Design and Optimization of Temporal Encoders Using Integrate-and-Fire and Leaky Integrate-and-Fire Neurons

Juliet Anderson

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

Master of Science
In
Electrical Engineering

Yang Yi, Chair
Jeffrey S. Walling, Member
Timothy J. Talty, Member

July 5, 2022
Blacksburg, VA

Keywords: Neuromorphic Computing, Spiking Neural Network, Temporal Encoder, Latency, ISI, IF neuron, LIF neuron, Analog, Circuit Design

Design and Optimization of Temporal Encoders Using Integrate-and-Fire and Leaky Integrate-and-Fire Neurons

Abstract

As Moore's law nears its limit, a new form of signal processing is needed. Neuromorphic computing has used inspiration from biology to produce a new form of signal processing by mimicking biological neural networks using electrical components. Neuromorphic computing requires less signal preprocessing than digital systems since it can encode signals directly using analog temporal encoders from Spiking Neural Networks (SNNs). These encoders receive an analog signal as an input and generate a spike or spike trains as their output. The proposed temporal encoders use latency and Inter-Spike Interval (ISI) encoding and are expected to produce a highly sensitive hardware implementation of time encoding to preprocess signals for dynamic neural processors. Two ISI and two latency encoders were designed using Integrate-and-Fire (IF) and Leaky Integrate-and-Fire (LIF) neurons and optimized to produce low area designs.

The IF and LIF neurons were designed using the Global Foundries 180nm CMOS process and achieved an area of 186μm^2 and 182μm^2, respectively. All four encoders have a sampling frequency of 50kHz. The latency encoders achieved an average energy consumption per spike of 277nJ and 316pJ for the IF-based and LIF-based latency encoders, respectively. The ISI encoders achieved an average energy consumption per spike of 1.07uJ and 901nJ for the IF-based and LIF-based ISI encoders, respectively. Power consumption is proportional to the number of neurons employed in the encoder and the potential to reduce power consumption through layout-level simulations is presented. The LIF neuron is able to use a smaller membrane capacitance to achieve similar operability as the IF neuron and consumes less area despite having more components. This demonstrates that capacitor sizes are the main limitations of a small size in spiking neurons for SNNs. An overview of the design and layout process of the two presented neurons is discussed with tips for overcoming problems encountered. The proposed designs can result in a fast neuromorphic process by employing a frequency higher than 10kHz and by providing a hardware implementation that is efficient in multiple sectors like machine learning, medical implementations, or security systems since hardware is safer from hacks.

Design and Optimization of Temporal Encoders Using Integrate-and-Fire and Leaky Integrate-and-Fire Neurons

General Audience Abstract

As Moore's law nears its limit, a new form of signal processing is needed. Moore's law anticipated that transistor sizes will decrease exponentially as the years pass but CMOS technology is reaching physical limitations which could mean an end to Moore's prediction. Neuromorphic computing has used inspiration from biology to produce a new form of signal processing by mimicking biological neural networks using electrical components. Biological neural networks communicate through interconnected neurons that transmit signals through synapses. Neuromorphic computing uses a subdivision of Artificial Neural Networks (ANNs) called Spiking Neural Networks (SNNs) to encode input signals into voltage spikes to mimic biological neurons. Neuromorphic computing reduces the preprocessing step needed to process data in the digital domain since it can encode signals directly using analog temporal encoders from SNNs. These encoders receive an analog signal as an input and generate a spike or spike trains as their output. The proposed temporal encoders use latency and Inter-Spike Interval (ISI) encoding and are expected to produce a highly sensitive hardware implementation of time encoding to preprocess signals for dynamic neural processors. Two ISI and two latency encoders were designed using Integrate-and-Fire (IF) and Leaky Integrate-and-Fire (LIF) neurons and optimized to produce low area designs.

All four encoders have a sampling frequency of 50kHz. The latency encoders achieved an average energy consumption per spike of 277nJ and 316pJ for the IF-based and LIF-based latency encoders, respectively. The ISI encoders achieved an average energy consumption per spike of 1.07uJ and 901nJ for the IF-based and LIF-based ISI encoders, respectively. Power consumption is proportional to the number of neurons employed in the encoder and the potential to reduce power consumption through layout-level simulations is presented. The LIF neuron is able to use a smaller membrane capacitance to achieve similar operability which consumes less area despite having more components than the IF neuron. This demonstrates that capacitor sizes are the main limitations of small size in neurons for spiking neural networks. An overview of the design and layout process of the two presented neurons is discussed with tips for overcoming problems encountered. The proposed designs can result in a fast neuromorphic process by employing a frequency higher than 10kHz and by providing a hardware implementation that is efficient in multiple sectors like machine learning, medical implementations, or security systems since hardware is safer from hacks.

Acknowledgements

I would like to thank my committee members. Thank you, Dr. Yi, for your support and encouragement! You have helped me build confidence in my work and were a great support to me when I felt lost at the beginning of my graduate journey. Thank you, Dr. Walling, for single-handedly teaching me analog circuit design and for having the patience to answer all of my questions! Analog circuit design is very confusing, and I very much appreciate your help throughout my research. Thank you Dr. Talty for inspiring me during your undergraduate class to pursue graduate school and giving me the confidence of knowing I had a faculty member willing to support me through it! Also, thank you for your career tips I will definitely be using them in future interviews and negotiations. Thank you, committee members!

I would also like to thank the students at the VT MICS lab. I would like to specifically thank PhD students Honghao and Fabiha for walking me through machine learning at the start of my degree and for answering all my questions throughout my year at MICS. Thank you, VT MICS lab mates, for distracting me when anxiety was weighing me down and for being so supportive all the time! I've never been part of such a great lab environment, I appreciate every single one of you. Lastly, I would like to thank my parents and my VT friends. Thank you for helping me stay sane during hard times and for giving me the confidence of knowing I can do anything I set my mind to! I love you all! I can't wait to hangout with you all without having to worry about homework!

Table of Contents

Acknowledgements......iv
Table of Contents...... v
List of Figures vii
List of Tables ix
Chapter 1: Introduction 1
1.1 Technology Comparisons: Neuromorphic Computing Overview 2
1.1.1 Why Neuromorphic Computing?...... 3
1.1.2 Why Mixed Signal? 5
1.2 Neuromorphic Computing Background 6
1.2.1 Neuromorphic Computing Timeline...... 7
1.3 Inspiration from Biology...... 7
Chapter 2: Spiking Neural Network & Neural Encoders...... 9
2.1 Why Neural Encoding? 9
2.2 Neural Encoding Overview...... 10
Chapter 3: Spiking Neurons 11
3.1 Integrate-and-Fire & Leaky Integrate-and-Fire Neurons Overview...... 12
Chapter 4: Experimental Results...... 12
4.1 IF Neuron Design & Layout...... 13
4.1.1 IF Circuit Sizing...... 16
4.1.2 IF Neuron Layout 22
4.2 LIF Neuron Design and Layout 26
4.2.1 IF Circuit Sizing...... 29
4.2.2 LIF Layout...... 32
4.3 Temporal Encoding...... 33
4.4 Latency Encoding Results 34
4.4.1 IF-based Latency Encoder...... 35
4.4.2 LIF-based Latency Encoder 36
4.5 ISI Encoder Results 39
4.5.1 IF-based ISI encoder 40
4.5.2 LIF-based ISI encoder...... 41

4.6 Comparisons 42
Chapter 5: Conclusion and Future Work 44
5.1 Conclusion 44
5.2 Future Work 45
References 46

List of Figures

Figure 1.1 Pros and Cons of Von-Neumann Computing vs. Neuromorphic Computing……….4

Figure 1.2 Biological neuron components and examples of hardware equivalents inspiring artificial neural networks for neuromorphic computing…………………………………..8

Figure 1.3 Biological neuron action potential……………………………………………………………8

Figure 2.1 Neural encoding schemes (voltage vs time) ……………………………………………10

Figure 4.1 IF neuron circuit schematic……………………………………………………………………14

Figure 4.2 Transient simulation of IF neuron for one sampling period. In black, input clock signal triggers sampling period. In red, output Vspike is the generated voltage spike. In green, the membrane voltage charging/discharging. In blue, input analog current…………………………………………………………………………………………15

Figure 4.3 CMOS inverter on left. On right, Case 1 represents a pull down (PDN) network, Case 2 represents a pull up (PUP) network……………………………………………………17

Figure 4.4 Operation regions of CMOS transistors………………………………………………………19

Figure 4.5 Inversion regions of CMOS operation…………………………………………………………21

Figure 4.6 IF neuron inversion region summary based on Gm/Id……………………………………21

Figure 4.7 IF neuron layout……………………………………………………………………………………23

Figure 4.8 Layout of PMOS transistors A and B…………………………………………………………24

Figure 4.9 LIF neuron circuit schematic……………………………………………………………………27

Figure 4.10 Transient simulation of LIF neuron for one sampling period. In black, input clock signal triggers sampling period. In red, output Vspike is the generated voltage spike. In green, the membrane voltage charging/discharging. In blue, input analog current…………………………………………………………………………………………28

Figure 4.11 LIF neuron inversion region summary based on Gm/Id…………………………………31

Figure 4.12 LIF neuron layout………………………………………………………………………………….32

Figure 4.13 Temporal encoding using the integration time of spikes………………………………..33

Figure 4.14 Transient simulation of IF-based Latency encoder for a full wave period. In black, input clock signal. In red, output spike train of latency encoder. In blue, input analog current…………………………………………………………………………………………36

Figure 4.15 Transient simulation of LIF-based Latency encoder for a full wave period. In black, input clock signal. In red, output spike train of latency encoder. In blue, input analog current……………………………………………………………………………………36

Figure 4.16 Case 1 Vleak adjustment: Purple signal represents overcharged membrane voltage without triggering a spike; needs Vleak adjustment……………………………..37

Figure 4.17 Case 2 Vleak adjustment 1: Blue signal represents overcharged membrane voltage. Red signal represents respective output voltage spikes. Vleak adjustments needed……………………………………………………………………………………38

Figure 4.18 Case 2 Vleak adjustment 2: Purple signal represents overcharged membrane voltage. Green signal represents a three-neuron ISI spike train output. Vleak adjustments needed since purple signal doesn't trigger spikes at low input current amplitudes……………………………………………………………………………………38

Figure 4.19 Case 2 Vleak adjustment 3: Blue signal represents overcharged membrane voltage. Red signal represents respective output voltage spikes………………………39

Figure 4.20 IF-based ISI encoder circuit schematic…………………………………………………….....40

Figure 4.21 Transient simulation of IF-based ISI encoder for a full wave period. In black, input clock signal. In red, output spike train of ISI encoder. In blue, input analog current……………………………………………………………………………………………………..41

Figure 4.22 Transient simulation of LIF-based ISI encoder for a full wave period. In black, input clock signal. In red, output spike train of ISI encoder. In blue, input analog current……………………………………………………………………………………………………..41

List of Tables

Table 4.1 IF neuron CMOS sizes……………………………………………………………………………………20

Table 4.2 Important layout design rules to keep in mind during layout process……………….26

Table 4.3 LIF neuron CMOS sizes…………………………………………………………………………………31

Table 4.4 Latency encoders input parameters……………………………………………………………….34

Table 4.5 ISI encoders input parameters……………………………………………………………………..39

Table 4.6 Design comparisons……………………………………………………………………………………..43

Table 4.7 Latency encoder integration times for a full wave period & ISI encoder intervals D1 and D2 for a full wave period……………………………………………………………………44

Chapter 1: Introduction

As advances in traditional computer architecture reach the limit of Moore's law, new technologies emerge to become an effective replacement in complex computational processes. In the early 1970's, Moore's law predicted that the number of transistors per square inch of a chip will double every one and a half years. This proved to be true until recent years have shown that the physical limitations of CMOS technologies have almost been reached as it continues to scale down. It is becoming harder, almost impossible, to continue the trend of Moore's law while keeping low-energy demands. But increasing transistor density in a chip is not the only way to make computer architectures more powerful and efficient. This has led to the increased research of emerging technologies by improving software algorithms through effectively using current transistor densities or using computer architectures that differ from the traditional Von Neumann computers and slowing down the downscaling of transistors. A desirable emerging computing architecture is neuromorphic computing as it has been increasingly researched throughout the last decade due to its ability to overcome traditional computer limitations in data-intensive applications. Neuromorphic computing mimics the information processing of neurons through artificial neural networks (ANN) and requires signal conditioning to process the encoded signal. It was developed from the inspiration of the most complex computer known to man, the brain. Scientists still don't fully understand how the human brain functions but advances in the research of biological neural systems inspired engineers to mimic biological neurons with mixed-signal circuits. The human brain uses much less power to process information when compared to traditional computers. Similarly, neuromorphic computing has better power efficiency and can outperform traditional computers in complex applications like machine learning and Artificial Intelligence (AI). Neuromorphic computing has proved to be very useful in pattern recognition and is made up of very large-scale integration (VLSI) systems [1]. Spiking Neural Networks (SNN) are a subsection of ANNs often used to condition sensory data using temporal encoders for later processing or to decode the processed signal.

The brain communicates by firing electric pulses with a height of a few millivolts and duration of a few hundred microseconds. In digital systems, tens of probes are needed to monitor brain activity and use electrodes to sense the series of pulses emanating from the brain to produce an analog signal based on the sensory information. This signal now needs to be amplified, digitized, and transmitted wirelessly to a digital processor to allow for patient mobility. This process must consume low power so a small battery can be employed for days or weeks and to minimize heat dissipation in chips and make it harmless for the user [2]. As CMOS technologies continue to scale down, these energy demands cannot be met, and alternatives are considered. The temporal encoders from SNNs can condition analog sensory inputs directly into voltage spikes without the need for signal-conversion circuits like an analog-to-digital converter (ADC), reducing area and power consumption significantly. The emerging field of neuromorphic computing hasn't been as extensively researched as the traditional computer architecture and only Virginia Tech has provided analog integrated circuit (IC) implementations of the Inter-spike Interval (ISI) temporal encoder. Through the inspiration of biological neurons, temporal encoders are investigated for their ability to convert analog signals into voltage spikes, as a signal conditioner for neuromorphic computing. Four low-area temporal encoders are designed with a sampling frequency of 50 kHz, faster than firing frequency of biological neurons, to be able to distinguish small time differences and achieve a fast overall neuromorphic process for data-intensive machine learning problems. For the temporal encoders, Integrate-and-Fire (IF) and Leaky Integrate-and-Fire (LIF) neurons are optimized to produce detectable spikes with a width of 10ns or longer, while maintaining the lowest power consumption possible. The design area of the encoders is optimized by creating the layout of the CMOS components within the passive layers of the capacitor while keeping in mind the possible electric field effects. This design prioritizes a small area over lower power consumption which differs from the current approach of most artificial neuron designs to contribute further to the research knowledge of the emerging field of neuromorphic computing.

