{3.18}$$

$$\frac{dn}{dt} = \frac{\eta(u)}{\tau}(g\left(v\right) - n) \tag{3.19}$$

$$\frac{dq}{dt} = \frac{\epsilon_q}{\tau}(h\left(v\right) - q) \tag{3.20}$$

$$\frac{du}{dt} = \frac{\epsilon_u}{\tau}(v + v_0 - \alpha_u u) \tag{3.21}$$

In equation 3.21, $\eta(u)$ is a heavy side step function.

All the parameters' values together with their equations can be found in [16].

Chapter 4

Neuromorphic hardware

Software-based simulations of SNN models on conventional computers offer flexibility and control, but they consume considerable computational power and exhibit slow simulation speed. GENESIS [75], NEURON [76], NEST [77], Brian [78], Auryn [79], and EDEN [80] operate on conventional CPUs, ANNarchy [81], GeNN [82], and Brian 2 [83] provide GPU support.
In contrast, SiNNs comprise electronic circuits designed to emulate SNN models. They enable power efficiency and support real-time or even accelerated-time simulations.

4.1 Analog and digital hardwares

Silicon neurons employing analog circuits exhibit exceptional power efficiency (1-10 pW/spike) but are sensitive to noise, mismatch, and variations in power, voltage, and temperature. Conversely, silicon neurons utilising digital circuits are considerably less sensitive to these factors. Although digital circuits consume more power (about one or two orders of magnitude) than their analog counterparts, they of-

fer increased portability, ease of operation, and seamless integration. Typically, small-scale systems find bioinspired edge computing applications, while large-scale systems are tailored for cognitive computing purposes [6].

Among the analog devices, the ROLLS chip [84] incorporates 256 AdEX neurons, the DYNAPs chip [85] features a quad-core setup with 2k neurons and 64k synapses, and Neurogrid [86], a 1-million-neuron system, aims to emulate the biophysics of cortical layers.

Concerning digital, the small-scale design by [87] integrates 256 LIF neurons. The ODIN chip accommodates 256 neurons with 20 Izhikevich behaviours [88]. The MorphIC chip allows large-scale multichip setups, totalling 2k LIF neurons [89]. SpiNNaker adopts a distributed von Neumann approach with a globally asynchronous locally synchronous (GALS) design, covering biological to accelerated time constants [90].

Large-scale networks have also been constructed using digital ASICs. Intel Loihi [91] supports up to 180k LIF neurons with configurable compartments, incorporating functionalities such as threshold adaptation, axonal and refractory delays and spike latency. IBM TrueNorth [92] has 1M neurons replicating 11 Izhikevich behaviours or 20 if three neurons are combined [93].

FPGAs are used in various scenarios, ranging from cognitive computing [94]–[97] to neuroscience applications [24], [98], [99].

4.2 Implementation of the PQN model

The work of [16] encapsulates a population of N (9993) identical neurons. This chip receives a step current as input, calculates the state variables for each cell

and outputs the voltage. Depending on the selected bitstream file among the eight available options, the population can consist of Class II, RS excitatory (RSexci), RS inhibitory (RSinhi), FS, EB, PB, IB, or LTS equal neurons. The design of these classes remains consistent, apart from the innermost module called *PQN engine* (Figure 4.1). All state variables are represented using an 18-bit fixed-point format, incorporating an 8-bit integer part representation except for the Class II mode that uses 28-bit format.

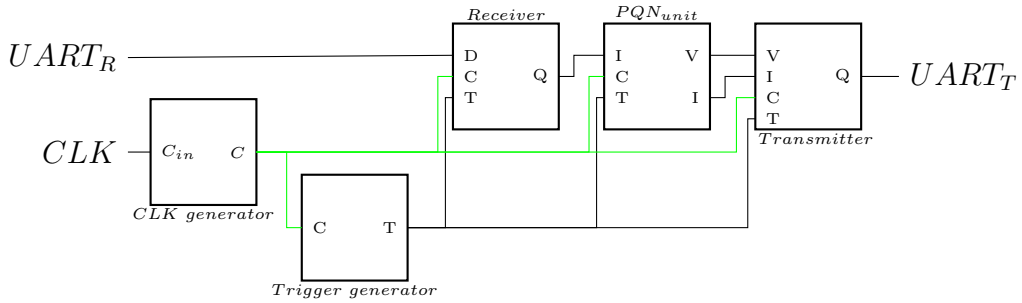


Figure 4.1: *The schematic diagram of the PQN model is equal for every class. $UART_R$ is the only signal received from the computer, and $UART_T$ the only one transmitted. The output of the clock module is fed in any other following module, such as the the trigger signal. The receiver communicates the input current to the PQN unit, which after calculating the voltage, gives it to the transmitter.*

The hardware schematic diagram for this project is illustrated in figure 4.1. It is composed of the *PQN unit*, *receiver*, *transmitter*, *trigger generator*, and *clock* modules.

The *receiver* and *transmitter* modules facilitate communication with the host PC. The *transmitter* takes the voltage (V) and current (I), outputs from the *PQN unit*, and encodes these 18-bit values into a 3-byte format. The *uart txd ctrl* submodule manages the transmission aspect of the UART communication pro-

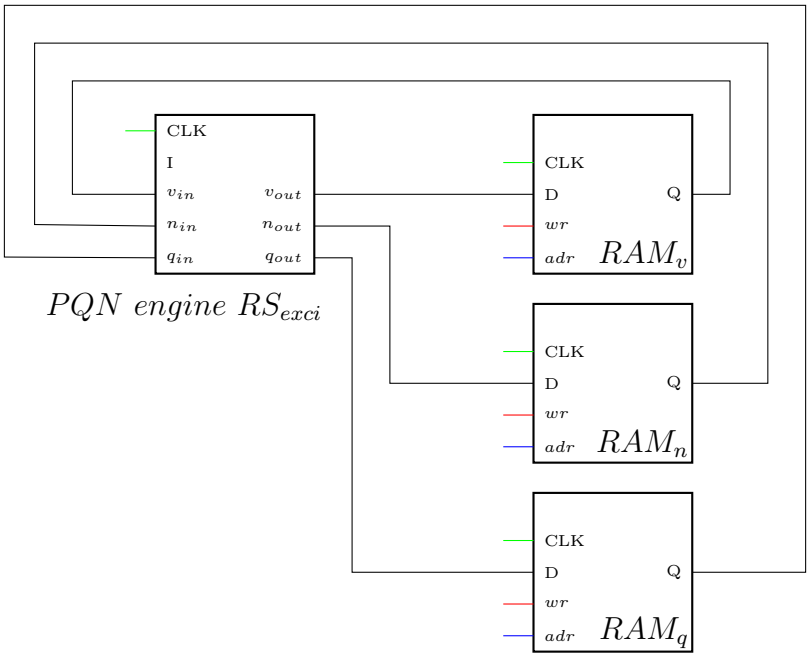


Figure 4.2: *Inside the PQN unit, there is always a PQN engine and a specific number (2-4) of blocks of RAM. Here CLK and I are external signals (Figure 4.1), while wr and adr are, respectively, the write enable and the address of the RAM. These signals are internal to the PQN unit and are equal for each RAM.*

tocol, sending these 3-byte words to the computer bit by bit. Meanwhile, the *uart rxd ctrl* submodule handles the reception part of the UART, assembling received bits into 3-byte words. The *receiver* then decodes the 3-byte word into an 18-bit value and stores it in a RAM block. Subsequently, this value is fed back as an input current to the *PQN unit*. Therefore, the *transmitter* transfers a single voltage and a single current, while the *receiver* receives only one current value. The *clock* is an IP used to synchronise data transmission with the FPGA's internal clock, and the *trigger generator* generates a trigger signal every 0.1 ms.

The *PQN unit* (Figure 4.2) comprises the *PQN engine* and RAM blocks. Here, N denotes the number of neurons processed by one *PQN unit*. The *PQN unit*

operation is straightforward — it retrieves the state variable values from a RAM and inputs them into the *PQN engine*. Simultaneously, it obtains an output (state variables) from the *PQN engine* and stores them back into the RAM. The RAM blocks store the state variable values during each cycle, and each block is dedicated to a single state variable.
The described structure remains consistent across all neural classes, with variations solely in the number of RAMs utilised based on the simulated class. Instead, the *PQN engine* serves as the computational core for the neuron and varies depending on the neural class. It calculates the differential equation of the specific class using the Euler method.