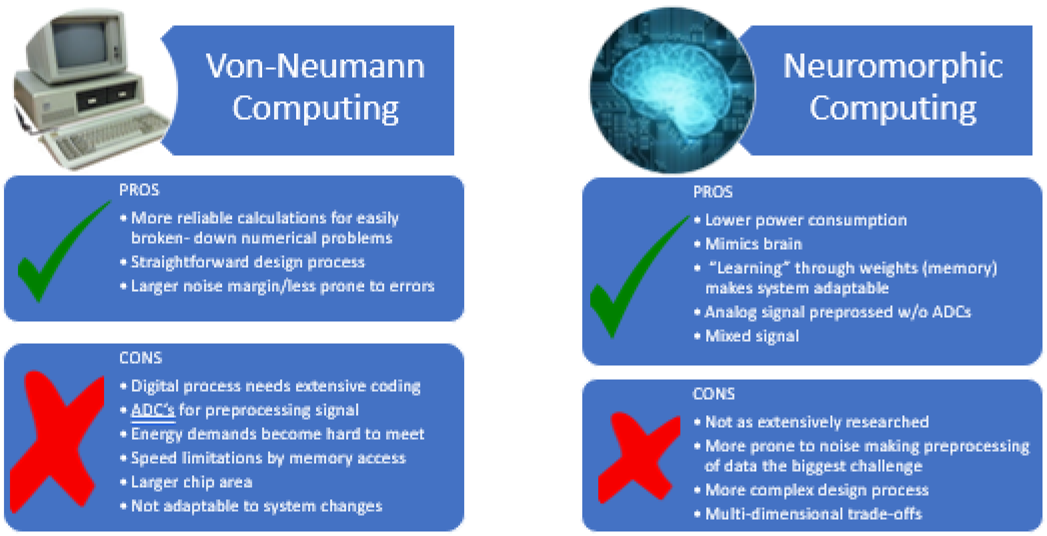
In the proposed encoders, predetermined artificial neurons map an analog input signal into an output spike train using the time-dependence of the signals to condition the input signal for a dynamic neural processor. The area and power of the neurons are optimized and designed using the CMOS GlobalFoundries 180 nm technology. The power of the temporal encoders is proportional to the number of neurons used and post-layout simulations show that the IC layout of the neurons has the ability to reduce power consumption significantly. The post-layout simulations of the IF-based latency encoder (one neuron used) showed a power consumption of 73.5mW which is proportional to the power consumption of the pre-layout simulations of the IF-based ISI encoder (three neurons used) which had a power consumption of 214mW. But the post-layout simulation of the LIF-based latency encoder (one neuron used) showed a power consumption of 155uW which is not proportional to the power consumption of the pre-layout simulations of the LIF-based ISI encoder (three neurons used) which had a power consumption of 158mW. The LIF-based encoders consumed less power than the IF-based encoders and the LIF-based latency encoder achieved the best area and power consumption comparable to current encoder designs, further discussion on Chapter 4. The layout of the neurons suggests a significant power reduction can be achieved by keeping the PMOS transistor of the spike-generating inverter outside of the capacitors' electric field effects. The IF-based encoders achieved longer integration times at the expense of a lower noise margin, higher power consumption, and higher area consumption. Overall, the encoders show high sensitivity to small input current amplitude changes by providing a varying range of integration times as the input current changes.

1.1 Technology Comparisons: Neuromorphic Computing Overview

The last decade has seen an increasing interest in AI. AI is the theory and creation of computer systems capable of doing tasks that require human intelligence like pattern recognition or decision-making [3]. AI benefits greatly from machine learning since it enables autonomy. AI has the potential to facilitate every-day living by using knowledge-representation machine learning to make decisions and take actions such as deleting spam emails, predicting science experiments, making networking suggestions, and improving healthcare by predicting how patients react to drugs. Achieving machine learning through traditional computer architectures has shown to be extremely power consuming due to large amounts of dynamic data so the benefits and setbacks of neuromorphic computing are explored as a possible replacement for traditional computing. Neuromorphic computing uses analog, digital, and mixed-signal VLSI to implement ANNs that mimic the information processing of the brain, consuming much less power than digital systems [3].

1.1.1 Why Neuromorphic Computing?

Traditional chips perform Boolean logic and period operations to reliably make calculations for any problem that can be easily broken down into a numerical problem, with more complex applications requiring more power. A traditional digital computer with machine learning capabilities can consume more than 20MW of power. Increasingly, applications are becoming difficult to perform, despite the usage of clusters of powerful traditional computers. In contrast, a human brain runs on approximately 20W of power [3]. This huge gap in power consumption inspired neuromorphic computing. Taking inspiration from multiple scientific areas like biology, physics, and engineering, neuromorphic computing has been able to mimic the data processing of biological nervous systems including its low power consumption. Neuromorphic computers will most likely replace traditional computers in data-intensive applications due to its ability to more realistically model neurons and parallelizable connection density [4]. Electronic systems receive signals and processes them to extract information. So, most computer problems require extensive data processing and input data usually comes in the form of sensory information, natural signals [2]. These input sensory information signals are analog and require analog circuits to condition it through sensors, receivers, filters, etc, for later processing [3]. Examples are cell phones receiving RF signals and returning voice or data information [2]. Traditional computing relies heavily on digital information processing and requires the conversion of analog signals into digital ones to be able to use memory units. So, as CMOS technology scales down, the Von Neumann architecture bottleneck is evident, energy demands are harder to meet. As an example of the architectural limitations of digital processors in machine learning, in a Google Corporation attempt to add smart features to their technology, they needed 16,000 processors for identifying a cat, which significantly increases power consumption when compared to the three processors needed when no smart features are present [5]. So, for AI applications, digital computers need a lot more components than analog circuits due to the need to preprocess signals by converting them to the digital domain using ADCs. This increases both area and power consumption. Additionally, ADCs require a sampling frequency lower than what analog circuits use to optimize power. The Nyquist criterion for ADCs suggest sampling frequency should be double the frequency of the input signal to optimize power. Analog encoders on the other hand, can have a frequency five times faster than the input signal yet remain at a lower area and power consumption compared to



Von-Neumann Computing

PROS
- More reliable calculations for easily broken- down numerical problems
- Straightforward design process
- Larger noise margin/less prone to errors

CONS
- Digital process needs extensive coding
- ADC's for preprocessing signal
- Energy demands become hard to meet
- Speed limitations by memory access
- Larger chip area
- Not adaptable to system changes

Neuromorphic Computing

PROS
- Lower power consumption
- Mimics brain
- "Learning" through weights (memory) makes system adaptable
- Analog signal preprossed w/o ADCs
- Mixed signal

CONS
- Not as extensively researched
- More prone to noise making preprocessing of data the biggest challenge
- More complex design process
- Multi-dimensional trade-offs

Figure 1.1 Pros and Cons of Von-Neumann Computing vs. Neuromorphic Computing [4, 6, 7]

ADCs due to their ability to encode signals directly without conversion. Analog signals are more prone to noise which makes the biggest challenge in neuromorphic computing to be the conversion of raw sensory information into a pre-processed version that is represented by action potentials or spikes, further discussed in Section 1.3 [4].

Traditional Von Neumann computer architecture consists of four functional units: the memory, the control processing unit, arithmetic and logic unit, and data paths [5]. The speed of the system is limited by the speed of accessing memory [8]. On the other hand, neuromorphic computing has interconnected processing and memory units which allows for the quick updating of weights (memory) for system adaptability. This overcomes the speed limitation of traditional computers. Unlike traditional computers, neuromorphic computing doesn't require extensive code for its operation since it employs "learning" techniques which makes it more adaptable than traditional Von Neumann computing, further discussed in Chapter 2. Neuromorphic computing uses ANNs to employ evolutionary learning and shows great adaptability in complex applications with extensive sensory data that require system changes during inputs variations and efficiently predict and detect data [3, 4]. Due to the lack of a lot of analog circuits, traditional computers have a more straightforward design process than the very complex design of the mixed signal circuits of neuromorphic VLSI systems. Although simpler, traditional computers are not adaptable to system changes since it requires extensive coding for its operability. On the other hand, due to the use of more analog circuits, VLSI mimics the adaptability of neurons but requires a more complex design due to the multi-dimensional trade-offs of analog circuits. This area also hasn't been as extensively researched so a lot of progress is possible. Computers will help research and understand the brain and the brain will help us build more efficient and powerful computers [3]. The following areas would benefit from neuromorphic computing: image processing, motion detection, pattern formation and recognition, robotics, bioinformatics, and sensor networks [4]. The summary of the comparison between traditional Von Neumann computing and neuromorphic computing can be seen in Figure 1.1. The biggest advantage of neuromorphic computing is the increased use of mixed signal circuits, so why has this improved performance?

1.1.2 Why mixed signal?

ICs can be either analog or digital. To simplify distinction, a signal is any detectable value of voltage, current, or charge and conveys information about the state of behavior of a physical system. An analog signal is defined over a continuous range of time and continuous range of amplitudes. A digital signal is defined only at discrete values of amplitude [9]. Digital circuits operate using binary stable states which lead to regularity in the system and the possibility of defining the circuit functions using algebra. This makes digital circuits less prone to errors and allows for a large noise margin. Analog signals use the time-dependence of signals and define the circuit functions with nonlinear equations [10]. Mixed signal systems, like VLSI, employ both digital and analog circuits to benefit from the advantages of both while using them to mitigate each other's fall backs.

To design an analog circuit, circuit models and simulations must be done and pose the biggest challenge in the design process. Analog design is based on experience and intuition to use simulators and is considered more "hands-on". Whereas digital circuits can enjoy Computer-Aided Design (CAD) methods which automate the design of digital circuits given certain parameters for a desired behavior [9]. Therefore, the design process of analog circuits is more complex than of digital circuits. The main trade-offs used in digital design are power consumption, speed of the circuit, and the chip area it requires [3]. In contrast, analog circuit design has a multi-dimensionality to its trade-offs. This means that by adjusting one of the trade-offs, another one will be sacrificed. The multi-dimensional trade-offs of analog design are power consumption, speed of the circuit, gain, precision, and the voltage of the power supply [3]. This multi-dimensionality is what leads analog circuits to be prone to error from noise. Despite digital signals being more reliable, analog circuits are very essential, relevant, and challenging and will continue being so for decades [2]. Analog circuits are mostly used for signal processing and rarely stand alone. Raw sensory data requires a high-performing analog circuit so input signals like sensor outputs are analog and traditionally require preprocessing using filters or ADCs that can perform with strict speed and accuracy for digital processing. For example, an antenna receives an RF signal that is digitized by an ADC and processed in the digital domain [2]. An ADC consumes a lot of power when digitizing miniscule RF signals while preventing noise effects when compared to cell phone receivers. The most important considerations for signal processing are the bandwidth of the input signal, cost, and integration so digital circuits are less efficient in processing data due to the need for domain conversion into digital [9].

In the past, data processing systems used digital processors and required multiple ICs with lots of passive components, but VLSI integrates CMOS technology using more analog circuits with digital ones to make a more efficient design [9]. For high data rates, around tens of gigabits per second, it has been proven that it's better to process data in the analog domain using an analog equalizer. For lower data rates, ADCs are traditionally used because they are more reliable and data processing is done in the digital domain [2]. Neuromorphic computing aims to employ analog circuits for low and high data rates that are much less power consuming than traditional preprocessing methods. These circuits are the temporal encoders of SNNs, further discussed in Chapter 2. VLSI is widely used in neuromorphic computing because it uses the best features of analog and digital to create an optimized mixed-signal system. CMOS technology is used for most VLSI systems since it provides density and power savings in the digital portions and allows for a mix of components in the analog circuits. In neuromorphic computing, it has been shown that replacing digital circuits with analog ones to create a VLSI system led to lower power consumption with comparable reliability.

1.2 Neuromorphic Computing Background

Neuromorphic computing consists of ANNs which are made up of layers of neuron clusters that send signals to each other and other layers. The two most commonly used neural networks in neuromorphic computing are the Feedforward Neural Network (FNN) and the Recurrent Neural Network (RNN) [7]. The FNN has connections between subsequent layers. The RNN builds on this architecture and includes random connections between layers to imitate the temporal behavior of biological nervous systems. Due to this characteristic, RNNs benefit from the use of SNNs. Some of the emerging applications that employ RNNs and SNNs include an energy efficient reservoir computing platform for 5G, a real time Damping Reduction Factor (DRF) based false data injection detection in smart grids, and improving animal welfare through smart farms [3].

Neural networks can be implemented using different technology styles and designs like a custom IC design, Field Programmable Gate Array (FPGA), or emerging devices. A custom design comes from a long design cycle and mainly involves manual designs like microprocessors, CPUs, and analog circuits. These cells are represented by Application Specific Integrated Circuits (ASIC). FPGA uses standard cells that come pre-designed and usually require a shorter design cycle since CAD software facilitates the design process. FPGA can create a fast automated design for a low cost since it is based on pre-designed cells. This allows for easy prototyping and reconfiguration of the system postproduction. Emerging devices like memristors and FinFETs can be implemented in neural networks by replacing traditional CMOS components for increased operability [11]. ASIC and FPGA implementations of neural networks are the most common. ASIC requires an extensive design process since it must be designed from the behavioral description of the circuit to the physical layout. After the design process, expensive and time-consuming manufacturing of the chip is required at a fabrication facility of a semiconductor foundry which does not allow for the reconfiguration of the system postproduction. On the other hand, FPGA is bought off the shelf and requires a bitstream to configure the device but requires no physical layout. It can be easily reconfigured by designer. Neuromorphic computing can use these technology implementations alone or can be integrated using multiple technology integrations. For commercial chips, ASICs are more desirable but for simulating prototypes using a combination of ASIC and FPGA is preferable.

Generally, neuromorphic computing aims to improve its brain emulation and sets a basic criterion for this goal. It should be able to handle a large number of neurons and synapses and have the ability to mimic the spiking behavior of neurons with the aim to increase operability. It should be able to distinguish time differences smaller than 1kHz, like at the order of 10kHz [11]. Lastly, it aims to have a power efficient platform to eventually be able to use for more than one brain the world, perhaps for billions of them to reduce the load of current computer architectures.

1.2.1 Neuromorphic Computing Timeline [5]

1988	First analog silicon retina proposed by Carver Mead, initiating the emerging field of physically based computations inspired by neural networks. He pioneered the use of VLSI for the brain inspired neural network architecture called neuromorphic computing [6].
2006	First attempt at neuromorphic computing with Field Programmable Neural Array (similar to FPGA) and first neural network implemented in silicon.
2011	MIT implemented ANN on a chip using 400 transistors.
2012	Neuron designed with lower power consumption than older designs using 6 emerging devices called memristors by Purdue University. Emerging device called the neuristor developed using memristors to mimic biological neurons by HP labs.
2013	Human Brain Project started and anticipated brain emulation using VLSI for neuromorphic systems. Founded by the European Union as a ten-year project, it produced the neuromorphic chips BrainScaleS & SpiNNaker.
2014	IBM made TrueNorth, the closest design to resemble the human brain using 256 programmable silicon neurons. It successfully overcame the bottleneck limitations of Von-Neumann architectures consuming 70mW of power.
2017	Intel released Loihi chip, Intel’s 5th generation digital neuromorphic chip using 2 billion transistors and 14nm CMOS process.

1.3 Inspiration from Biology

To understand the behavior of biological neurons, they can be broken down into four parts with different functions: dendrites, soma, axon, and synapse, labeled in Figure 1.2. Dendrites are the parts of the neuron that receive data from other cells at synapse connections. The dendrite transmits the received information to the soma. The soma is considered the main processing unit of the neuron where the nonlinear processing happens. The soma membrane is charged when receiving input data until it exceeds a threshold which triggers an output spike, this is called the firing stage. When a threshold is not met, the membrane potential leaks out. When the output signal is triggered, the axon transmits it to the synapse which then transmits

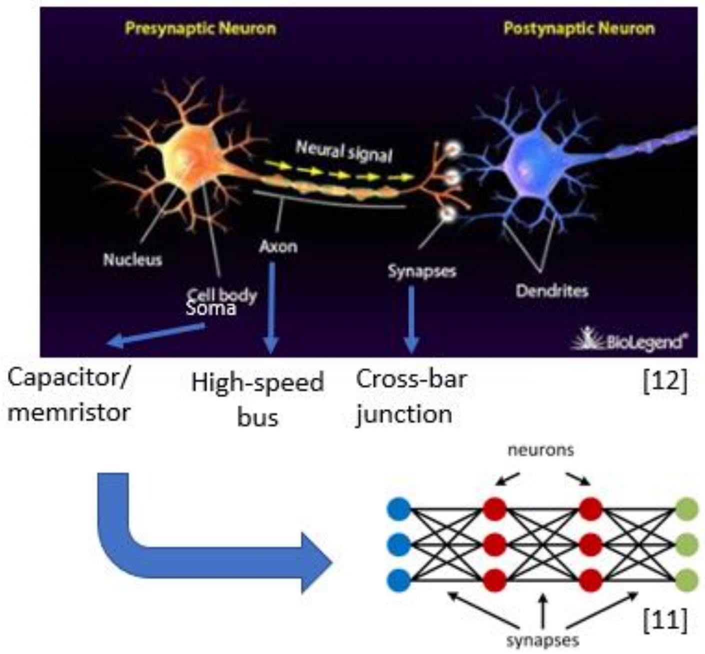


Figure 1.2 Biological neuron components and examples of hardware equivalents inspiring artificial neural networks for neuromorphic computing

the output to all connected neurons at that node [6]. In general terms, neurons process and generate a signal pattern. Neuromorphic computing aims to replace these biological components with ones made up of electrical circuits. As can be seen in Figure 1.2, some examples of electrical circuit replacements are using a capacitor or memristor as the soma, a hardware high-speed bus as the axon, and a hardware crossbar junction as the synapse to create an ANN [11].

In a biological neuron, signals get received, processed, and transmitted as a nerve impulse, also called an action potential or spike, pictured in Figure 1.3 [6]. During resting periods where no input signal is integrated, the output of the neuron will be null. When the

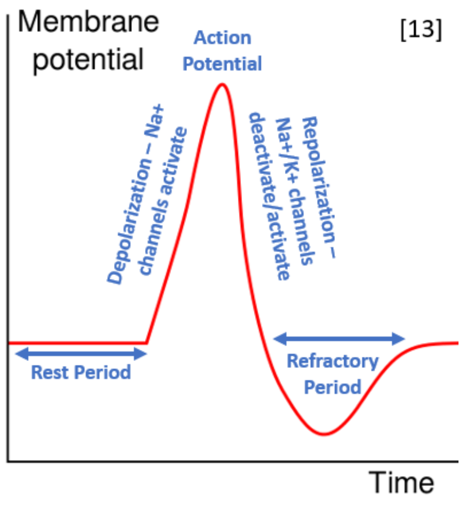


Figure 1.3 Biological neuron action potential

spike is triggered, depolarization will begin and the input ion channel Na+ will activate. The biological action potential will last a few hundred microseconds and repolarization will begin, the input ion channel Na^{+}will deactivate and the output ion channel K^{+} will activate. The output signal will then enter a refractory period where the neuron is not likely to fire again. SNNs have been used to encode raw analog input signals into voltage spike trains to mimic the communication between biological neurons. SNNs are considered the third generation of neural networks and are based on event-driven solutions while using low power consumption so they are preferable for encoding signals. Due to the smaller noise margin of analog circuits, encoding analog signals into voltage potentials becomes neuromorphic computing's biggest challenge [8].

Chapter 2: Spiking Neural Network & Neural Encoders

As mentioned before, neuromorphic computing requires conditioning of raw input data into spike trains for processing. Thus, using temporal encoders and decoders from SNNs is desirable since they use the event-driven time dependence of signals to map raw sensory data into spike trains [3]. This time-based encoding represents input data with voltage spikes and can offer perfect recovery for band-limited stimuli [4]. Temporal encoders can be broken down into rate and temporal encoding. An emerging technique is to combine more than one temporal encoding method to create a multiplexing encoder. Multiplexing encoding enhances accuracy of circuits and can produce similar power consumptions as regular temporal encoders. Accuracy is enhanced because multiplexing encoding conveys more information which makes voltage spikes less susceptible to noise [1]. Rate encoding has been extensively used in the past for its simplicity but it lacks the temporal aspect of input analog signals, further explained in Section 2.2. Temporal encoding has overpowered rate encoding in desirability because it uses the timing response of signals to map information and embeds the signal's time dependence into the output spike train. Multiplexing techniques haven't gained as much traction because it's still an emerging technique and has not been researched as much [1].

2.1 Why Neural Encoding?

In traditional computer architectures, ADCs and filters have been used for the preprocessing of raw analog signals; ADCs being the most commonly used. ADCs have dominated this area of signal processing due to its efficiency and high throughput in low data rates. For high data rates, ADCs need a lot more power to produce the same efficiency since noise is added during integration and included in the conversion of the analog signal into the digital domain. In order to mitigate power consumption in ADCs, a lower supply voltage can be used. ADCs are more limited than neural encoders using this power optimization technique because a low power supply for an ADC makes the threshold voltage of the transistor similar in amplitude to the supply voltage. This forces the input swing for ADCs to be rail to rail to switch the transistors [10]. On the other hand, neural encoders use analog circuits to map an input into voltage spikes and require less components than ADCs. The requirement of less components for neural encoding reduces both power and area consumption making neural encoders desirable for both high and low data rates. The fallback of using neural encoding is the use of analog circuits because it makes the design process more complex and susceptible to noise due to its multidimensional tradeoffs. On the other hand, ADCs are a simple mixed signal structure and only contain two analog components making the design process more straightforward. The encoding process of ADC require two steps: quantizing and encoding. Neural encoding encodes signals directly which speeds up the process. So, for both high and low data rates, neural encoding can provide a low area and low power design that is much faster than ADCs with a smaller noise margin.

2.2 Neural Encoding Overview

Under SNNs, most common encoders are temporal encoding and rate encoding. Temporal encoders have two subgroups that are most commonly used: latency (also called Time-to-Fist-Spike) and ISI encoding. Rate, latency, and ISI encoding can be seen in Figure 2.1. Rate encoding uses the number of spikes (firing frequency) during the encoding period to map the analog signal. The amplitude of the input signal will change the firing frequency. Although extensively used in the past, it has been replaced by temporal encoding because rate encoding lacks the consideration of the timing between spikes and becomes ambiguous in changing environments [4]. Temporal encoding uses the time between spikes to map input signals into voltage spikes. Latency encoding uses the distance between the clock signal that triggers a spike and the time of the first spike to be generated. In other words, it uses the latency of the generated spike to map the amplitude of analog signals. As can be seen in Figure 2.1, the latency of the generated spike changes based on different stimuli. ISI encoding uses the relative timing between the generated spikes within one encoding signal to map the input signal into voltage spikes. Figure 2.1 shows that for different stimuli, the distance between the three generated spikes vary. This encoder doesn't need the external reference of the clock signal for its encoding. Using the time differences of successive spikes makes temporal encoding more efficient in changing environments when compared to rate encoding since the latter can't explain the correlation between spikes. In comparison, latency encoding responds to the absolute time relative to the clock trigger while ISI responds to the relative time between

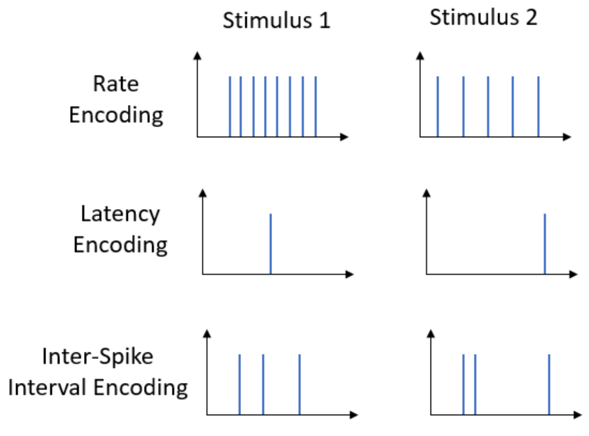


Figure 2.1 Neural encoding schemes (voltage vs time)

spikes. In this regard, ISI has the functionality of being able to rely on internal reference frames as consecutive spikes carry the information of the previous spikes. Thus, each spike acts as a reference for the next spike and allows ISI encoding to carry more information per sampling period autonomously [4]. ISI encoding requires more than one neuron to integrate at least two spikes per sampling period making the power consumption proportional to the number of neurons used. Latency encoding only requires one neuron and has a larger noise margin than ISI encoding because of the integration of less components for the encoding scheme at the expense of less information conveyed per sampling period.

Chapter 3: Spiking Neurons

There are two types of artificial neuron models: electrical input-output membrane voltage models and natural stimulus models. The first one predicts the output membrane voltage as a function of an electrical stimulus given an input current or voltage. It predicts the moment that the spike will occur at. The natural stimulus model represents the probability of a spike event happening and the stimulus is in the form of natural signals or chemical reactions. There are a lot of types of neuron models. At the most detailed model there is the Hodgkin-Huxley model which describes the membrane voltage as function of the input current and the activation of ion channels. This detailed model describes the relationship between the flow of ionic currents across the cell membrane and the corresponding membrane voltage of the cell. This model consists of a set of nonlinear differential equations that describe the behavior of ion channels and may include additional ionic currents with added modules like the inward currents Ca2+ & Na+ and the outward currents K^+ & leakage current, mimicking the action potential of biological neurons more closely. The end result of this model needs at least 20 parameters to accurately estimate or measure, so it becomes a very complex system of neurons [14]. The numerical integration of all the necessary equations becomes computationally expensive so mathematically simpler models are more desirable. The mathematically simpler models describe the membrane voltage as a function of input current and predict the timing of spikes without the description of the biophysical processes that create the biological action potentials. These models include IF and LIF neurons and have the possibility to add on circuit modules that would mimic biological components better despite not describing the biophysical current effects. These models contain less nonlinear equations since it uses some linear integration properties that simplify the neuron design and implementation. Lastly, there are abstract neuron models that only predict the output spikes as a function of the stimulus without describing the membrane voltage. The stimulus can be in the form of sensory input or induced pharmacologically. Due to the lack of the membrane voltage description these abstract models become less accurate and less desirable [4]. There are a lot of neuron models because there's different possible experimental settings and it's difficult to separate intrinsic properties of single neurons from measurement effects and the interaction between many cells in full neuron networks [14]. The mathematically simpler neuron models are the most used due to their usefulness in artificial neuron networks by mimicking biological neurons while maintaining a low area and power consumption.

3.1 Integrate-and-Fire & Leaky Integrate-and-Fire Neurons Overview

The IF and LIF neuron models are the most used in artificial neural networks. The IF neuron is the simplest with the basic behavior of firing a spike based on the input stimulus. The IF neuron circuit could be simplified to a capacitor being charged by an input analog signal until it reaches the threshold voltage of its output resistance triggering an output spike with the illusion of current flow. Thus, the charging of the capacitor is dependent on the capacitance and the neuron's threshold voltage and described using $I_m(t) \ = C_m(\frac{dV_m(t)}{dt})$ where $I_m(t)$ is the time-dependent current illusion of the membrane capacitor, Cm is the membrane capacitance, and $V_m(t)$ is the time-dependent membrane voltage of the neuron [14]. The increase of $V_m(t)$ is limited by the neuron's threshold voltage and will trigger a spike when the threshold is met. After the spike is generated, the capacitor is discharged. This model lacks the ability to mimic the diffusion of ions when an equilibrium is not reached so during resting periods, where a spike will not happen, noise will affect the membrane voltage and it will be retained until the next sampling period is triggered. This increases power consumption and reduces the noise margin. Thus, the IF neuron can produce voltage spikes from an analog input and mimic the refractory periods exhibited in biological neuron behavior but is susceptible to noise and misfiring during the refractory period. The LIF neuron adds on to the IF neuron model where it operates the same but has an additional leakage current module. This module allows any accumulated membrane voltage during resting periods to leak and mimic the refractory period of biological neurons more closely. This is because the LIF neuron model doesn't assume that the membrane is a perfect insulator and includes the membrane resistance in the time-dependent current illusion of the capacitor described by $I_m(t) = C_m\left(\frac{dV_m(t)}{dt}\right) + \frac{V_m(t)}{R_m}$ where R_m is the membrane resistance [14]. This increases the noise margin during resting periods and reduces power consumption significantly. Although IF neurons don't have the leakage module that the LIF neuron does, it can still produce the basic function of transforming analog signals into voltage spikes. When compared to the IF neuron, the LIF neuron is more complex which theoretically will result in a bigger size and power consumption but based on the four designed temporal encoders, the added functionality modules of LIF neurons allow for a smaller size and less power consumption when compared to the IF neuron. From the basic operation of these two neurons, it can be observed that the LIF neuron operation is more complex than the latency neuron. Designing the LIF neuron requires more complex circuit analysis and adjustments to have similar operations as the latency neuron but with enhanced properties.