The focus of this section is on the implementation of the three-variables model. The computation of the x_{out} is done summing to x_{in} additional terms (Equations 4.1-4.3). These terms consist of constants or variables derived from the differential equations discussed in Chapter 3.2.2. The variable terms are computed by multiplying the state variables and the input current I_{in} with a set of 17 constant coefficients obtained through experimental parameter calculations. In particular, the variables depending on v and v^2 needs two coefficients each, the ones depending on I_{in}, n and q only one, while the dependence on x indicates a constant.

$$v_{out} <= v_{in} + v_{vv} + v_v + v_x + v_n + v_q + v_{Iin} \tag{4.1}$$

$$n_{out} <= n_{in} + n_{vv} + n_v + n_x + n_n \tag{4.2}$$

$$q_{out} <= q_{in} + q_{vv} + q_v + q_x + q_q \tag{4.3}$$

The multiplication operations between variables and coefficients are executed

using shifters and adders (Figure 4.3). Employing a shifter is a more efficient multiplication method than using a dedicated multiplier. This efficiency arises from the binary nature of the values, where multiplication by 2 is effectively a left shift of one position (towards most significant bit), and multiplication by 0.5 is a right shift (towards less significant bit). Given that, the values of the parameters have a direct impact on the utilisation of circuit resources. Each additional number of ones in the binary representation of the coefficients consumes an additional pair of shifters and adders.

```
-- x * 0.16064453125 (= 00000000.00101001001000000000)
x_0 <= "000" & x(17 downto 0) & "00000" when x(17)='0'
    else "111" & x(17 downto 0) & "00000";
x_1 <= "00000" & x(17 downto 0) & "000" when x(17)='0'
    else "11111" & x(17 downto 0) & "000";
x_2 <= "00000000" & x(17 downto 0) & "" when x(17)='0'
    else "11111111" & x(17 downto 0) & "";
x_3 <= "00000000000" & x(17 downto 3) & "" when x(17)='0'
    else "11111111111" & x(17 downto 3) & "";
x_4 <= x_0 + x_1 + x_2 + x_3;
x_product <= x_4(25 downto 8);
```

Figure 4.3: *Example: the binary representation of 0.16064453125 is done utilising four 1-bits, respectively, in position 3, 5, 8 and 11. To multiply a variable (x) by this coefficient it's thus necessary a left shift of 3 positions (x_0), one of 5 (x_1), one of 8 (x_2), and one of 11 (x_3) and sum these four values together.*

For each clock cycle, the *PQN engine* computes the next step of the state variables based on their previous values (Figure 4.4). It takes the module 4 cycles to calculate the next step, employing a pipelined architecture. A pipeline is a sequence of stages, each performing a part of the overall task. Each stage completes a portion of the operations and passes the partial result to the next stage. By passing incomplete tasks down the pipeline, each stage can initiate work on a new

task without waiting for the overall task to conclude [32]. This pipelined structure computes a new value every cycle after the initial four cycles.

The only multiplier in the entire code is used for calculating v^2. This is accomplished during the initial stage (Figure 4.4). Subsequently, the second stage involves the multiplication of variables and coefficients. In the third stage, the module calculates v_{next}, n_{next}, and q_{next}, and in the fourth stage, these values are outputted.

For real-time simulation with a 100 MHz clock and a time step of 0.1 ms, up to 10,000 cycles can be consumed to update all state variables. Considering that the PQN engine requires four cycles and read/write operations to the RAM necessitate three clocks, the maximum number of neurons (N) is 9993.

It's crucial to emphasize that, although synaptic currents are calculated, they are not applied or utilized in this work. As a result, the neurons in the population are not interconnected.

The FPGA is stimulated from the computer using a Python code. The number of bits transmitted per second (Baud Rate) is set at 1 Mbits/s. Figure 4.5 illustrates the result of the stimulation of each one of the 8 modes.

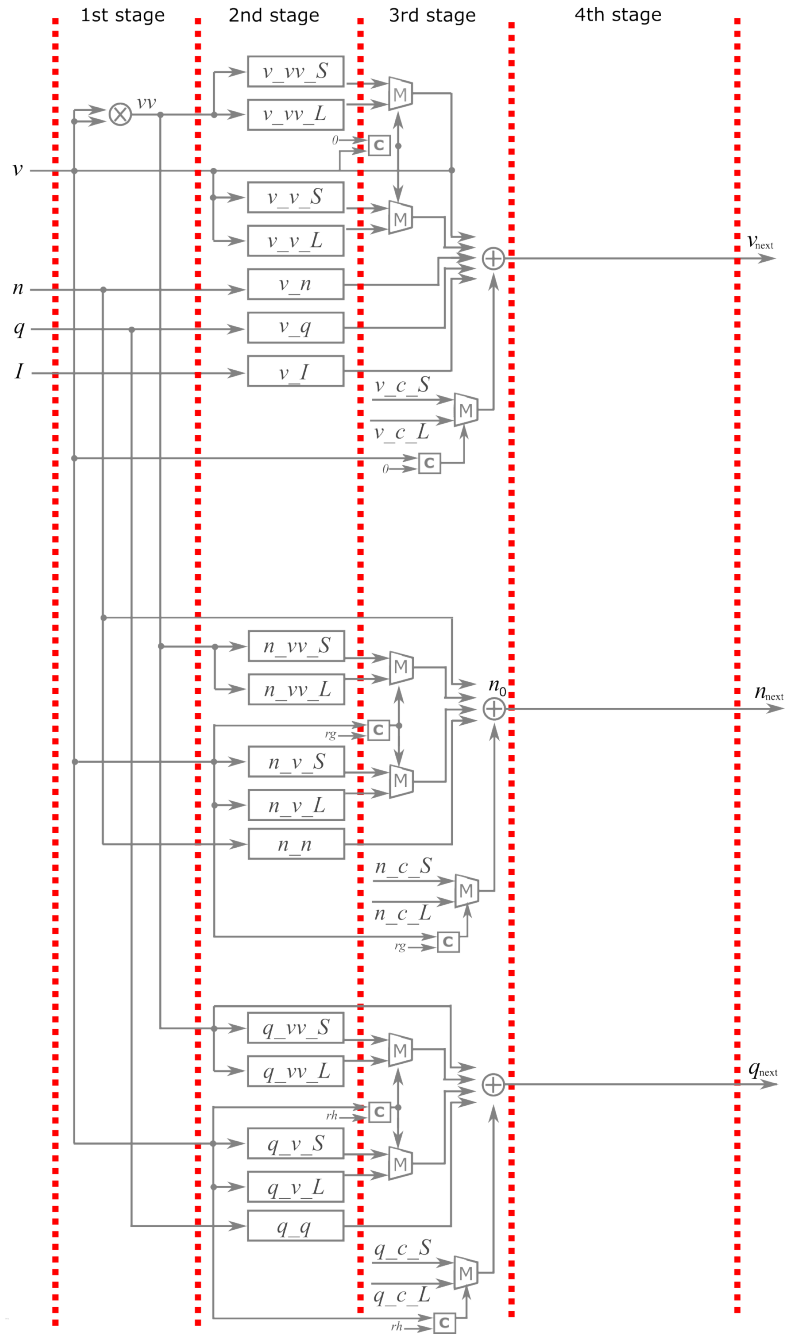


Figure 4.4: *PQN engine diagram for the three variables model. Starting from the inputs v, n, q and I_{stim}, the PQN engine computes the outputs v_{next}, n_{next} and q_{next} using multipliers ($\times$), adders ($+$), multiplexers (M), and comparators (C).*

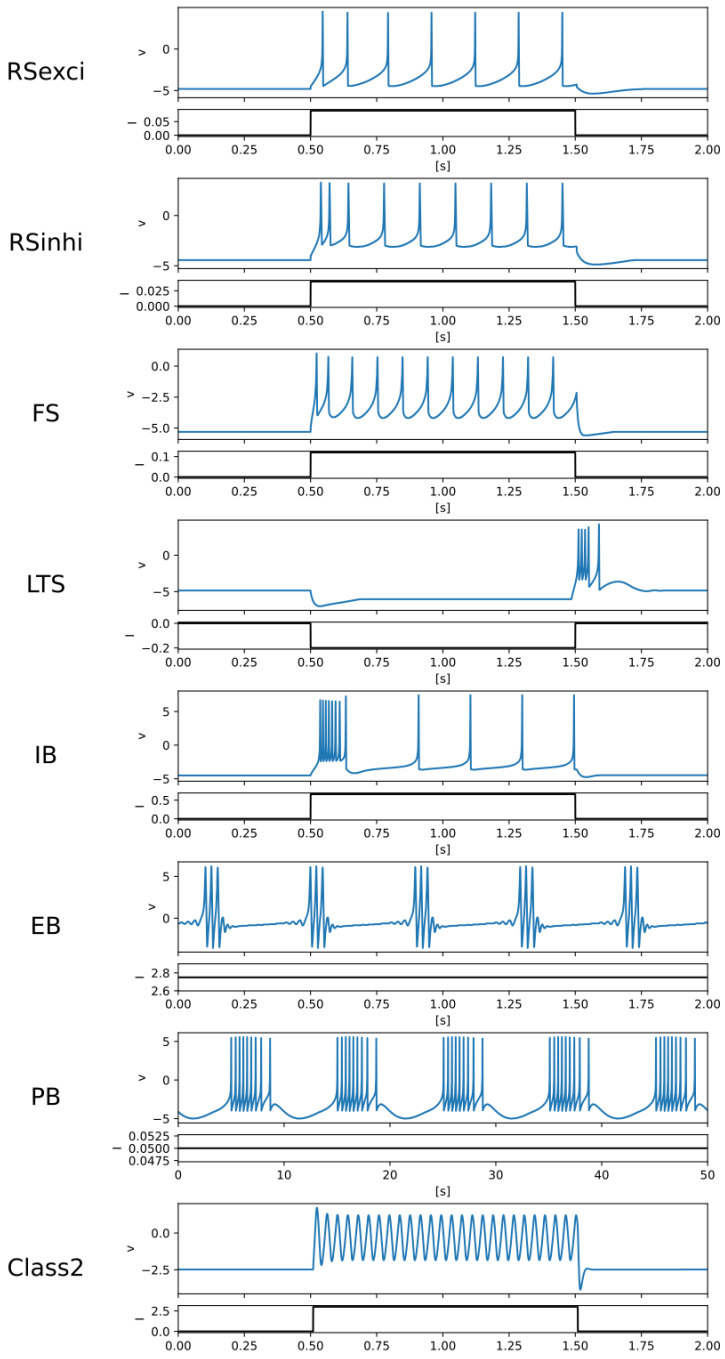


Figure 4.5: *Simulation of the 8 modes of the PQN model. Each class is stimulated using a different step current during 1s. PB and EB are stimulated using a continuous current instead.*