Chapter 4: Experimental Results

Three main design parameters were considered for the temporal encoders and neurons designed. The first consideration was to have a small area while optimizing the circuit to achieve detectable spikes wider than 10 nanoseconds. A peak detector needs a spike width of at least 10ns for its voltage level to reach the threshold for detection before the spike disappears [1]. The low area was prioritized as the most important parameter by making the layout of the CMOS components within the layers of the capacitors and reducing the membrane capacitance as much as possible since it was used as the basis of the layout boundaries of the devices. This is risky for circuits that require precise capacitor values because the current flow through the devices will disrupt the electric field created by the capacitor layers and change the value of capacitance. But my circuit employed neurons that used capacitance to mimic a membrane and small capacitance changes from the electric field didn't affect it much. Since the area is dominated by the capacitor size, small devices are not necessary for the area reduction. Small devices have fast short-circuit currents which lead to faster switching delays reducing the width of the voltage spikes at the output. Since the IF neuron didn't contain a circuit module that increased its spike width, size modulation was used to increase the switching delays of some transistors in the IF neuron to achieve a minimum spike width of 10ns. The LIF neuron contained the spike width controller module, so it relied less on increasing switching delays. The spike width controller module uses a capacitor to delay the spike repolarization so in order to keep the area small, a minimum sized capacitor was used from the Global Foundries 180nm CMOS process. The minimum sized capacitor allowed the LIF neuron to rely less on switching delays to achieve a detectable spike but still didn't allow for the full minimization of switching delays which would significantly reduce power consumption. Bigger transistor widths for inverters helped to increase switching delays and to reduce the effects to the transistors of the electric field from the capacitor layers. After optimizing area for detectable spikes, the integration time it takes for the spike to be generated was increased as much as possible to produce a bigger range of integration times to theoretically increase accuracy; this property was not tested with a processor but a varying range of integration times was achieved. A low power consumption is considered after optimizing the previously mentioned parameters because increasing the integration time, switching delay, and transistor sizes increases the power consumption of the circuit as well. The techniques used to optimize power consumption without affecting the integration time and area too much was to reduce the current drawn from some devices by increasing the length and minimizing switching delays when the spike generation wasn't affected [1]. Analog circuit design has multi-dimensional trade-offs as mentioned in Section 1.1.2 so the best area, spike width, integration time, and power consumption will not be possible in a single design. Therefore, the focus of this design is on minimizing area while producing detectable spikes and maximizing integration time over power consumption.

4.1 IF Neuron Design & layout

The IF neuron consists of 11 CMOS devices and one passive capacitor. It requires three inputs: the input excitation current, a voltage source for the reference voltage (V_{ref}), and a clock signal. The circuit can be further simplified to basic CMOS circuits and function modules. Figure 4.1 shows the circuit schematic of the IF neuron and in the figure, basic CMOS circuits are enclosed in blue boxes and the function modules are enclosed in red boxes. With some

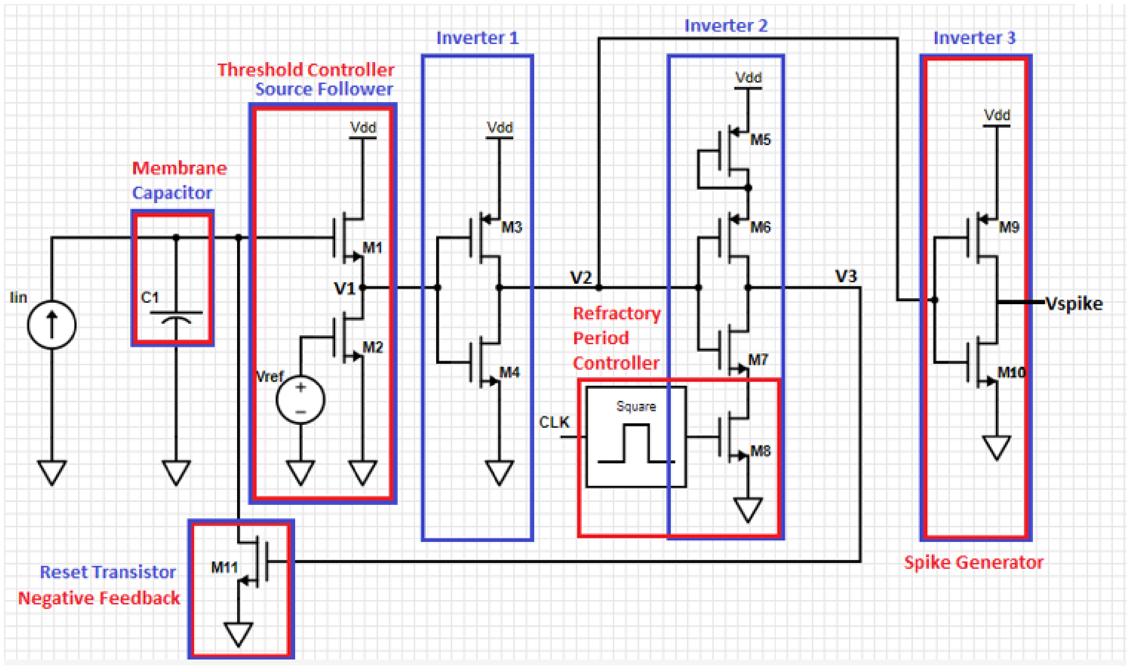


Figure 4.1 IF neuron circuit schematic

circuit components, both blue and red boxes are used to portray their functionality. Starting from the left of Figure 4.1, the first basic CMOS component is the membrane capacitor which serves as the function module of the circuit that mimics a neurons membrane by charging the membrane capacitor through an input current until the neuron's threshold voltage is met. This sets the maximum possible voltage that the capacitor will charge to, which then triggers the output voltage spike generation and simultaneously the discharge of the capacitor until the next sampling period. The basic CMOS circuit next to the capacitor is the source follower with an active load. This source follower increases the linear integration range of the membrane capacitor by the input current. The voltage bias of the source follower's active load creates the threshold controller function module where the neuron's threshold voltage is controlled by the amplitude of the bias voltage. Following the source follower is Inverter 1 and subsequently Inverter 2 and Inverter 3 in parallel. Inverter 1 is a basic CMOS inverter serving to trigger the generation of a spike and simultaneously discharge the membrane voltage. Inverter 2 has two added CMOS transistors where one is a diode connected PMOS M5 serving as a voltage drop to lower power consumption and the other is a load NMOS M8 whose voltage bias is a clock signal that creates the refractory period controller by adjusting the width of the clock pulse and the period of no CLK signal. The period between clock pulses is the sampling period where the spike will happen, and the clock pulse is the sampling period trigger at which the capacitor will start to charge. The amplitude of the clock signal may have effects on the circuit operations but in this case a rail-to-rail clock was used to impose a vdd voltage at each pulse. M8 sets the slew rate since it's responsible for the speed of the feedback discharge. Inverter 3 serves as a fast-switching digital inverter to produce the output voltage spikes when Inverter 1 triggers a switch. Inverter 3 becomes the spike generator, and the width and length of these transistors

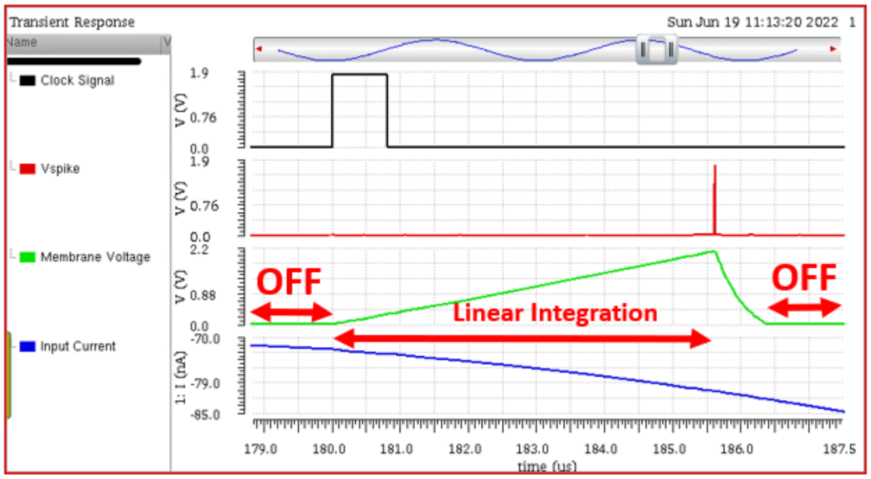


Figure 4.2 Transient simulation of IF neuron for one sampling period. In black, input clock signal triggers sampling period. In red, output Vspike is the generated voltage spike. In green, the membrane voltage charging/discharging. In blue, input analog current.

affected the spike width of the output spikes more significantly. The last circuit component in the IF neuron is the minimum sized reset transistor that provides negative feedback to the membrane capacitor during spike generation to discharge the accumulated membrane voltage to get the circuit ready for the next sampling period. This negative feedback module is what starts the refractory period by discharging the membrane capacitor.

IF Threshold Modulation Overview

When a sampling period is triggered by the clock signal, black in Figure 4.2, the membrane voltage V_{mem} begins to increase through the linear integration of the membrane capacitor by the input current. As can be seen by the green signal in Figure 4.2, the linear integration happens when the clock signal is triggered after an off (refractory) period in the circuit operation. As V_{mem} increases towards the neuron's threshold voltage, V_1 also increases and starts to reach the threshold voltage of M4. When this threshold voltage is met, V_2 to start to decrease as previously V_{in} was too low to signal an inverter switch. As the membrane capacitor charges, the source-follower produces a signal described by $V_1 = k(V_{mem} - V_{ref})$, where Vref is the reference voltage used to bias the active load of the source-follower and k is the slope coefficient of the signal [15]. So, the neuron's threshold voltage can be adjusted by changing the amplitude of V_{ref} and the source follower becomes the neuron's threshold controller. The neuron's threshold voltage will linearly increase as V_{ref} increases. This property models the long-term adaptation of cortical cells that allow neurons to change shape and thus, electrical properties.

IF Power Consumption Overview

The most power consuming parts of the IF neuron circuit were Inverter 1 and 3. This comes from the short-circuit currents through the inverters at switching times, especially because inverter delays were not minimized to produce output spikes wider than 10 nanoseconds. The short-circuit currents of Inverter 2 consume less power than the other two inverters because of the diode connected PMOS M5 acting as a small resistor. When there is no input current or a sampling period hasn't been triggered by the clock signal, the neuron is in an off state, as can be seen in Figure 4.2. The off period comes after the reset transistor drains V_{mem} post spike-generation to prepare for the next sampling period. Ideally, V_{mem} will remain zero during this time making the power consumption null. But power simulations showed that when integrated into more complex circuits, noise affects the off period of this neuron and will increase the power consumption.

During the on state of the circuit where the linear integration and spike generation happens, the main multi-dimensional trade-offs affecting power consumption for the IF neuron are the input current offset, the firing rate, and the integration time set by V_{ref}. While keeping V_{ref} and the vdd the same, increasing input current increases the current flowing through Vdd as the neuron's fire rate increases by increasing the switching speed of the transistors which also decreases the integration time. Since the integration time was prioritized over power consumption, the reduction of the input current to save power was limited. The sizes of the transistors using Vdd were optimized by reducing their drain current to reduce the power consumption, further explained in Section 4.1: IF Circuit Sizing. The power dissipation also depends on V_{ref} because increasing V_{ref} increases integration time and this in turn increases power consumption since current is drawn for longer. This is a problem because the IF neuron needed to not minimize the inverter delays fully which led to an even bigger power consumption increase. But shorter integration times increases a neuron's firing rate which increases power consumption as well. In typical applications, the refractory period is used to limit the neuron's maximum firing rate. The firing rate of the designed neurons was high because they are made up of very small transistors, but the clock signal is made small enough for the voltage spike to occur outside of it. This triggers the reset transistor to drain V_{mem} and limits the spike generation to only one spike per neuron, reducing power consumption [15]. Smaller V_{ref} values can lead to a smaller power consumption as well because it reduces integration time but for very small V_{ref} values, power consumption increases again because the source follower acts as a low-pass filter on the falling edge of the spike and the switching time of inverter 1 is increased, increasing power consumption.

4.1.1 IF Circuit Sizing

Since area consumption was the number one priority for this design, the membrane capacitance became the leading circuit parameter since it needed to be small to reduce the chip area. By reducing the capacitor size, other parameters that increase power consumption had to be increased as well like V_{ref} and input current offset to keep a consistent operation. The input parameters of the circuit will be discussed during the implementation of the temporal encoders in Section 4.4 and 4.5. Components were minimum sized when possible but to optimize the circuit the size of some components had to be increased. To make sure the area would remain small during the size modulation process, the size increases were limited to a

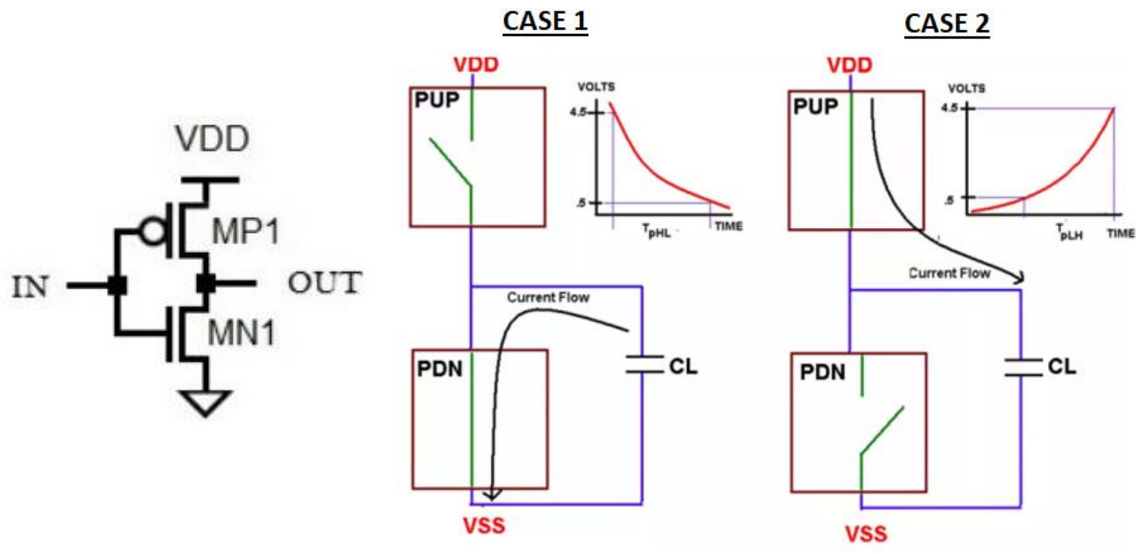


Figure 4.3 CMOS inverter on left. On right, Case 1 represents a pull down (PDN) network, Case 2 represents a pull up (PUP) network. [16]

small number of transistors so all CMOS components would fit within the capacitor layers during layout. To start the designing of the IF neuron, the basic CMOS circuits were sized individually and continuously adjusted based on simulation results of the neuron, starting with the three inverters.

Basic CMOS inverters are the main structure of this IF neuron and have the operability of inverting an input signal and its operation can be explained using the corresponding pull-up and pull-down current network that can be seen in Figure 4.3. On the left, the basic CMOS inverter schematic is made up of a PMOS and NMOS inverter with connected gates and drains. This circuit connection produces a simple circuit of operation of two cases. In case 1, IN provides a high signal and triggers the pull-down network (PDN) with a high-to-low time delay called t_{pHL}. In this case, MP1 turns off and the output voltage OUT discharges through MN1 to zero, as can be seen in Case 1 of Figure 4.3. The load capacitance, CL of this circuit can include intrinsic capacitances and external loads. As can be seen in the chart at the top right corner of case 1 in Figure 4.3, t_{pHL} is the time it takes for the current to fully discharge through MN1 and can be described by $t_{pHL} = Ron_n * C_L$ where $\mathrm{Ron}_n$is the on resistance of MN1. In case 2, IN provides a low signal and triggers the pull-up network (PUP) with a low-to-high time delay called t_{pLH}. In this case, MN1 turns off, MN1 turns on, and the active load capacitance gains charge from Vdd since the path between MP1 and OUT has low resistance, as seen in case 2 of Figure 4.3 [16]. As can be seen in the chart at the top right corner of case 2 in Figure 4.3, t_{pHL} is the time it takes for the current to fully charge to Vdd through MP1 and can be describes by $t_{pLH} = Ron_p * C_L$ where Ron_p is the on resistance of MP1. To minimize t_{pHL} and t_{pLH}, the lower mobility of holes in PMOS transistors must be compensated by making its width needs to be at least 1.5-3 times bigger than that of the NMOS transistor. Generally, static CMOS logic style circuits are considered robust against technology scaling and can perform reliably at low voltages and arbitrary transistor sizes since the parallelization of the PUP & PDN networks creates a ratioless logic where there is no dependence between the output voltage and transistor sizes [16]. But minimizing the delay through circuit sizing is still important because longer delays consume more power, and the inverter will be limited to the speed of the slowest delay.

The IF neuron has the limitation of needing delays longer than minimum because it needs to provide output spikes with a minimum width of 10 nanoseconds for them to be detectable. Since I aimed for small devices, the transistors have very small internal capacitances which reduces the switching delays t_{pHL} and t_{pLH}. To guide the start of the transistor sizing of the three inverters, I minimized the switching delays by making the PMOS width bigger and leaving the length at a minimum. Since inverter 2 contains two extra transistors, I started with minimum sized transistor and adjusted based on simulation results. I used the delay equations to guide the size adjustments of the inverters during simulations to optimize the circuit. Table 4.1 shows the final transistor sizes of the IF neuron.

Inverter 1 is made up of PMOS M3 and NMOS M4 and inverter 3 is made up of PMOS M9 and NMOS M10. I noticed these inverters required the most power consumption and subsequently the biggest area. Inverters 1 and 3 are supposed to act as simple and ideally fast logic switches, but the small component sizes predict a spike width less than 10ns, so the switching delays are increased through size adjustments. The switching delay equations show that it depends on the on resistance (Ron) and load capacitance of the inverter. Ron can be described by $R_{on} = \frac{1}{g_{ds}} = k' \left(\frac{W}{L}\right) V_{ov}$ which shows its proportionality to the ratio of the width and length, also called the aspect ratio, of the transistors where k' is the product of $\mu * C_{ox}$ of either PMOS or NMOS based on which pull up or pull down network is active. So, in order to increase the delay of the transistors, Ron can be increased. To increase Ron, W can be increased, or L can be decreased. Another way to increase the delay of the transistor switching is to increase the load capacitance. The load capacitance will be dominated by Cgs since it is the highest internal capacitance for each transistor. Cgs can be described by $C_{gs} = \left(\frac{2}{3}\right) C_{ox}WL$ and can be increased by increasing W and L.

So, to increase both Ron and C_{load}, W can be increased but the effects of modulating L are contradictory for these two variables and this effect becomes less reliable. To increase the delay of inverter 1 and 3, the W of PMOS M3 and M9 were increased. Inverter 3 had negative effects from channel length modulation, so inverter 3's M9-M10 L was kept at a minimum. For inverter 3, it was important to match the PMOS and NMOS transistors' lengths. The width of NMOS M10 was reduced to minimum to draw more current through inverter 3 as output loses were noticed in simulations (voltage values less than vdd). In inverter 1, the length of PMOS M3 was used for power optimization and will be discussed in Section 4.1.1 **IF Power Saving Techniques**. Since the first prioritized design parameter was optimized by reducing area while increasing switching delays to provide detectable spikes, inverter 1's NMOS M4 is used to increase the integration time of the neuron based on its operability. A wide range of integration time was the second prioritized design parameter to provide varying integration times for small current amplitudes. Inverter 1 increased the integration time of the neuron by increasing the width and length of M4 since this changed the threshold voltage that V1 had to reach to trigger M3 to start drawing current and switch the inverter, as mentioned in Section 4.1.

Inverter 2 didn't require as much current drawn from the source due to its diode-connected minimum-sized PMOS M5. The focus of inverter 2 design is in increasing the delay of the load NMOS M8 since it sets the speed of the feedback discharge to minimize inverter area. Thus, the NMOS M7 is minimum sized and the PMOS M6 increases the width to increase the inverter delay. For M5-M7, it's important to match the lengths of these devices to reduce mismatch errors. The load NMOS M8 of inverter 2 can lower the speed of the discharge by increasing its width and decreasing its length but it required a minimum length increase to fully discharge the capacitor. MN8 sets the refractory period and for longer periods, L must increase to get neuron functionality for the longest integration time at the minimum input current amplitude. The reset transistor, M11, that provides the negative feedback is minimum sized. The source follower is the third most power consuming component and the current required resulted in an increase in the width of NMOS M1. The length of the NMOS M1 was used for power optimization and will be discussed in Section 4.1.1 **IF Power Saving Techniques**. The size of the active load, M2, in the source follower was kept at a minimum but it's effects on power will be discussed in Section 4.1.1 **IF Power Saving Techniques**. It should be noted that the size modulations of the source follower affected spike width and spike frequency. It could be used to optimize these parameters.

IF Power Saving Techniques

The first and second design parameters prioritized for optimization were considered in the previous section. The third prioritized design optimization parameter was power

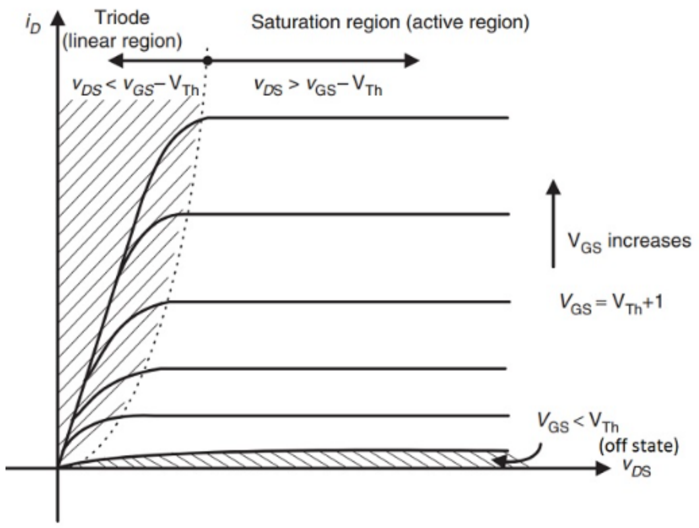


Figure 4.4 Operation regions of CMOS transistors [23]

Table 4.1. IF neuron CMOS sizes

IF Neuron Sizes	CMOS	W (m)	L (m)	Fingers	gm/Id (V^{-1})
M1	NMOS	1u	950n	-	24.1
M2	NMOS	460n	180n	-	2.6
M3	PMOS	3u	1.5u	2	1.7
M4	NMOS	1u	600n	-	6.67
M5	PMOS	460n	180n	-	31.9
M6	PMOS	1u	180n	-	31.6
M7	NMOS	460n	180n	-	1.16
M8	NMOS	7u	600n	3	28.7
M9	PMOS	5u	180n	2	29.6
M10	NMOS	300n	180n	-	0.6
M11	NMOS	460n	180n	-	24.1

consumption and will be optimized without affecting the previously optimized design parameters. The optimization strategy I used to reduce power consumption is to use the drain current equation of the transistors to guide size modulations. Almost half of the circuit components operate under subthreshold techniques, and this helps to reduce the overall power consumption as well.

CMOS transistors can operate under the linear triode region and the active saturation region. Transistors operate in the triode region when $V_{ds} < V_{gs} - V_T$, where V_{ds} is the drain-to-source voltage, V_{gs} is the gate-to-source voltage, and VT is the transistor's thermal voltage. To simplify, triode region operation occurs when $V_{gd} > V_T$ and saturation region occurs when $V_{gd} < V_T$, as can be seen in Figure 4.4. Based on the CMOS drain current equations for both the triode and saturation regions, the drain current is proportional to the aspect ratio in both. So, for any CMOS device that is on, where $V_{gs} > V_T$, the drain current will decrease by decreasing the width and increasing the length. The width of most transistors was set to optimize the switching delays but the width of the NMOS M10 was reduced to the minimum because channel length modulation for inverter 3 did not work well. Inverters 1 and 3 were the most power consuming circuit components. For inverter 3, there weren't many techniques that could be used to decrease power consumption except for the width reduction of the NMOS M10. The length of the PMOS M3 in inverter 1 was increased to reduce power because this decreased the firing rate of the neuron. The only other possible power optimization technique for circuit sizing was increasing the length of the source follower M1 to be as close to its width as possible.

When transistors operate under the triode region, this is called subthreshold operation. From Table 4.1, it shows that five out of the 11 CMOS devices operate under subthreshold techniques based on the gm/id parameter based on the ratio of the transconductance gm and the quiescent current id. This parameter was gathered from the DC operating point simulation of each transistor. Looking at Figure 4.5, we can see that subthreshold techniques fall under weak inversion where $V_{gs} < V_T$ and overlaps in the moderate inversion region where the strong-inversion equation creates a threshold barrier between the triode and saturation regions. The moderate inversion region also overlaps the active saturation region on the right and continues on to strong inversion once steady state is reached in the saturation region.

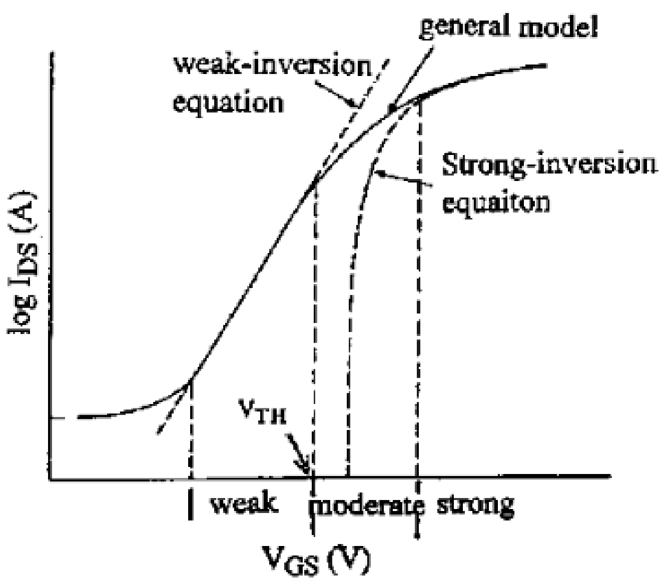
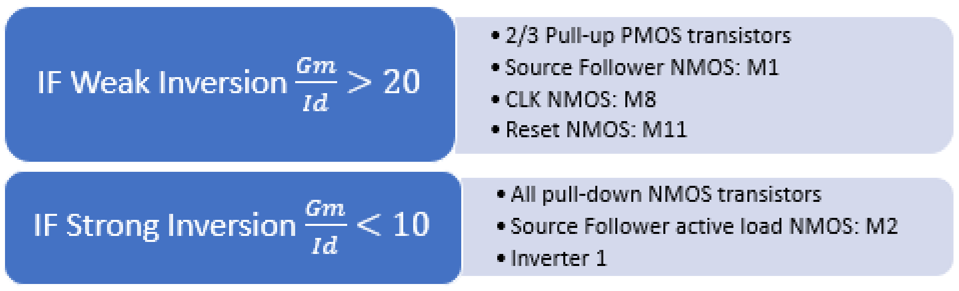


Figure 4.5 Inversion regions of CMOS operation [24]

Inversion regions are looked at for a specific V_{ds}. Using the gm/id parameter, inversion regions can be analyzed because the operation will fall under weak inversion approximately when $\frac{g_m}{i_d} > 20$ and under strong inversion when $\frac{g_m}{i_d} < 10$. Under weak inversion, the gain bandwidth of the circuit is reduced when compared to strong inversion. But due to the ability of small transistors to switch fast, a high sampling frequency can still be achieved under weak inversion.

From the IF neuron's inversion region summary in Figure 4.6, we can notice that most pull-up network transistors fall in weak inversion whereas all pull-down network transistors fall in strong inversion. M3 was the only pull-up transistor that fell in strong inversion and the only transistor that increased its length to reduce power consumption. Inverter 1 falls fully under strong inversion which makes it understandable for it to be the second most power consuming circuit component. Subthreshold operation has the benefit of a much lower power consumption with comparable reliability to operation under the saturation region at the expense of stability. Making the $\frac{g_m}{i_d}$ ratio bigger can be done by increasing the width of transistors; which is why most pull-up transistors fall under weak inversion. The stability issue



IF Weak Inversion $\frac{Gm}{Id} > 20$	• 2/3 Pull-up PMOS transistors • Source Follower NMOS: M1 • CLK NMOS: M8 • Reset NMOS: M11
IF Strong Inversion $\frac{Gm}{Id} < 10$	• All pull-down NMOS transistors • Source Follower active load NMOS: M2 • Inverter 1

Figure 4.6 IF neuron inversion region summary based on $\frac{Gm}{Id}$

with big $\frac{gm}{id}$ ratios is that in weak inversion, the neuron's thermal voltage VT is not well controlled and noise susceptibility is more noticeable. This is because at such small transistor sizes, a lower stability causes an increase in irregularities within the fabrication materials due to imperfections. This disadvantage can be overcome by using bigger transistor sizes to allow for a bigger noise margin. Because the layout of this device is done so that the transistors will fall within the capacitor layers, bigger transistor sizes won't increase the area by much. The active load of the source follower did not use subthreshold techniques since $V_{ref} > Vth_N$ due to its small size. This means that for M2, $V_{gs} > Vth_N$, where Vth_N is the NMOS M2 transistor's threshold voltage and the operation can't fall within the subthreshold region. Inversion regions were looked at after the sizing of the circuit was optimized but further improvement can be achieved by increasing the use of subthreshold techniques to more components.