Chapter 5

Design projects

This chapter elucidates the innovative work undertaken in this thesis.

5.1 Multiple classes population

Classifying cortical neurons into distinct classes provides a fundamental framework for understanding neural diversity. The four prominent cell classes in the cortex are FS, RS, IB, inspired by Connors and Gutnick's classification [70], and the LTS cells.

RS neurons, characterised by a spiny pyramidal-cell morphology, comprise the predominant cell class in the neocortex. These neurons can exhibit either excitatory or inhibitory characteristics, with the former being more abundant. Moreover, the internal dynamics of these two cell types differ, necessitating the use of distinct models to represent each accurately.

The classification of neurons as excitatory or inhibitory is based on the neurotransmitter synthesised and released at the synapse. Glutamatergic neurons release glutamate, exciting postsynaptic cells, while GABAergic neurons release GABA (γ-aminobutyric acid), inhibiting them. Maintaining a delicate balance between

excitation and inhibition is crucial for the precise functioning and plasticity of the nervous system.
Another significant class is the FS neurons, typically corresponding to aspiny inhibitory neurons.
The IB neurons, generating bursts of action potentials in response to depolarising stimuli, represent a smaller percentage of cells in the primary sensory cortex.
LTS activity was observed in about 10 % of intracellularly recorded cells in the cat association cortex in vivo [71]. These LTS neurons resemble the classic RS response and additionally produce bursts of action potentials. While the LTS cell shares similarities with the thalamic relay cells, key distinctions exist. Thalamic relay cells lack spike-frequency adaptation, eliminating the need for adaptation currents. Furthermore, thalamic relay cells generate more powerful bursts than cortical LTS neurons in both the cortex and thalamus [29].
Despite these differences, the LTS class can be utilised to model thalamic neurons. A chip featuring these four neuronal classes can be a robust model for a thalamocortical network and prove valuable in various experimental scenarios, as previously done in other studies, like [24].

5.1.1 Implementation

As illustrated in figure 5.1, the principal distinction from the design presented in Chapter 4.2 lies in the *PQN unit* module. It features multiple *PQN engines* (Figure 5.2), each equipped with a corresponding number of RAM blocks.

Within the *PQN unit*, four distinct *PQN engines* have been incorporated — RSexci, FS, LTS, and IB. The design needs three RAMs for the first two classes and four for the latter two classes. Each class is associated with a unique stimulus

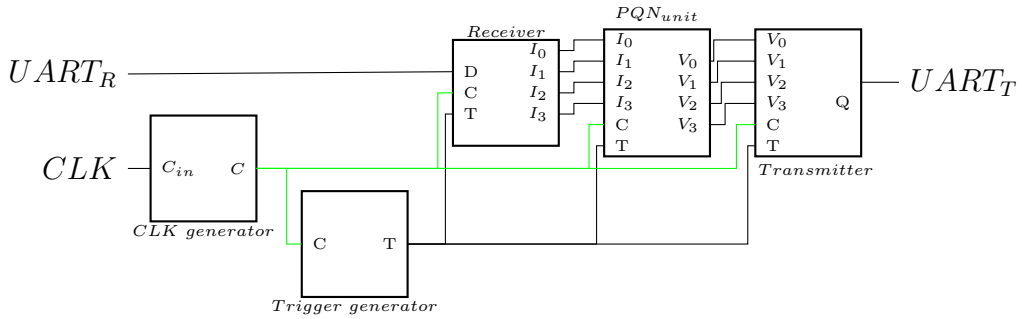


Figure 5.1: *The schematic diagram for the implementation of a multi class population differs from 4.1 only for the number of currents output of the receiver and thus in the number of voltages calculated by the PQN unit and sent to the transmitter.*

current, differing in amplitude. Consequently, each of the four different currents must be transmitted to the respective class.

This required modifications to *transmitter* and *receiver*, with the first that transfer four voltages and the second that receives four currents and stores them in the four RAMs. The baud rate has been adjusted from 1 Mbits/s to 2 Mbit/s because of the increase in the data rate.

Figure 5.3 shows the simultaneous stimulation of the four different classes with diverse currents.

The code relative to this section can be found in the *multiple_classes*[1] repository on GitHub.

[1] `https://github.com/gle0/PQN_thesis/tree/main/multiple_classes`

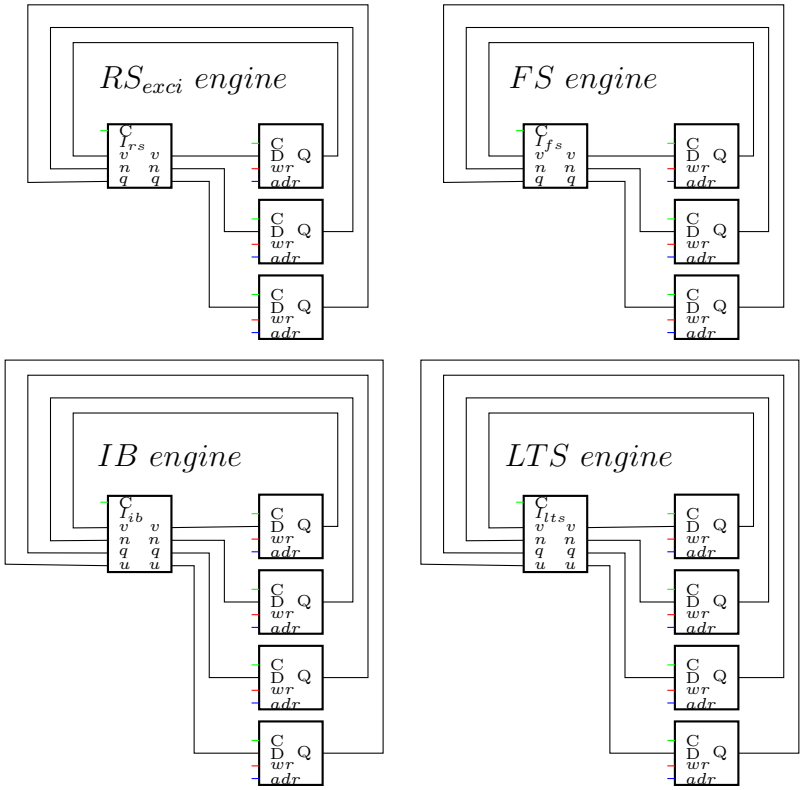


Figure 5.2: *The PQN unit is composed of four independent and parallel compartments like the one in figure 4.2. IB and LTS classes have four-variables, so their implementation requires an additional RAM. The wr and adr signals are equal for all the blocks.*

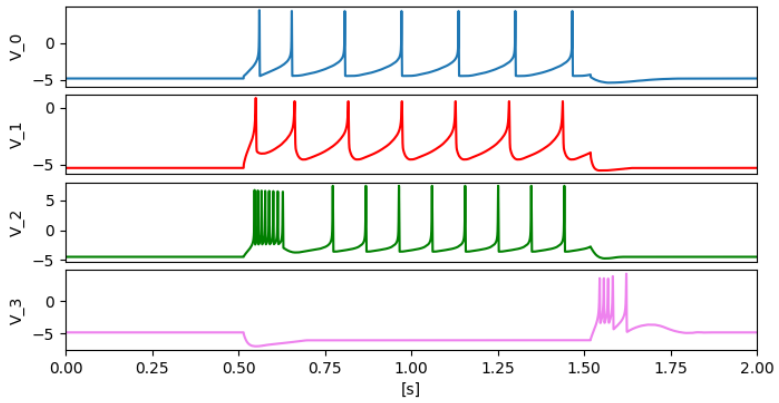


Figure 5.3: *Result of the stimulation of RSexci (V_0), FS (V_1), IB (V_2) and LTS (V_3) classes with a step current of 0.09, 0.1, 0.7 and -0.2, respectively.*

5.2 Cell-to-cell variability

In constructing thalamocortical networks, including diverse classes is essential, along with modelling the inherent cell-to-cell variability within each class [29]. Unlike computers with billions of identical gates, neurons in the brain exhibit significant individuality [100], [101]. Even neurons of the same type display considerable variances in their properties, introducing variability to signal processing. While this variability might seem to challenge the reliability of neuronal networks, it has been observed that instead it enhances a system's inherent functional behaviour [102]–[104]. In fact, this intrinsic mechanism contributes to increased network speed, responsiveness, and robustness [30].