The first prioritized design parameter was a small area, so the circuit sizing was led by increasing the delay of the transistors, so a smaller membrane capacitor was able to be used while still maintaining a varying integration range for a sinusoid input current. Since the layout of the CMOS transistors are made within the capacitor layers, increasing the size of the transistors to increase the switching delay did not affect the overall neuron area. The overall switching delay of this neuron was distributed amongst the three inverters and the source follower by increasing their widths based on circuit simulations. The integration time was increased as well, and lastly power consumption was optimized without affecting the first prioritized design parameters by only adjusting a couple circuit components and the input parameters during encoder integration. Cadence Virtuoso was used for the circuit simulations and layout of the design and a transient analysis was used to predict the time-dependence of spikes. To optimize the circuit performance, the voltage at each node and the current drawn from vdd were the most important signals to analyze. By analyzing each voltage node, the circuit weaknesses were evident and size adjustments were made to overcome these weaknesses. An example of this process is having semi-working voltage spikes but not for the entire input current range, so voltage nodes are analyzed to see which part of the circuit requires more current. The circuit simulations pointed out that circuit matching issues lead to varying outputs. When the mismatch is caused by V_T, it is desirable to have devices that are longer than minimum to reduce material irregularity issues, especially when working under subthreshold techniques. Longer than minimum transistors are also helpful when working within the capacitor layers due to its electric field effects. Bigger transistors are less susceptible to electric field effects, and it was noticeable that inverter 3 required the biggest width and aspect ratio to conduct the circuit operation effectively. Overall, this neuron possesses the ability to encode an analog input signal into voltage spikes and will be implemented into temporal encoders in Section 4.4 and 4.5.

4.1.2 IF Neuron Layout

The proposed IF neuron design is designed using the Global Foundries 180nm standard CMOS process and the Cadence Virtuoso layout. Figure 4.7 shows the final layout of the transistor sizes within the blue Place and Rout (PR) boundary. The capacitor layers are separated next to the final circuit layout for visual simplicity but in the final circuit layout they fall within the PR boundary enclosing most of the CMOS transistors. This layout achieved a final area of 14.9um by 12.5um for a total of 186.25um^2 with 12 circuit components. To start the layout of the neuron, it is clearly evident by Figure 4.7 that the membrane capacitor was used to guide the PR boundary area plus a few micrometers. Following this step, the supply voltage vdd and ground (gnd) nodes were placed at the top and bottom of the circuit to reduce path lengths and reduce parasitic path resistances. PMOS transistors were placed at the top of the layout close to the vdd node and NMOS transistors were placed at the bottom of the layout close the gnd node unless otherwise inconvenient. This proved to be effective and resulted in a small power consumption drop of 60nW at post-layout simulations. Apart from this path optimization, no other paths were optimized but there is an opportunity to reduce power by studying the effects of the capacitor's electric field on the circuit components and optimize the paths further based on these effects.

Table 4.1 shows that the use of fingers was employed for transistors with big widths to reduce the area of the transistors. On Figure 4.7, it can be seen that there is a lot of empty space within the capacitor layers that is not used by any transistor which makes it possible to improve the design by increasing transistor sizes, if necessary, at the expense of power consumption. Bigger transistor sizes can reduce the membrane capacitance while keeping similar functionality. Adding fingers can be helpful when components won't fit in a desired spot and a change in the component's shape is necessary. Adding fingers means adding the contacts per diffusion in a CMOS transistor and is referred to with the letter N. When fingers are added the shape of the transistor changes by $\frac{W}{N}$ and reshapes it to taller components. Usually, analog designs prefer to keep N=2 because adding fingers changes the load capacitance and can often mess with circuit operations when N>2.

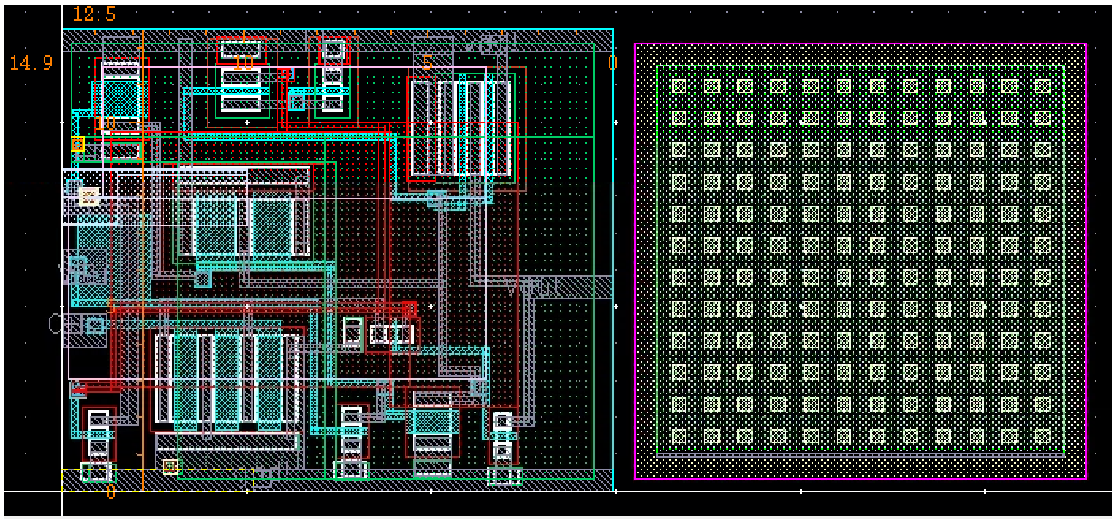


Figure 4.7 IF neuron layout

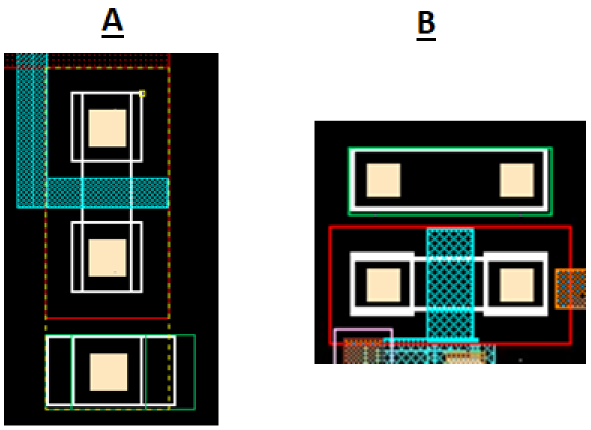


Figure 4.8 Layout of PMOS transistors A and B

To start my layout, the circuit sizes had to be optimized first. During circuit sizing, the capacitance was reduced as much as possible and the area of the capacitor was checked continuously to ensure a small overall area. When circuit sizing was finished the circuit components were generated into the layout. The PR boundary was set based on the capacitor area and added 1.5um to its width to compensate for the design rule of the metal via needed to connect to the capacitors outer layer called TOP metal. The capacitor layers consist of TOP metal for the input node and metal 5 for the gnd node. The via needed to connect the input node to the capacitor requires a path between metal 1 to the TOP metal which means it has to pass through metal 1-5 and then to the TOP metal. Since I aimed at a small area, to reduce area, I strictly followed design rules. The metal via had a metal 5 width of 0.3um and the design rule limitation needed a distance of 1.2um between the metal 5 via layer and the capacitor's metal 5 layer. This is what lead to the PR boundary area selection. The following step is to add detached substrates to all transistors. The more substrates possible the better but adding substrates for small lengths can be tricky. When the substrate is added to the shortest side, it is generated with a smaller n+/p+ and active area than the minimum specified in the design rules. Figure 4.8 shows an example of this challenge, on the left of the picture is the IF PMOS M10 (transistor A) and on the right if the LIF PMOS M10 (transistor B). Both of these transistors have similar widths and lengths, but transistor A resulted in a design rule error because the substrate didn't meet the minimum area for n+ and active boundaries. The n+ boundary can be seen as the green boxes located at the bottom of transistor A and on top of transistor B. The white boxes are the active boundaries, and the red are the p+ boundaries. There are two possible fixes to this situation. The first fix is utilized for transistor A where a new p+ and active boundaries are drawn to meet the area requirements. This is a tricky fix because there are a lot of design rules that must be met. Transistor A shows the best possible way to draw these layer boundaries. The second fix is utilized for transistor B where the substrate is generated along the width of the transistor. This latter solution is the simplest but requires more area consumption which led to difficulties during the layout of the LIF neuron, further discussed in Section 4.2.

After components were generated and substrates were added to them, the layout placement of the components can begin. This requires a game of Tetris with the circuit components while trying to optimize the vdd and gnd paths by keeping PMOS transistors at the top of the circuit and NMOS transistors at the bottom. It is useful to begin with the TOP metal via needed for the capacitors input node or to leave a space for it to be added later. The number of fingers used for transistors was adjusted at this step and Table 1 shows that all three inverters employed one use of fingers. It was evident that finger sizes bigger than 2-3 were not necessary and required more work than they were worth. For this step of Tetris, it was very important to keep in mind the design rules that required minimum spacing between transistors. The spacing between transistors was dominated by the n+ and p+ design rules for NMOS and PMOS transistors accordingly. The PMOS transistors required the most spacing between each other but for this layout it was not much of a problem since there was plenty of empty space within the capacitor layers. The design rules that were important to keep in mind during component Tetris were: spacing between n+ layers needs to be 0.4um, spacing between n+ and p+ layers needs to be 0.6um, and spacing between p+ layers needs to be 1.4um. While playing Tetris with the transistors keeping in mind design rules, it is useful to run the Design Rule Check (DRC) layout simulation to adjust component placement based on overlooked design rules. Important layout design rules to keep in mind to simplify the layout process are in Table 4.2. Once all transistors and the TOP metal via were placed within the PR boundary and the DRC simulation resulted in no errors, the paths to vdd and gnd were created while continuously running the DRC simulation to adjust paths breaking design rules. These paths were created using metal 1. Metal 1 was used for the rest of paths that didn’t overlap the vdd and gnd paths. Once metal 1 can’t be used without overlapping itself on other paths, other metals can be used with the employment of vias. Vias can be used within the capacitor layers for path connections as long as the vias don’t pass through the capacitor layers. Metal 1-3 were used for the circuit paths of this neuron, metal 5 was used for the capacitor gnd, and TOP metal was used for the capacitor input node. The next step is to run the Layout Versus Schematic simulation and fix any inconsistencies between the schematic and layout. In this step, I found most of my errors came from either inconsistent node labels or most often forgotten connections of some transistor path. To avoid the first, avoid changing node names once the components are generated in a layout. These were the hardest errors to fix because of the confusing error messages in the LVS results. It was useful to visualize the errors by redirecting the layout and schematic view to the mentioned coordinates in the error summaries and see which parts of the circuit were the problem. Most often, one fix to an error lead to many other errors being fixed on this step. The last step is to run the Parasitic Extraction (PEX) simulation that gathers a netlist from the layout that includes the parasitic resistances and capacitances and correlates them to nets on the schematic. This allows for the post-layout simulations of the circuit with included parasitic components.

Table 4.2 Important layout design rules to keep in mind during layout process

From	To	Minimum distance
p+	p+	1.4um
p+	n+	0.6um
n+	n+	0.4um
Metal 5	Metal 5	1.2um
TOP metal	TOP metal	1.5um
Poly-layer	Active layer	0.1um

When using Cadence virtuoso for the layout of a circuit, especially when running it through a Linux computer, it is important to keep in mind that it is prone to errors and should be reset when display issues arise to avoid design complications. Some reasons to reset Cadence can be when layers are not able to be selected or display issues are evident like "invisible" circuit components, vias, or PR boundary. In these scenarios it is best to close everything and reset Cadence before making new changes. The last weird error I encountered that is worth mentioning is that sometimes errors arise when connecting NMOS substrates to gnd. When incorrect path connections are made, for example two paths using the same metal overlapping each other, a colored X (in my case yellow) appeared on the incorrect overlap. The weird error occurred when I connected some NMOS substrates to gnd and the yellow X appeared. Based on circuit knowledge I knew my connection was right; NMOS substrates go to gnd and PMOS substrates go to vdd. This error wasn't common but happened at least twice. The error went away by taking away the substrate by applying "none" substrates, saving the modified layout, and then adding the substrate again.

4.2 LIF Neuron Design and Layout

The IF neuron consists of 16 CMOS devices and two passive capacitors. It requires four? inputs: the first three inputs are the same as the IF neuron with the extension of a voltage source bias for the leakage current module. The circuit can be further simplified to the same basic CMOS circuits and function modules as the IF neuron but includes additional ones. Figure 4.9 shows the circuit schematic of the LIF neuron and same color coding as the IF circuit schematic is used for this figure. Starting from the left we can see that it contains the same membrane capacitor as the IF neuron and possess the same operability for this module with the added functionality of being controlled by feedback current (I_{fb}). This changes the linear integration of the membrane capacitor to an exponential form briefly as V_{mem} approaches the neuron's threshold voltage. Moving on to the next circuit component, its noticeable that it also contains the same type of source follower for the linear integration of the membrane capacitor by the input current. This module has the same functionality as the IF neuron but the two additional connections at the input of the source follower increase its operability. Thus, a

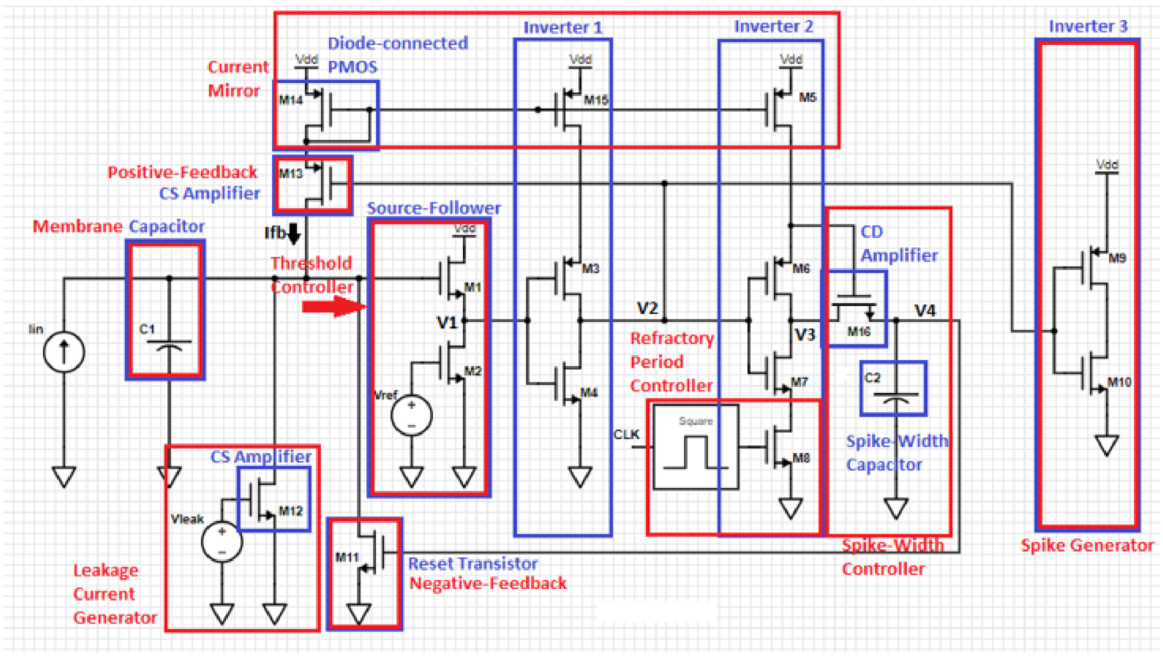


Figure 4.9 LIF neuron circuit schematic

different approach was used during circuit sizing than the one for the IF neuron's source follower. Next to the source follower, its noticeable that the circuit is made up of mainly three CMOS inverters similar to the IF neuron. The only inverter that is the same as the IF neuron is inverter 3 due to the use a current mirror to provide power to inverter 1 and 2 in the LIF neuron circuit but it does have one additional node connection at its input. Inverter 1-3 in the LIF neuron have the same functional abilities described in the IF circuit overview but contain additional components that add on to their functionality. The PMOS transistors of inverters 1 and 2 are in series with a current mirror generated by the diode connected PMOS M14. The use of this current mirror led to a lower power consumption when compared to the IF neuron despite having more components in the LIF neuron because it reduces the switching delays of the neuron. Inverter 2 also contains the additional function block of the spike width controller where a common-drain (CD) amplifier is used to trigger the integration of the spike-width capacitor until it charges enough to switch the minimum-sized reset transistor M11. Although inverter 3 is the same for both IF and LIF neurons, the input node has two connections instead of just one. The IF neuron connected the input node of inverter 3 solely to node V2 but the LIF neuron adds the additional connection of the gate of the CS amplifier M13 to node V2 to trigger a positive feedback increase as V_{mem} reaches the neuron's threshold voltage exponentially increasing V_{mem}, discussed in Section 4.2 **LIF Threshold Modulation & Positive Feedback Overview**. The last additional function module that the LIF neuron has is the leakage current generator that uses a CS amplifier with a constant voltage bias to discharge any accumulated membrane potential during the neuron's off periods. When the input current is at rest or a spike has been generated within one sampling period and the membrane voltage has been discharged, this is considered an off period. During this time, the membrane potential can

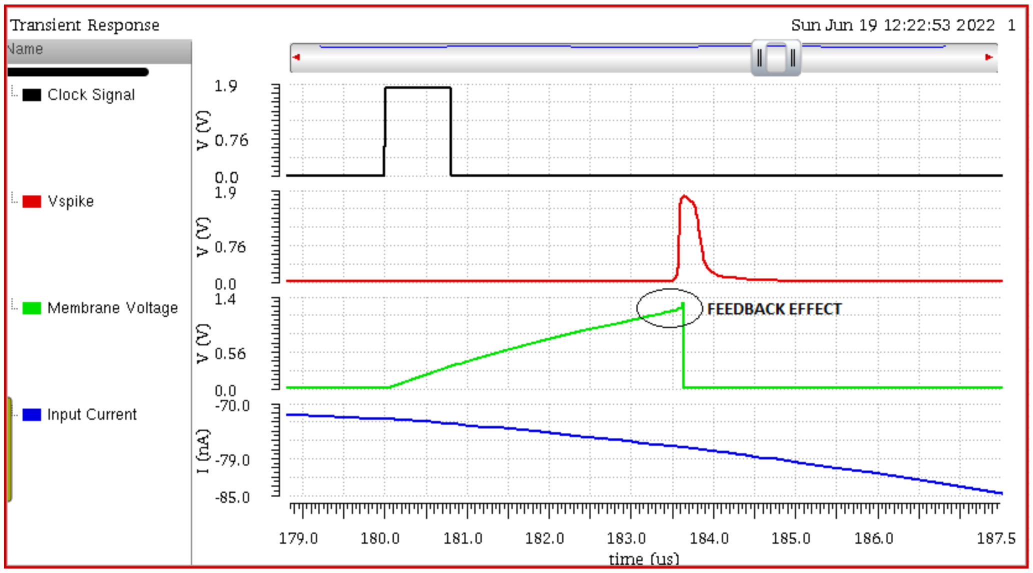


Figure 4.10 Transient simulation of LIF neuron for one sampling period. In black, input clock signal triggers sampling period. In red, output Vspike is the generated voltage spike. In green, the membrane voltage charging/discharging. In blue, input analog current.

accumulate voltage through noise and the leakage current module discharges any voltage accumulation and rests the membrane potential.

LIF Threshold Modulation & Positive Feedback Overview

The LIF neuron linearly integrates the input current when charging C_{mem} the same as the IF neuron, mentioned in Section 4.1, but as Vin approaches the threshold voltage of M4, the current through M15* starts to rise more rapidly as can be seen on the top right graph of Case 2 where the red voltage signal begins to change into a steeper slope at approximately $\frac{vdd}{2}$. At this point, the positive feedback current starts to increase V_{mem}, and consequently V_1, exponentially faster, as can be seen circled in Figure 4.10. Once the M4 threshold voltage is reached by V_1, inverter 1 changes into a pull-down network and V_2 decreases. Because I kept the diode-connected transistor M14 small to reduce device area, the positive feedback is not strong in my design because I_{fb} is not strong as an effect; a small power consumption is achieved regardless. The feedback could be stronger by employing a bigger M14 transistor to increase the feedback current. This would further reduce power consumption further since it would decrease the area under the integration curve by promoting an exponentially faster charge as V_{mem} approaches the neuron's threshold voltage. The positive feedback has the effect of making inverters 1 and 2 switch very rapidly, which reduces their power consumption significantly. The threshold modulation of the LIF neuron has the same ability as the IF neuron to change the neuron's threshold voltage by changing V_{ref} [15]. After V_2 is discharged, V_3 switches to vdd. This means that M5 is conducting current and M16 switches and the current flow begins to charge the spike-width capacitor until it reaches the threshold voltage of the reset transistor M11.

*M3 in the case of IF neuron

LIF Power Consumption Overview

The main power consumption comes from the short-circuit currents of inverter 3 during switching time and in the DC current flowing through the source-follower during membrane integration. This is because inverters 1 and 2 have a significant power reduction from the employment of the current mirrors. During off periods, the LIF neuron behaves similarly to the IF neuron but in this case, the leakage current module discharges any accumulated potential reducing noise effects. When V_{ref} is low, power consumption is dominated by short-circuit currents and depends mainly on the neurons firing rate. Typically, the refractory period is used to limit the neuron's maximum firing rate. Additionally, the current mirrors providing current to inverters 1 and 2, make them switch very fast and this reduces power consumption further. The same design considerations were used for the LIF neuron as for the IF neuron. Therefore, power consumption is not reduced as much as possible to achieve varying integration times for small current amplitudes to theoretically improve accuracy. Although power reduction was not the first prioritized design parameter, this neuron achieved a lower power consumption than the IF neuron during post-layout simulations, further discussed in Section 4.6.

4.2.1 IF Circuit Sizing

The same design considerations used for the IF neuron as relevant in the circuit sizing of the LIF neuron. The LIF neuron does have more flexibility in minimizing switching delays since it contains the spike-width module. The membrane capacitor was aimed to be reduced as much as possible to reduce area. It is noticeable that because of the added circuit components of the LIF neuron, a capacitance half the size of the one for the IF neuron can be used. This smaller capacitance still results in varying integration times for small current amplitudes. Although a smaller membrane capacitor was able to be used, the spike-width capacitor added to the area consumption. To reduce area, the minimum capacitance possible for Global Foundries 180nm CMOS technology was used. The input parameters of the circuit will be discussed during the implementation of the temporal encoders in Section 4.4 and 4.5. Components were minimum sized when possible, but the LIF neuron still needed some delay increases to produce detectable spikes since a minimum sized spike-width capacitor was used. To start the design of the LIF neuron, the basic CMOS circuits were sized individually and continuously adjusted based on simulation results. Table 4.3 shows the final transistor sizes of the IF neuron.

The LIF neuron has the same basic CMOS circuits as the IF neuron where it is mostly made up of three inverters but additionally, a current mirror is used in the LIF neuron. Since the LIF neuron is an extension of the IF neuron, the circuit sizing of the components found in both the IF and LIF neuron has the same process. The only difference in size modulation was the effects caused by the additional node connections at the input node, node V2, node V3, and at the power supply of inverters 1 and 2. For size modulations that increased switching delays, the LIF neuron didn't need as wide of a channel width as the IF neuron components. The lengths of inverter 3 were used for power optimization unlike in the design of the IF neuron and will be discussed in Section 4.2 **LIF Power Saving Techniques**. The design of the current mirror module consists of matching the sizes of the PMOS transistors in series with inverters 1 and 2 to their corresponding inverter PMOS. Inverter 1 is made up of PMOS M15 & M3 and NMOS M4 and inverter 2 is made up of PMOS M5-M6 and NMOS M7-M8. PMOS M15 and M5 are part of the current mirror produced by the diode connected PMOS M14. The size of M15 was matched to the size of M3 and the size of M5 was matched to the size of M6. This matching minimizes delay and circuit-mismatch issues. The diode connected PMOS M14 was kept at a minimum size, but this resulted in a weak positive feedback current. Although it did reduce power consumption, it can be reduced further by increasing the size of M14. Making the current mirror transistors M15 and M5 be bigger than the diode connected one allowed for a stronger current flow in the inverters while still keeping positive feedback at the input.

The two additional connections at the input from the positive feedback module and the leakage current module caused a different design approach to be used for the source follower. The source follower M1 transistor increased W to increase the current drawn from the power supply but channel-length modulation reduced operability, so it was kept at a minimum. Therefore, M2 L was used to optimize power and will be discussed in Section 4.2 **LIF Power Saving Techniques.** The width of M2 was used to change the current speed of the source follower and increase integration time. The sizing of the positive feedback transistor M13 was guided by the need for a stronger current. So, to increase drain current, the W increased, and the length was kept at a minimum to match the current mirror transistor M14. The last additional connection at the input is the leakage current module. The leakage current generator transistor M12 is kept at a minimum size and its bias voltage, Vleak, is modulated based on simulation results. It's important to keep Vleak in mind at changes in circuit operation due to size modulation because it needs to be fine-tuned for each change in the neuron's threshold voltage, even minor ones. An example of Vleak modulation is in Section 4.4.2. The additional connection at node V3 creates the spike-width module and its operability depends on the transistor M16. M16 sets the refractory period by setting the maximum length of time that spike generation is allowed. The spike-width NMOS M16 has a big area to allow spikes to occur at the longest integration time that the neuron produces. M16 controls the refractory period during OFF clock periods and can be set to limit how far within the sampling period a spike can occur. It controls how long the refractory period is by increasing W & L to increase the time period in which a spike can occur within the sampling period.

LIF Power saving techniques

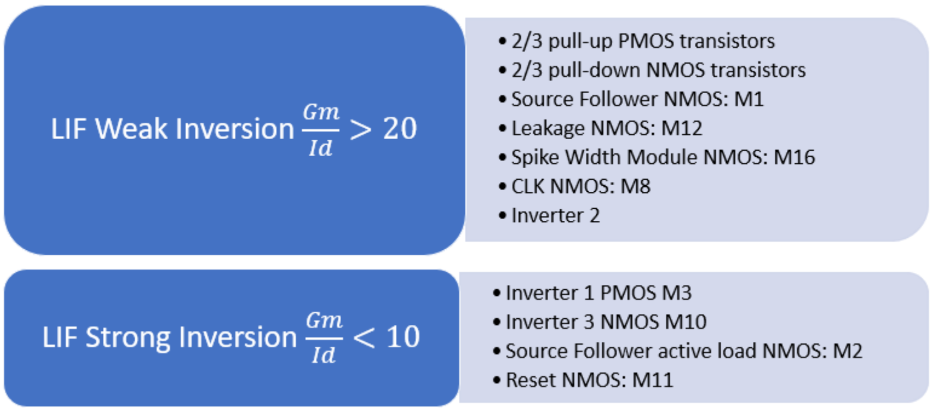
Since detectable spikes are generated after the area optimization and integration time optimization, circuit components that don't affect the neuron's operability are modulated to optimize power. For the same reasons as the IF neuron, the LIF neuron has limited options to optimize power consumption due to the prioritization of area, spike width, and integration time. The inverter 3 and source follower are the most power consuming circuit components and consequently M2 and M9 are the biggest transistors due to needing a strong current. This is different from the power consumption of the IF neuron because the current mirror utilized by

Table 4.3 LIF neuron CMOS sizes

LIF Neuron Sizes	CMOS	W (m)	L (m)	Fingers	gm/Id (V^{-1})
M1	NMOS	1u	180n	-	28.8
M2	NMOS	5u	2u	2	1.92
M3	PMOS	3u	1u	3	0.52
M4	NMOS	1u	600n	-	28.7
M5	PMOS	2u	180n	3	31.4
M6	PMOS	2u	180n	2	31.6
M7	NMOS	460n	180n	-	29.8
M8	NMOS	2u	600n	2	28.7
M9	PMOS	4u	300n	4	31.4
M10	NMOS	330n	300n	-	0.61
M11	NMOS	460n	180n	-	0.49
M12	NMOS	460n	180n	-	24.9
M13	PMOS	5u	180n	3	31.4
M14	PMOS	460n	180n	-	32.2
M15	PMOS	3u	1u	3	33.4
M16	NMOS	3u	1u	2	30.2

inverter 1, allows for the fast switching of the inverters and reduces its power consumption. Inverter 3's input is connected to the gate of the positive feedback transistor M13, and this reduced the channel-length modulation limitation of the inverter 3 used in the IF neuron. The lengths of inverter 3 were able to be increased by 1/3 of the minimum to reduce power consumption. The technique used to reduce the power consumption of the source follower was to modulate the length of M2. Due to the additional circuit component, M1 was not able to increase its length without jeopardizing functionality so the L of M2 was increased instead. This modulation was not as effective as increasing the L of M1, but it did reduce the power by some.

More subthreshold techniques were used in the LIF neuron to control power consumption compared to the IF neuron. From Table 4.3, it shows that 12 out of the 16 CMOS devices of the LIF neuron operate under subthreshold techniques based on their gm/id parameter gathered from their DC operating point. Figure 4.11 shows a summary of the inversion regions that the transistors operate in. Most of the pull-up and pull-down transistors' operation fell under the weak inversion region with the exception of M3 and M10. M3 operated in weak inversion in the IF neuron. The only difference was the length of the transistor. The


LIF Weak Inversion $\frac{Gm}{Id} > 20$

- 2/3 pull-up PMOS transistors
- 2/3 pull-down NMOS transistors
- Source Follower NMOS: M1
- Leakage NMOS: M12
- Spike Width Module NMOS: M16
- CLK NMOS: M8
- Inverter 2

LIF Strong Inversion $\frac{Gm}{Id} < 10$

- Inverter 1 PMOS M3
- Inverter 3 NMOS M10
- Source Follower active load NMOS: M2
- Reset NMOS: M11

Figure 4.11 LIF neuron inversion region summary based on $\frac{Gm}{Id}$

only two other transistors that fell under strong inversion operation were the source-followers active load M2, similar to the IF neuron operation, and the reset transistor M11. Under the same size modulations, M11 operated differently based on which neuron was using it. The active load of the source follower did not use subthreshold techniques because, $V_{ref} > Vth_N$. Vref was higher than VthN because of the priority of increasing the integration time. The operation region of M2 is easily changeable since it will operate in the subthreshold region when $V_{ref} < Vth_N$ and this size modulation has the effect of significantly reducing power consumption since it was the second most power consuming circuit component in the LIF neuron.