5.2.1 Implementation

To exploit the cell-to-cell variability, it's pivotal to provide each neuron with a heterogeneous set of parameters (seen the PQN model's equations in Chapter 3.2.2). In this project the coefficients discussed in Chapter 4.2 are a semplification of those parameters.
In order for each neuron to have a proper set of coefficients, it's important to make them change at any cycle rather than being constants. In fact, the coefficients in [16] were fixed, allowing for straightforward multiplication using simple shifters and adders. However, this approach is no longer applicable, and a novel solution is required. Instead of constructing hardware around fixed values for the coefficient, the goal is to develop generic hardware capable of accommodating any value.

Therefore, there is the need for a *barrel shifter* module (Figure 5.4) capable of shifting a variable based on a number. This module takes a variable and a 5-bit

signal (*shift amount*) as input, outputting the variable shifted by the specified number [105]. The shift amount determines the positions by which the input is shifted.

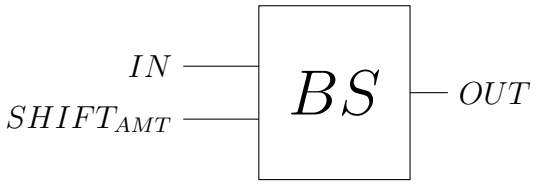


Figure 5.4: *The barrel shifter is a module composed of many MUXs. It takes two inputs, one is a variable and the other one serves as the selector for these MUXs. This module allows to shift a variable by any value.*

Another challenge arises in determining how to store these dynamic coefficients. Since all the values are binary, only the 1 bits are relevant to the multiplication operation. To address this, a strategy is employed to store the positions of 1s exclusively.

For instance, the binary coefficient 0000100110 (10-bit fixed-point, with a 1-bit integer part, equivalent to 0.07421875 in decimal) is encoded by the positions 4, 7, and 8 expressed in a 5-bit notation (00100, 00111, 01000).

The maximum number of 1s that can be stored for each coefficient has been determined as 10 considering the coefficients in [16]. Summing up, ten positional values are stored for each coefficient, with each position expressed using 5 bits (Figure 5.5 top). After doing this, the ten shifted values are summed, giving the result of the multiplication between a variable and a coefficient.

Consequently, for each neuron 17 different 50-bit arrays are stored in the respective ROM block. In cases with fewer than ten 1s, the remaining operations involve a 37-bit shifting, resulting in an empty array. The addition of these null values does not impact the calculation of the coefficient (Figure 5.5 bottom).

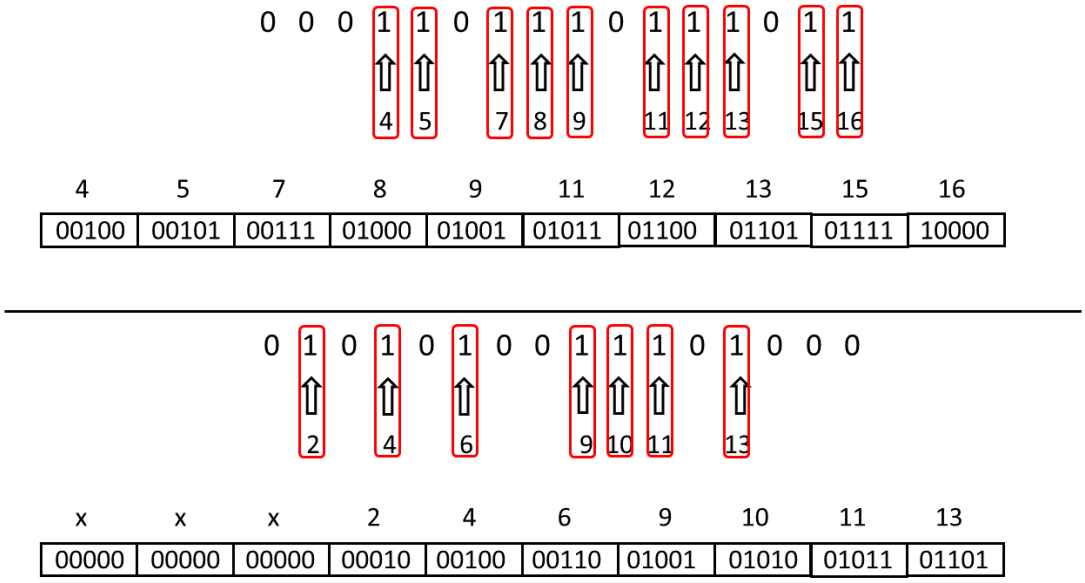


Figure 5.5: *In the first case (top) the coefficient is encoded using 10 1s, in the second (bottom) case less than 10.*

The overall design is illustrated in figure 5.6. The modification is focused on the *PQN engine*, which incorporates 17 ROMs (one for each coefficient) and 10 *barrel shifters* for each ROM.

The revised *PQN engine* operates through a new process involving eight distinct stages (Figure 5.7). The initial stage involves the computation of v^2. In the second ten shifted values are calculated for each coefficient. In the third stage, these ten values are summed in pairs. This results in five values, summed together in the fourth and the fifth stages. In the sixth stage, nine coefficients become negative. The seventh stage computes v_{out}, n_{out}, and q_{out}. Finally, in the eighth stage, these values are outputted.

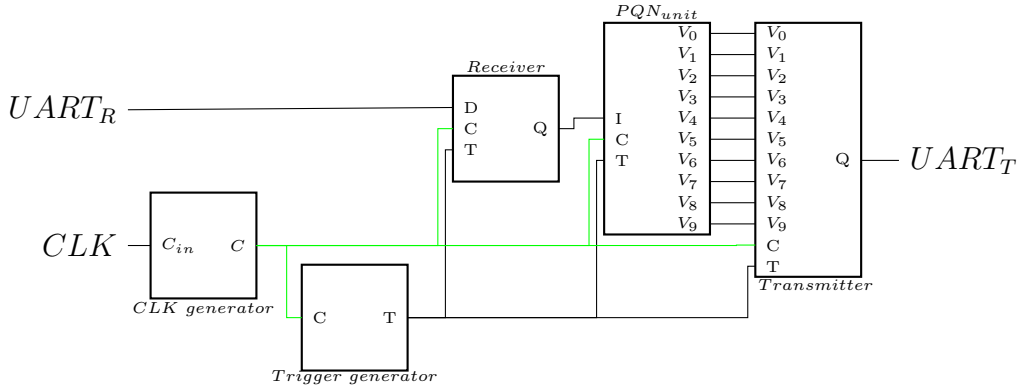


Figure 5.6: *The schematic diagram for the heterogeneous population differs from the one in figure 4.1 only in the number of voltages computed by the PQN unit and sent to the transmitter.*

One current is received and ten voltages are transmitted. The baud rate has thus been increased from 2 Mbits/s to 4 Mbits/s.
Figure 5.8 shows the stimulation of a heterogeneous population of RSexci neurons with the same step input current.

The relevant code for this section can be found in the GitHub repository called $cell - to - cell\_variability$[2].

[2] `https://github.com/gle0/PQN_thesis/tree/main/cell-to-cell_variability`

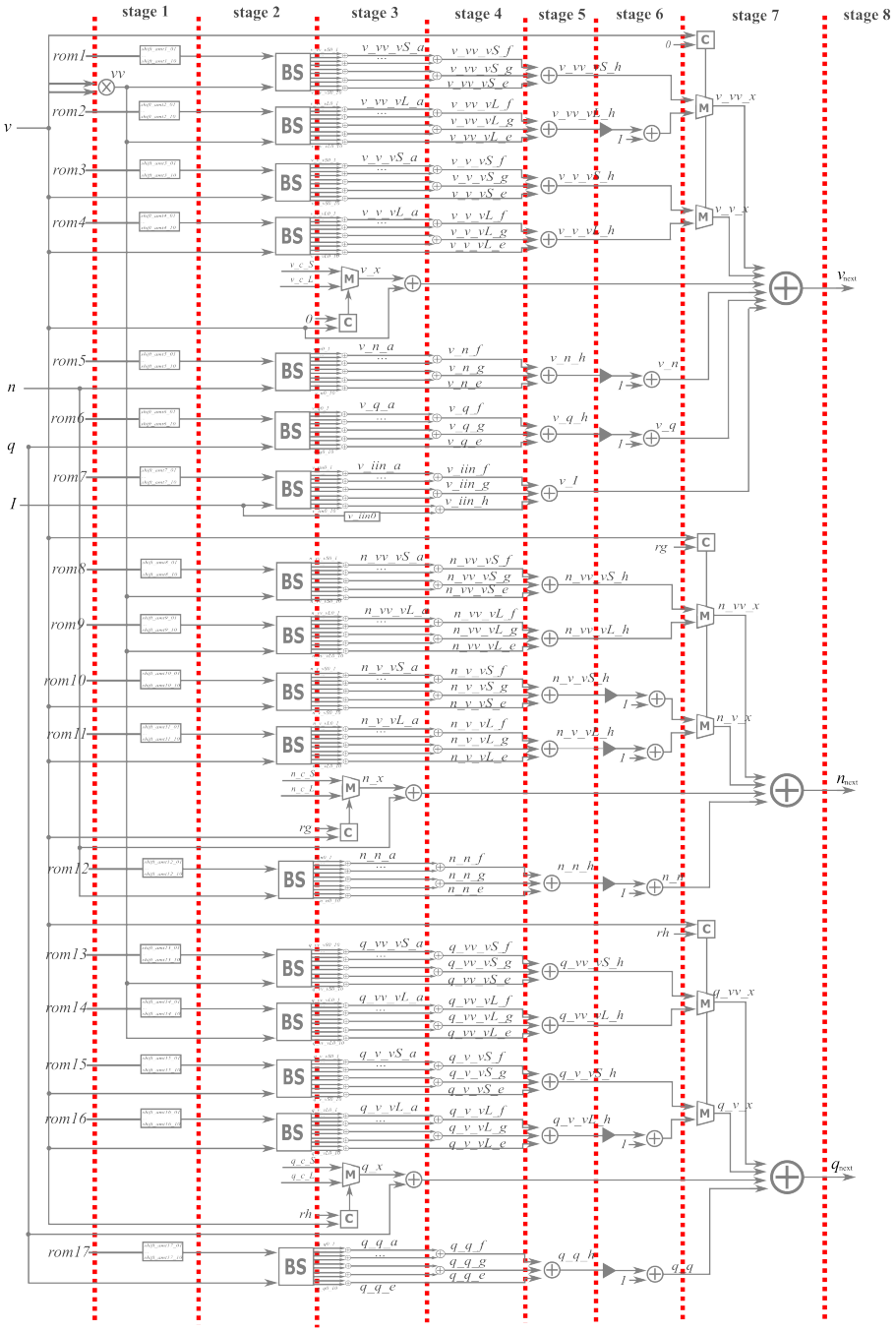


Figure 5.7: *PQN engine diagram for the three variables model that implement neuronal heterogeneity. BS is the symbol indicating the barrel shifter.*

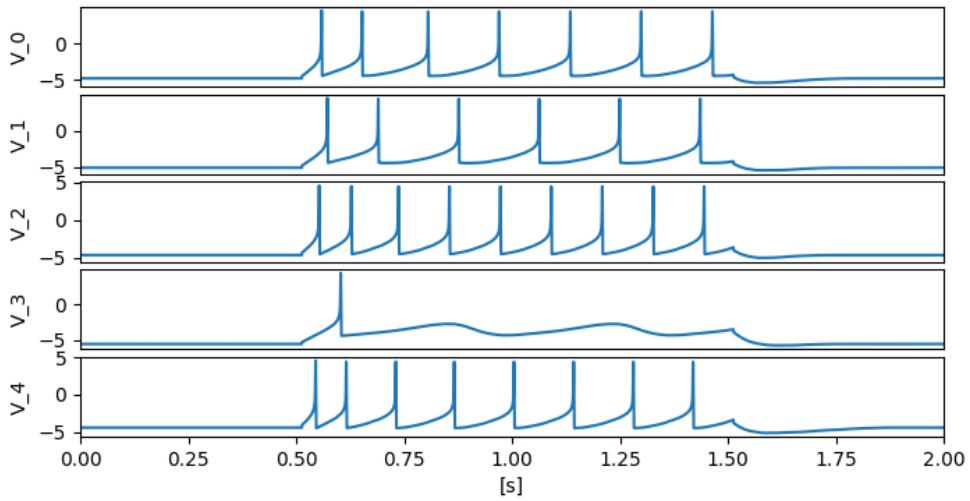


Figure 5.8: *Stimulation of 5 RSexci neurons with a step current. Although the neurons are all from the same class, their response to the same current is different.*

Chapter 6

Outlook

The PQN model ensures biological accuracy while maintaining lower power consumption than conductance-based models. It can model various neuron classes and guarantees efficient implementation through digital circuits. Its FPGA application offers high flexibility and represents an ultralow-power, large-scale, general-purpose silicon neuronal network platform.
The two designs introduced in this study both contribute to increasing the biological features. The first project consolidates the four principal electrophysiological classes of cortical and thalamic neurons into a single device, while the second one introduces intraclass variability. The common objective is the development of a silicon thalamocortical network.
To further advance the project, future endeavours should focus on reducing FPGA resource utilisation, integrating both designs synergistically, and incorporating synapses.

While the applications of these devices are yet to be explored, the expectations are exceedingly high. A multitude of event-based sensors has already been developed, such as silicon retina [106], silicon cochlea [107], [108], neuromorphic

vestibular systems [109], neuromorphic olfactory circuit [110], and touch [111].
In addition to perception and inference, the event-based paradigm also applies to control [112]. Various methods have been proposed to achieve locomotion in a range of robots, employing a Central Pattern Generator (CPG). It is a neural network wherein interconnected excitatory and inhibitory neurons generate an oscillatory, rhythmic output without requiring rhythmic inputs [113].

Moreover, the recent availability of extensive anatomical and physiological data has led to the development of data-driven conductance-based SNN models [18], [114], [115] with the ambitious goal of replicating the mammalian brain. The significant computational demand of these conductance-based models precludes their widespread use and the construction of larger-scale networks [18]. In contrast, SNNs employing the PQN model demonstrate a remarkable reduction in power consumption, making them a promising tool for these applications [16].
Furthermore, representing a bottom-up approach with cell-level granularity and accurately replicating complex spiking activities of neurons, these SiNNs are well-suited for artificially recreating brain regions [74]. Consequently, it holds potential applications in biohybrid systems, including neuroprosthetic devices designed to treat neurological disorders and injuries [19].

Brain disorders represent a leading cause of global disabilities, and current pharmacological treatments, along with the limited availability of robotic aids, fall short in providing effective solutions. This situation necessitates a re-evaluation of methodologies for studying human cells and addressing brain-related illnesses. A promising avenue in this pursuit is the exploration of "hybrid neuromorphic engineering" which involves integrating technological devices with biological neurons to devise innovative solutions [24]. Novel approaches to brain repair [22] or re-

placement [21], [25], that establish adaptive bidirectional communication through a real-time closed-loop architecture between biological and artificial components have emerged. This approach aims to overcome the limitations of current treatments by fostering dynamic interactions between biological and artificial systems. Notably, the characteristics of SiNNs studied in this project align well with the need for a trade-off between the complexity of computational models, energy consumption, and real-time constraints inherent in these applications. The use of a SiNN employing the PQN model presents an exciting prospect in this context. Furthermore, the selection of FPGA hardware is a strategic choice to optimise the system's real-time performance. This not only addresses the need for efficient computation but also accelerates the development of an implanted bioelectronic device, offering potential applications in the biomedical field in the future.

This integrated approach holds promise for advancing our capabilities in treating and understanding brain-related disorders. Moreover, exploring the interaction between artificial devices and living cells raises questions about the essence of intelligence and cognition. This offers the potential to find the boundary between the artificial and the biological life, other than the charming possibility of stumbling upon plausible answers to those questions.

Acronyms

AdEX Adaptive Exponential Integrate-and-Fire. 37, 40, 46

AI Artificial Intelligence. 10, 14

AN Artificial Neuron. 12, 13

AP Action Potential. 31, 32, 34

ASIC Application Specific Integrated Circuit. 27

ASSP Application Specific Standard Product. 27

CLB Complex Logic Block. 27

CMOS Complementary Metal-Oxide-Semiconductor. 19

CNN Convolutional Neural Network. 12

CPG Central Pattern Generator. 66

CPLD Complex Programmable Logic Device. 24, 26

CPU Central Processing Unit. 11

DFF Delay Flip Flop. 22, 23

DNN Deep Neural Network. 12

DSSN Digital Spiking Silicon Neuron. 41

EB Elliptic Bursting. 41, 42, 53

EIF Exponential Integrate-and-Fire. 36

EPROM Erasable Programmable Read-Only Memory. 26

FF Flip Flop. 22, 26

FPGA Field Programmable Gate Array. 4, 13, 24, 26–30, 41, 46, 48, 51, 65

FS Fast Spiking. 16, 41, 42, 54, 55

HDL Hardware Description Language. 28, 29

HH Hodgkin-Huxley. 37, 38

IB Intrinsically Bursting. 16, 41, 43, 54, 55, 57

IF Integrate-and-Fire. 34

IFB Integrate-and-Fire-or-Burst. 36

IP Intellectual Property. 48

IZH Izhikevich. 37, 40

LE Logic Element. 26

LIF Leaky Integrate-and-Fire. 34, 40, 46

LTS Low Threshold Spiking. 16, 41, 43, 55, 57

LUT Look-Up Table. 27

MEA Microelectrode Array. 15, 16

MUX Multiplexer. 60

PAL Programmable Array Logic. 24, 26

PB Parabolic Bursting. 43, 53

PC Personal Computer. 30, 47

PLA Programmable Logic Array. 26

PLD Programmable Logic Device. 24, 26

PQN Piecewise Quadratic Neuron. 3, 14, 41–43, 47, 53, 57, 65, 66

PRC phase-resetting curve. 40

PROM Programmable Read-Only Memory. 24, 26

QIF Quadratic Integrate-and-Fire. 36

RAM Random Access Memory. 11, 23, 27, 48, 49, 55–57

RNN Recurrent Neural Network. 12

ROM Read-Only Memory. 20, 21, 23, 60, 61

RS Regular Spiking. 16, 41, 42, 54, 55

RWM Read-Write Memory. 23

SiNN Silicon Neural Network. 4, 13, 14, 16, 40, 42, 45, 66, 67

SN Spiking Neuron. 13

SNN Spiking Neural Network. 11, 40, 42, 45, 66

TBI Traumatic Brain Injury. 15

UART Universal Asynchronous Receiver-Transmitter. 30

VHDL Very High-Speed Integrated Circuit (VHSIC) Hardware Description Language. 28, 29

Bibliography

[1] S. Ghosh, "Camillo golgi (1843-1926): Scientist extraordinaire and pioneer figure of modern neurology," *Anatomy & cell biology*, vol. 53, Oct. 2020. DOI: 10.5115/acb.20.196.

[2] B. Hoeneisen and C. Mead, "Fundamental limitations in microelectronics—i. mos technology," *Solid-State Electronics*, vol. 15 (7), pp. 819–829, 1972. DOI: https://doi.org/10.1016/0038-1101(72)90103-7.

[3] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff.," *IEEE Solid-State Circuits Society Newsletter*, vol. 11 (3), pp. 33–35, 2006. DOI: 10.1109/N-SSC.2006.4785860.

[4] J. Shalf, "The future of computing beyond moore's law," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 378 (2166), p. 20 190 061, 2020. DOI: 10.1098/rsta.2019.0061. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2019.0061.

[5] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78 (10), pp. 1629–1636, Oct. 1990. DOI: 10.1109/5.58356.

[6] C. Frenkel, D. Bol, and G. Indiveri, "Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence," *Proceedings of the IEEE*, vol. 111 (6), pp. 623–652, 2023. DOI: 10.1109/JPROC.2023.3273520.

[7] A. de Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7 (10), pp. 2191–2194, 2023. DOI: https://doi.org/10.1016/j.joule.2023.09.004.

[8] C. Mead, "Neuromorphic Engineering: In Memory of Misha Mahowald," *Neural Computation*, vol. 35 (3), pp. 343–383, Feb. 2023. DOI: 10.1162/neco_a_01553. eprint: https://direct.mit.edu/neco/article-pdf/35/3/343/2072229/neco_a_01553.pdf.

[9] C. Mead, *Analog VLSI and Neural Systems* (Addison-Wesley VLSI system series). Addison-Wesley, 1989.

[10] F. Sandin, A. Khan, A. Dyer, *et al.*, "Concept learning in neuromorphic vision systems: What can we learn from insects?" eng, *Journal of Software Engineering and Applications*, vol. 7, p. 9, 2014.

[11] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, pp. 1379–1397, 2015.

[12] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10 (9), pp. 1659–1671, 1997. DOI: https://doi.org/10.1016/S0893-6080(97)00011-7.

[13] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 52 (1), pp. 99–115, 1990. DOI: https://doi.org/10.1016/S0092-8240(05)80006-0.

[14] F. B. Fitch, "Mcculloch warren s. and pitts walter. a logical calculus of the ideas immanent in nervous activity. bulletin of mathematical biophysics, vol. 5 , pp. 115–133," *Journal of Symbolic Logic*, vol. 9 (2), pp. 49–50, 2014.

[15] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," *Proceedings of ICML*, vol. 27, pp. 807–814, Jun. 2010.

[16] T. Nanami and T. Kohno, "Piecewise quadratic neuron model: A tool for close-to-biology spiking neuronal network simulation on dedicated hardware," *Frontiers in Neuroscience*, vol. 16, 2023. DOI: 10.3389/fnins.2022.1069133.

[17] T. Levi, T. Nanami, A. Tange, K. Aihara, and T. Kohno, "Development and applications of biomimetic neuronal networks toward brainmorphic artificial intelligence," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65 (5), pp. 577–581, 2018. DOI: 10.1109/TCSII.2018.2824827.

[18] M. J. Bezaire, I. Raikov, K. Burk, D. Vyas, and I. Soltesz, "Interneuronal mechanisms of hippocampal theta oscillations in a full-scale model of the rodent ca1 circuit," *eLife*, vol. 5, F. K. Skinner, Ed., e18566, Dec. 2016. DOI: 10.7554/eLife.18566.

[19] M. Chiappalone, V. Cota, M. Carè, *et al.*, "Neuromorphic-based neuroprostheses for brain rewiring: State-of-the-art and perspectives in neuroengineering," *Brain Sciences*, vol. 12, p. 1578, Nov. 2022. DOI: 10.3390/brainsci12111578.

[20] R. George, M. Chiappalone, M. Giugliano, *et al.*, "Plasticity and adaptation in neuromorphic biohybrid systems," *iScience*, vol. 23 (10), p. 101 589, 2020. DOI: https://doi.org/10.1016/j.isci.2020.101589.

[21] M. Ambroise, T. Levi, S. Joucla, B. Yvert, and S. Saïghi, "Real-time biomimetic central pattern generators in an fpga for hybrid experiments," *Frontiers in Neuroscience*, vol. 7, 2013. DOI: 10.3389/fnins.2013.00215.

[22] S. Buccelli, Y. Bornat, I. Colombi, *et al.*, "A neuromorphic prosthesis to restore communication in neuronal networks," *iScience*, vol. 19, pp. 402–414, 2019. DOI: https://doi.org/10.1016/j.isci.2019.07.046.

[23] G. Le Masson, S. Le Masson, and M. Moulins, "From conductances to neural network properties: Analysis of simple circuits using the hybrid network method," *Progress in Biophysics and Molecular Biology*, vol. 64 (2), pp. 201–220, 1995. DOI: https://doi.org/10.1016/S0079-6107(96)00004-1.

[24] 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, Apr. 2019. DOI: 10.3389/fnins.2019.00377.

[25] R. Jung, E. Brauer, and J. Abbas, "Real-time interaction between a neuromorphic electronic circuit and the spinal cord," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 9, pp. 319–26, Oct. 2001. DOI: 10.1109/7333.948461.

[26] S. Joucla, M. Ambroise, T. Levi, *et al.*, "Generation of locomotor-like activity in the isolated rat spinal cord using intraspinal electrical microstimulation driven by a digital neuromorphic cpg," *Frontiers in Neuroscience*, vol. 10, 2016. DOI: 10.3389/fnins.2016.00067.

[27] T. Levi, P. Bonifazi, P. Massobrio, and M. Chiappalone, "Editorial: Closed-loop systems for next-generation neuroprostheses," *Frontiers in Neuroscience*, vol. 12, 2018. DOI: 10.3389/fnins.2018.00026.

[28] J. P. Dominguez-Morales, S. Buccelli, D. Gutierrez-Galan, I. Colombi, A. Jimenez-Fernandez, and M. Chiappalone, "Real-time detection of bursts in neuronal cultures using a neuromorphic auditory sensor and spiking neural networks," *Neurocomputing*, vol. 449, pp. 422–434, 2021. DOI: https://doi.org/10.1016/j.neucom.2021.03.109.

[29] M. Pospischil, M. Toledo-Rodriguez, C. Monier, *et al.*, "Minimal hodgkin-huxley type models for different classes of cortical and thalamic neurons," *Biological cybernetics*, vol. 99, pp. 427–41, Dec. 2008. DOI: 10.1007/s00422-008-0263-8.

[30] J. Lengler, F. Jug, and A. Steger, "Reliable neuronal systems: The importance of heterogeneity," *PLOS ONE*, vol. 8 (12), pp. 1–10, Dec. 2013. DOI: 10.1371/journal.pone.0080694.

[31] C. Mead, *Insight 1: Logic in physical form*, Accessed 12/23, May 2018.

[32] W. J. Dally, R. C. Harting, and T. M. Aamodt, *Digital Design Using VHDL: A Systems Approach*, 1st. USA: Cambridge University Press, 2016.

[33] S. Timothy, *Introduction to fpga design for embedded systems*, https://www.coursera.org/learn/intro-fpga-design-embedded-systems, Accessed 12/23, 2011.

[34] J. O. Hamblen, T. S. Hall, and M. D. Furman, *Rapid Prototyping of Digital Systems: SOPC Edition*, 1st. Springer Publishing Company, Incorporated, 2007.

[35] P. Wilson, "Introduction," in *Design Recipes for FPGAs (Second Edition)*, P. Wilson, Ed., Second Edition, Oxford: Newnes, 2016, p. 65. DOI: https://doi.org/10.1016/B978-0-08-097129-2.09983-9.

[36] D. Pellerin and D. Taylor, *VHDL Made Easy!* USA: Prentice-Hall, Inc., 1997.

[37] K. R. Jessen, "Glial cells," *The International Journal of Biochemistry & Cell Biology*, vol. 36 (10), pp. 1861–1867, 2004. DOI: https://doi.org/10.1016/j.biocel.2004.02.023.

[38] P. Dayan and L. Abbott, *Theoretical Neuroscience*. Cambridge, MA: MIT Press, 2001.

[39] S. Sharma, G. Kumar, D. Mishra, and D. Mohapatra, "Design and implementation of a variable gain amplifier for biomedical signal acquisition," *international journal of advanced research in Computer science and software engineering*, Feb. 2012.

[40] F. Walter, F. Röhrbein, and A. Knoll, "Computation by time," *Neural Processing Letters*, vol. 44, Aug. 2016. DOI: 10.1007/s11063-015-9478-6.

[41] C. Koch and I. Segev, *Methods in Neural Modeling*. MIT Press, Jan. 2001.

[42] C. Koch, *Biophysics of Computation: Information Processing in Single Neurons (Computational Neuroscience Series)*. USA: Oxford University Press, Inc., 2004.

[43] L. Lapicque, "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation," *J. Physiol. Pathol. Gen.*, vol. 9, pp. 620–635, 1907.

[44] L. Abbott, "Lapicque's introduction of the integrate-and-fire model neuron (1907)," *Brain Research Bulletin*, vol. 50, pp. 303–304, 1999.

[45] H. Tuckwell, "Continuum models in neurobiology and information processing," *Bio Systems*, vol. 48, pp. 223–8, Nov. 1998. DOI: 10.1016/S0303-2647(98)00069-0.

[46] A. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological cybernetics*, vol. 95 (1), pp. 1–19, Jun. 2006. DOI: 10.1007/s00422-006-0068-6.

[47] G. Zheng and A. Tonnelier, "Chaotic solutions in the quadratic integrate-and-fire neuron with adaptation," *Cognitive neurodynamics*, vol. 3, pp. 197–204, Dec. 2008. DOI: 10.1007/s11571-008-9069-6.

[48] N. Fourcaud-Trocmé, D. Hansel, C. van Vreeswijk, and N. Brunel, "How spike generation mechanisms determine the neuronal response to fluctuating inputs," *Journal of Neuroscience*, vol. 23 (37), pp. 11 628–11 640, 2003. DOI: 10.1523/JNEUROSCI.23-37-11628.2003. eprint: https://www.jneurosci.org/content/23/37/11628.full.pdf.

[49] G. D. Smith, C. L. Cox, S. M. Sherman, and J. Rinzel, "Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model," *Journal of Neurophysiology*, vol. 83 (1), pp. 588–610, 2000, PMID: 10634897. DOI: 10.1152/jn.2000.83.1.588. eprint: https://doi.org/10.1152/jn.2000.83.1.588.

[50] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of Neurophysiology*, vol. 94 (5), pp. 3637–3642, 2005, PMID: 16014787. DOI: 10.1152/jn.00686.2005. eprint: https://doi.org/10.1152/jn.00686.2005.

[51] E. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14 (6), pp. 1569–1572, 2003. DOI: 10.1109/TNN.2003.820440.

[52] E. Hendrickson, J. Edgerton, and D. Jaeger, "The capabilities and limitations of conductance-based compartmental neuron models with reduced branched or unbranched morphologies and active dendrites," *Journal of computational neuroscience*, vol. 30, pp. 301–21, Apr. 2011. DOI: 10.1007/s10827-010-0258-z.

[53] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117 (4), pp. 500–544, 1952. DOI: https://doi.org/10.1113/jphysiol.1952.sp004764. eprint: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1952.sp004764.

[54] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical Journal*, vol. 1 (6), pp. 445–466, 1961. DOI: https://doi.org/10.1016/S0006-3495(61)86902-6.

[55] J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *Proceedings of the IRE*, vol. 50 (10), pp. 2061–2070, 1962. DOI: 10.1109/JRPROC.1962.288235.

[56] T. Kepler, L. Abbott, and E. Marder, "Reduction of conductance-based neuron models," *Biological cybernetics*, vol. 66, pp. 381–7, Feb. 1992. DOI: 10.1007/BF00197717.

[57] J. Rinzel and B. Ermentrout, "Analysis of neural excitability and oscillations," *Methods of Neuronal Modeling*, Jan. 1998.

[58] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[59] E. Izhikevich, "Dynamical systems in neuroscience," *MIT Press*, p. 111, Jul. 2007.

[60] M. Cunningham, M. Whittington, A. Bibbig, *et al.*, "A role for fast rhythmic bursting neurons in cortical gamma oscillations in vitro," *Prcoeedings of the National Academy of Sciences*, vol. 101 (18), pp. 7152–7157, May 2004, Copyright © 2004 by the National Academy of Sciences.

[61] J. Benda, A. Longtin, and L. Maler, "Spike-frequency adaptation separates transient communication signals from background oscillations," *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 25, pp. 2312–21, Apr. 2005. DOI: 10.1523/JNEUROSCI.4795-04.2005.

[62] S. Grillner, "Biological pattern generation: The cellular and computational logic of networks in motion," *Neuron*, vol. 52 (5), pp. 751–766, 2006. DOI: https://doi.org/10.1016/j.neuron.2006.11.008.

[63] H. Alle and J. Geiger, "Combined analog and action potential coding in hippocampal mossy fibers," *Science (New York, N.Y.)*, vol. 311, pp. 1290–3, Apr. 2006. DOI: 10.1126/science.1119055.

[64] A. L. Hodgkin, "The local electric changes associated with repetitive action in a non-medullated axon," *The Journal of Physiology*, vol. 107 (2), pp. 165–181, 1948. DOI: https://doi.org/10.1113/jphysiol.1948.sp004260. eprint: https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1948.sp004260.

[65] D. Hansel, G. Mato, and C. Meunier, "Synchrony in excitatory neural networks," *Neural Computation*, vol. 7 (2), pp. 307–337, 1995.

[66] J. Hindmarsh and R. Rose, "A model of neuronal bursting using three coupled first order differential equations," *Proceedings of the Royal Society of London. Series B, Containing papers of a Biological character. Royal Society (Great Britain)*, vol. 221, pp. 87–102, Apr. 1984. DOI: 10.1098/rspb.1984.0024.

[67] T. Kohno and K. Aihara, "Digital spiking silicon neuron: Concept and behaviors in gj-coupled network," in *International Symposium on Artificial Life and Robotics 2007*, Jan. 2007.

[68] T. Nanami and T. Kohno, "Simple cortical and thalamic neuron models for digital arithmetic circuit implementation," *Frontiers in Neuroscience*, vol. 10, 2016. DOI: 10.3389/fnins.2016.00181.

[69] T. Nanami, K. Aihara, and T. Kohno, "Elliptic and parabolic bursting in a digital silicon neuron model," in *International Symposium on Nonlinear Theory and Its Applications, NOLTA2016*, Nov. 2016.

[70] B. W. Connors and M. J. Gutnick, "Intrinsic firing patterns of diverse neocortical neurons," *Trends in Neurosciences*, vol. 13 (3), pp. 99–104, 1990. DOI: https://doi.org/10.1016/0166-2236(90)90185-D.

[71] A. Destexhe, M. Neubig, D. Ulrich, and J. Huguenard, "Dendritic low-threshold calcium currents in thalamic relay cells," *Journal of Neuroscience*, vol. 18 (10), pp. 3574–3588, 1998. DOI: 10.1523/JNEUROSCI.18-10-03574.1998. eprint: https://www.jneurosci.org/content/18/10/3574.full.pdf.

[72] E. M. Izhikevich, "Subcritical elliptic bursting of bautin type," *SIAM Journal on Applied Mathematics*, vol. 60 (2), pp. 503–535, 1999.

[73] G. B. Ermentrout and N. Kopell, "Parabolic bursting in an excitable system coupled with a slow oscillation," *SIAM Journal on Applied Mathematics*, vol. 46 (2), pp. 233–253, 1986.

[74] T. Kohno, M. Sekikawa, J. Li, T. Nanami, and K. Aihara, "Qualitative-modeling-based silicon neurons and their networks," *Frontiers in Neuroscience*, vol. 10, 2016. DOI: 10.3389/fnins.2016.00273.

[75] J. M. Bower, H. Cornelis, and D. Beeman, "Genesis, the general neural simulation system," in *Encyclopedia of Computational Neuroscience*. New York, NY: Springer New York, 2013, pp. 1–8. DOI: 10.1007/978-1-4614-7320-6_255-1.

[76] N. T. Carnevale and M. L. Hines, *The NEURON Book*. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511541612.

[77] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, p. 1430, Jan. 2007. DOI: 10.4249/scholarpedia.1430.

[78] D. Goodman and R. Brette, "Brian: A simulator for spiking neural networks in python," *Frontiers in Neuroinformatics*, vol. 2, 2008. DOI: 10.3389/neuro.11.005.2008.

[79] F. Zenke and W. Gerstner, "Limits to high-speed simulations of spiking neural networks using general-purpose computers," *Frontiers in neuroinformatics*, vol. 8, p. 76, Sep. 2014. DOI: 10.3389/fninf.2014.00076.

[80] S. Panagiotou, H. Sidiropoulos, D. Soudris, M. Negrello, and C. Strydis, "Eden: A high-performance, general-purpose, neuroml-based neural simulator," *Frontiers in Neuroinformatics*, vol. 16, 2022. DOI: 10.3389/fninf.2022.724336.

[81] J. Vitay, H. Dinkelbach, and F. Hamker, "Annarchy: A code generation approach to neural simulations on parallel hardware," *Frontiers in Neuroinformatics*, vol. 9, 2015. DOI: 10.3389/fninf.2015.00019.

[82] E. Yavuz, J. Turner, and T. Nowotny, "Genn: A code generation framework for accelerated brain simulations," *Scientific Reports*, vol. 6, p. 18 854, Jan. 2016. DOI: 10.1038/srep18854.

[83] M. Stimberg, R. Brette, and D. F. Goodman, "Brian 2, an intuitive and efficient neural simulator," *eLife*, vol. 8, F. K. Skinner, R. L. Calabrese, F. K. Skinner, F. Zeldenrust, and R. C. Gerkin, Eds., e47314, Sep. 2019. DOI: 10.7554/eLife.47314.

[84] N. Qiao, H. Mostafa, F. Corradi, *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in Neuroscience*, vol. 9, 2015. DOI: 10.3389/fnins.2015.00141.

[85] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps)," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, Aug. 2017. DOI: 10.1109/TBCAS.2017.2759700.

[86] B. V. Benjamin, P. Gao, E. McQuinn, *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, pp. 699–716, 2014.

[87] J.-s. Seo, B. Brezzo, Y. Liu, *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp. 1–4, Sep. 2011. DOI: 10.1109/CICC.2011.6055293.

[88] 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 28nm cmos," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, pp. 145–158, Feb. 2019. DOI: 10.1109/TBCAS.2018.2880425.

[89] C. Frenkel, J.-D. Legat, and D. Bol, "Morphic: A 65-nm 738k-synapse/mm 2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, pp. 999–1010, Oct. 2019. DOI: 10.1109/TBCAS.2019.2928793.

[90] E. Painkras, L. A. Plana, J. Garside, *et al.*, "Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48 (8), pp. 1943–1953, 2013. DOI: 10.1109/JSSC.2013.2259038.

[91] M. Davies, N. Srinivasa, T.-H. Lin, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. PP, pp. 1–1, Jan. 2018. DOI: 10.1109/MM.2018.112130359.

[92] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345 (6197), pp. 668–673, 2014. DOI: 10.1126/science.1254642. eprint: https://www.science.org/doi/pdf/10.1126/science.1254642.

[93] A. Cassidy, P. Merolla, J. Arthur, *et al.*, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," *Proceedings of the International Joint Conference on Neural Networks*, Aug. 2013. DOI: 10.1109/IJCNN.2013.6707077.

[94] D. Thomas and W. Luk, "Fpga accelerated simulation of biologically plausible spiking neural networks," *17th IEEE Symposium on Field Programmable Custom Computing Machines*, pp. 45–52, Jan. 2009. DOI: 10.1109/FCCM.2009.46.

[95] A. Cassidy, A. G. Andreou, and J. Georgiou, "Design of a one million neuron single fpga neuromorphic system for real-time multimodal scene analysis," in *2011 45th Annual Conference on Information Sciences and Systems*, 2011, pp. 1–6. DOI: 10.1109/CISS.2011.5766099.

[96] J. Li, Y. Katori, and T. Kohno, "An fpga-based silicon neuronal network with selectable excitability silicon neurons," *Frontiers in Neuroscience*, vol. 6, 2012. DOI: 10.3389/fnins.2012.00183.

[97] D. Neil, "Minitaur, an event-driven fpga-based spiking network accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, p. 1, Jan. 2014. DOI: 10.1109/TVLSI.2013.2294916.

[98] J. Luo, G. Coapes, T. Mak, T. Yamazaki, C. Tin, and P. Degenaar, "Real-time simulation of passage-of-time encoding in cerebellum using a scalable fpga-based system," *IEEE transactions on biomedical circuits and systems*, vol. 10, Oct. 2015. DOI: 10.1109/TBCAS.2015.2460232.

[99] S. Yang, J. Wang, B. deng, *et al.*, "Real-time neuromorphic system for large-scale conductance-based spiking neural networks," *IEEE Transactions on Cybernetics*, vol. 49, pp. 2490–2503, Apr. 2019. DOI: 10.1109/TCYB.2018.2823730.

[100] R. Stein, E. Gossen, and K. Jones, "Neuronal variability: Noise or part of the signal?" *Nature Reviews Neuroscience*, vol. 6, pp. 389–397, Jan. 2005.

[101] N. Urban and S. Tripathy, "Neuroscience: Circuits drive cell diversity," *Nature*, vol. 488, pp. 289–90, Aug. 2012. DOI: 10.1038/488289a.

[102] V. Balasubramanian, "Heterogeneity and efficiency in the brain," *Proceedings of the IEEE*, vol. 103, pp. 1346–1358, Aug. 2015. DOI: 10.1109/JPROC.2015.2447016.

[103] N. Perez-Nieves, V. Leung, P. Dragotti, and D. Goodman, "Neural heterogeneity promotes robust learning," *Nature Communications*, vol. 12, p. 5791, Oct. 2021. DOI: 10.1038/s41467-021-26022-3.

[104] D. Zendrikov, S. Solinas, and G. Indiveri, "Brain-inspired methods for achieving robust computation in heterogeneous mixed-signal neuromorphic processing systems," *Neuromorphic Computing and Engineering*, vol. 3, Jul. 2023. DOI: 10.1088/2634-4386/ace64c.

[105] M. R. Pillmeier, M. J. Schulte, and E. G. Walters, "Design alternatives for barrel shifters," in *SPIE Optics + Photonics*, 2002.

[106] M. Mahowald, "The silicon retina," *Sci. Am.*, vol. 264, pp. 4–65, Jul. 2011. DOI: 10.1007/978-1-4615-2724-4_2.

[107] R. Lyon and C. Mead, "An analog electronic cochlea," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36 (7), pp. 1119–1134, 1988. DOI: 10.1109/29.1639.

[108] S.-C. Liu, A. van Schaik, B. A. Minch, and T. Delbruck, "Event-based 64-channel binaural silicon cochlea with q enhancement mechanisms," in *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 2027–2030. DOI: 10.1109/ISCAS.2010.5537164.

[109] F. Corradi, D. Zambrano, M. Raglianti, G. Passetti, C. Laschi, and G. Indiveri, "Towards a neuromorphic vestibular system," *IEEE Transactions on Biomedical Circuits and Systems*, vol. PP, p. 1, Oct. 2014. DOI: 10.1109/TBCAS.2014.2358493.

[110] T. Koickal, A. Hamilton, S. Tan, J. Covington, J. Gardner, and T. Pearce, "Analog vlsi circuit implementation of an adaptive neuromorphic olfaction chip," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 54, pp. 60–73, Feb. 2007. DOI: 10.1109/TCSI.2006.888677.

[111] C. Bartolozzi, "Neuromorphic circuits impart a sense of touch," *Science (New York, N.Y.)*, vol. 360, pp. 966–967, Jun. 2018. DOI: 10.1126/science.aat3125.

[112] K. Yamazaki, K. Vo, and D. Bulsara, "Spiking neural networks and their applications: A review," *Brain sciences*, vol. 12, Jun. 2022. DOI: 10.3390/brainsci12070863.

[113] P. Katz, "Evolution of central pattern generators and rhythmic behaviours," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 371, p. 20 150 057, Jan. 2016. DOI: 10.1098/rstb.2015.0057.

[114] H. Markram, E. Muller, S. Ramaswamy, *et al.*, "Reconstruction and simulation of neocortical microcircuitry," *Cell*, vol. 163, pp. 456–492, Oct. 2015. DOI: 10.1016/j.cell.2015.09.029.

[115] A. Ecker, A. Romani, S. Sáray, *et al.*, "Data-driven integration of hippocampal ca1 synaptic physiology in silico," *Hippocampus*, vol. 30, Jun. 2020. DOI: 10.1002/hipo.23220.