The same design parameter prioritization used for the IF neuron design was used for the LIF neuron design. Since this neuron had more components than the IF neuron, the size increases of transistors were limited more to achieve a small area while still achieving comparable functionality. It was noticeable that inverter 3 required the biggest width and aspect ratio to conduct the circuit operation effectively, similar to the IF neuron functionality. Overall, this neuron possesses the ability to encode an analog input signal into voltage spikes and will be implemented into temporal encoders in Section 4.4 and 4.5.

4.2.2 LIF Layout

The proposed IF neuron design is designed using the Global Foundries 180nm standard CMOS process and the Cadence Virtuoso layout. Figure 4.12 shows the final layout of the transistor sizes within the PR boundary. The capacitor layers are separated next to the final circuit layout for visual simplicity but in the final circuit layout they fall within the PR boundary enclosing most of the CMOS transistors. This layout achieved a final area of 14um by 13um for a total of $182um^2$ with 18 circuit components. Although there are more components in the LIF neuron, it achieved a smaller area than the IF neuron. The layout process for this neuron was the exact

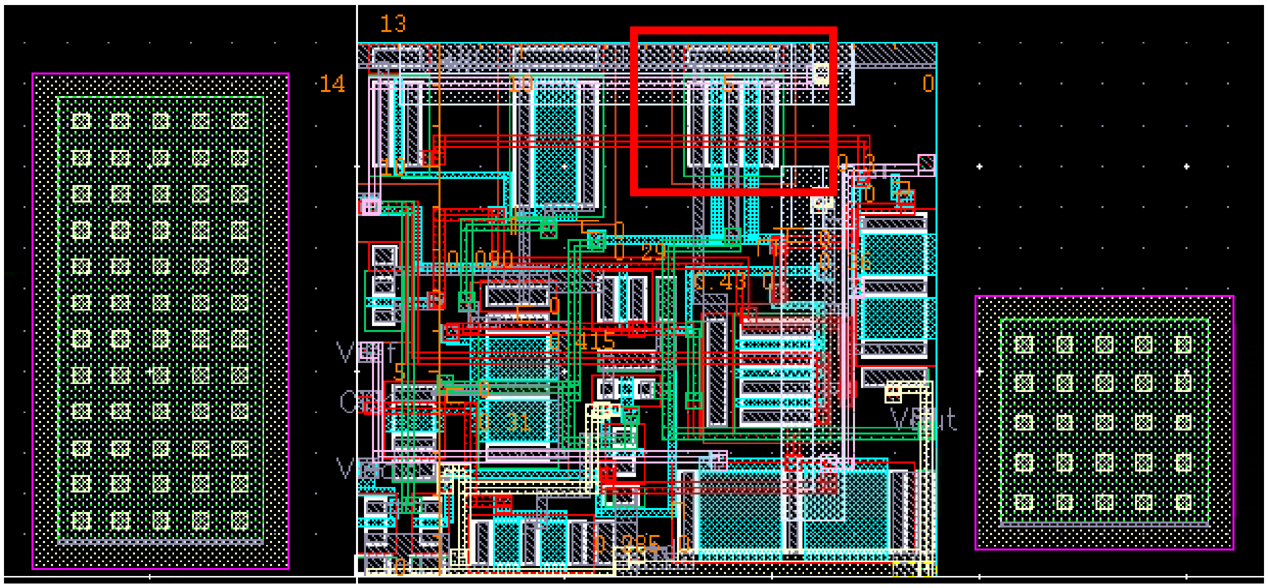


Figure 4.12 LIF neuron layout

same as the IF neuron with a couple exceptions. For the LIF neuron, the PR boundary was guided by the two capacitor areas and added the minimum distance required for metal 5 to be next to each other. The TOP metal via wasn't a challenge for this neuron because of the lack of capacitor layers at the top right corner of the neuron. The Tetris portion of the design process was the most difficult because in order to fit all CMOS devices within the PR boundary set, design rules had to be strictly followed and most transistors employed the minimum distance required between objects. In Figure 4.12, the red box encloses transistor M9. This transistor is part of inverter 3 and had one of the biggest area consumptions and the biggest power consumption. Post-layout simulations showed a power consumption drop of three decimal points which suggests that M9 benefits greatly from being placed away from the effects of the electric fields generated by the capacitor layers. The layout of the LIF neuron did require a small change in Vleak for correct operability in post-layout simulations.

4.3 Temporal Encoding

Temporal encoders are desirable because they reduce the preprocessing of analog signals by encoding signals directly through a one step process which results in a much faster topology than the typical two-step process of ADCs for the digital domain. Encoding signals in the analog domain is preferable because raw sensory data requires a high-performing analog circuit which eases the implementation of analog encoders and allows the preprocessing of signals to become a one step process since signals are encoded directly without the need of domain changes. In high data rates, digital signal processing struggles to keep a low power design since it requires significantly more processors for additional "smart" applications. The use of temporal encoders for preprocessing signals for neuromorphic computing signal processing produces a low area and low power design when compared to the digital domain since it requires less steps and thus, less components. For latency encoding, the input signal's amplitude is encoded within the integration time that it takes for a spike to happen. As can be seen by the latency encoder example in Figure 4.13. At the clock trigger, a sampling period begins and the time that it takes for the capacitor to charge enough to produce a spike is the integration time. The integration time of the first spike to appear in a sampling period is used for latency encoding. On the other hand, ISI encoding uses the relative time between spikes to encode the analog input's amplitude. As can be seen in Figure 4.13, the distance between spikes varies for two different stimuli. The distance between spikes can be found by using the integration times of the spikes and using them to produce the formulas on the left of the ISI encoder in Figure 4.13.

All four encoders have a sampling frequency of 50kHz. The latency encoders achieved an average energy consumption per spike of 277nJ and 316pJ for the IF-based and LIF-based latency encoders accordingly. The ISI encoders achieved an average energy consumption per spike of 1.07uJ and 901nJ for the IF-based and LIF-based ISI encoders accordingly. The energy consumption is simulated by integrating the current drawn from the power supply per every spike and multiplying it by Vdd. Since the spike widths vary based on the delay of generation,

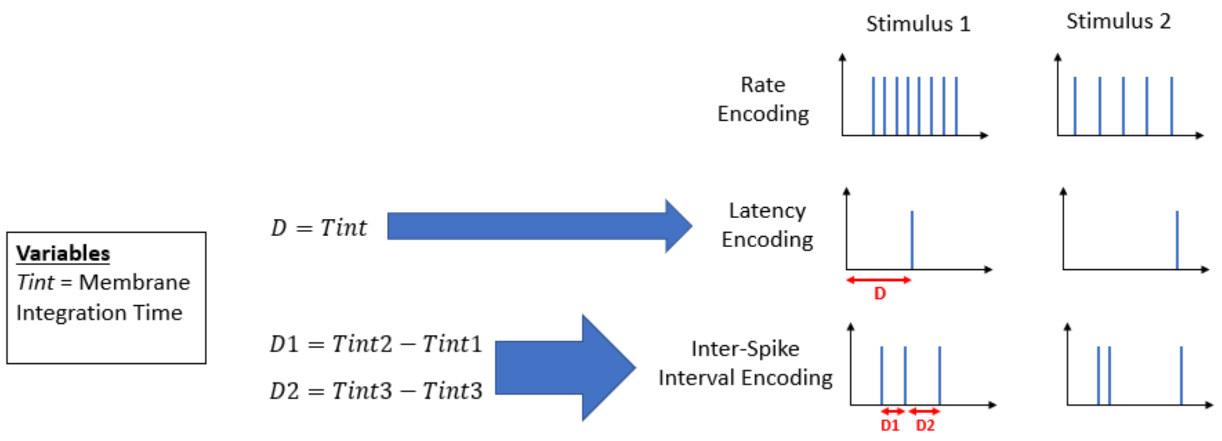


Figure 4.13 Temporal encoding using the integration time of spikes

the average energy per spike was calculated. The IF-based encoders achieved wider ranges of integration times and the LIF-based encoders achieved wider detectable spikes and smaller neuron areas. The input parameters of all four encoders were kept relatively the same for easier comparison. The neuron sizing of the IF and LIF neuron served mainly to produce detectable spikes by increasing the delay and integration times of the produced spikes. Therefore, the input parameters can be used solely to decrease power consumption while keeping the already optimized design parameters relatively the same. Changes to the neuron operation will change based on input parameter changes but these can be minimal changes to not affect the neuron operability while reducing power consumption.

4.4 Latency Encoding Results

The latency encoder only requires one neuron. The input parameters of the IF-based and LIF-based latency encoders can be seen in Table 4.4. Highlighted in red are the design parameters most useful to reduce power consumption. Because power is proportional to integration time, these highlighted parameters will also affect integration time. To increase integration time, V_{ref} was kept high but if power optimization was prioritized, V_{ref} would decrease power consumption by having a smaller amplitude. But for very low V_{ref}values, the power consumption increases because it makes the source follower act as a low-pass filter as mentioned in Section 4.1 **IF Power Consumption Overview**. Since V_{ref} was limited to optimize

Table 4.4 Latency encoders input parameters

Latency Input Parameters			
IF Neuron Based		LIF neuron Based	
Vdd	1.8 V	Vdd	1.8V
Vref	900 mV	Vref	900 mV
Input Current Frequency	10 kHz	Input Current Frequency	10 kHz
Input Current Offset	120 nA	Input Current Offset	120 nA
Input Current Amplitude	50 nA	Input Current Amplitude	50 nA
Clock Period (Frequency)	20 us (50 kHz)	Clock Period (Frequency)	20 us (50 kHz)
Clock Pulse Width	800 ns	Clock Pulse Width	800 ns
Cmem	200 fF	Cmem	100 fF
		Spike Width Capacitance	49 fF
		Vleak	330 mV (335 pre)

integration time, the input current and the clock signal was used to reduce power. Decreasing input current increases integration time significantly because of the small membrane capacitor which increases power consumption. Therefore, my design decreased the input current offset by tens of nanometers so that the integration time increased slightly while benefitting from a reduction in power consumption from the smaller input current. Since the amplitude of the current signal is small, these neurons are sensitive to small input changes making them better able to distinguish small signal changes at the input.

The last highlight input parameter is the clock pulse width. The clock pulse width can be used to lower power consumption because while the clock pulse is on and triggering the sampling period, the remaining duration of the pulse forces V_2 to discharge more quickly and thus, the capacitor charges slightly faster. To optimize power by using the clock pulse, the width of the clock pulse was increased to slightly less than the shortest integration time that occurred during spike generation. Typically, the clock pulse width is approximately 10-20% of the integration time but for this design it's 25% to overlap with the integration times as much as possible without allowing more than one spike to occur. When the neuron fires during the clock pulse, it will keep firing until the clock pulse discharges and the refractory period is triggered. So, the increase of the clock pulse is limited to the integration time of the fastest generated spike. The LIF neuron requires the modulation of Vleak at every input parameter change so it's advisable to change the input parameters of the LIF-based encoders by very small increments so that the neuron will keep some operability and Vleak can be adjusted accordingly. The neurons used for the two proposed latency encoders were simulated post-layout.

4.4.1 IF-based Latency Encoder

Figure 4.14 shows the simulation result of the IF-based latency encoder. The integration time of each spike varies as the input current amplitude changes. By looking at the red Vspike signal and the blue input current signal, we notice there is a shorter charging period at the highest current amplitude and the longest charging period at the lowest current amplitude. This proves true the ability of the latency encoder to map analog signals into the delay of each spike. The black clock signal serves as a trigger to start the integration process and when there's no clock pulse V_{mem} discharges after a spike is generated. As can be seen from the figure, the spikes are generated quickly but have a minimum width of 16.6ns which is greater than the spike detector required minimum of 10ns. The integration times varied between 2.36us to 5.63us for a small input current amplitude which makes it sensitive to small signal changes. Post-layout simulations showed an energy consumption of 277nJ per spike and needed no alterations to produce good results. Power simulations showed that this encoder is prone to noise effects.

Internally, the IF neuron doesn't have spike width modulation so to overcome this challenge the neuron switching delays can be increased or an additional circuit module is necessary. [1]'s solution to this problem was to use a spike expander circuit. The spike expander required a capacitor and two inverters. Based on the use of an extra capacitor and the size of

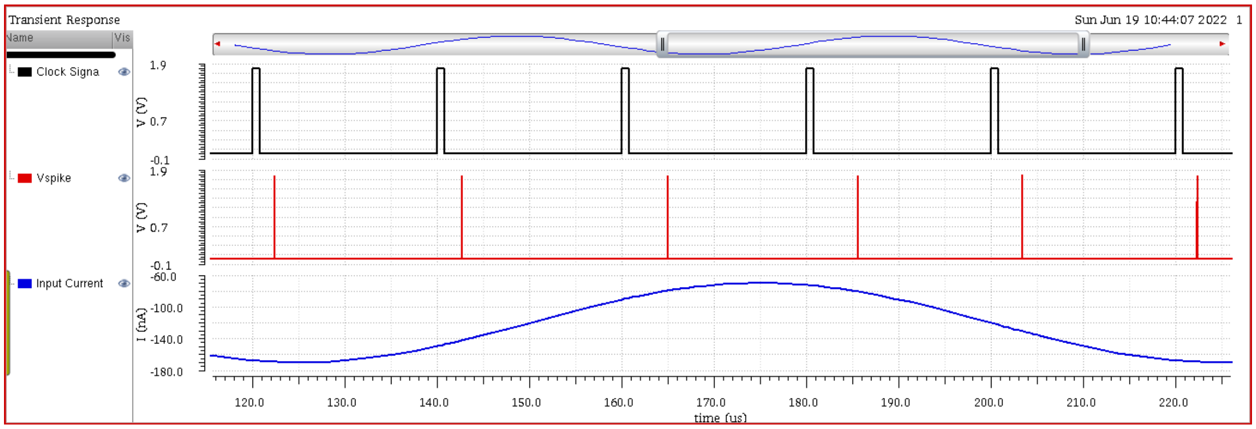


Figure 4.14 Transient simulation of IF-based Latency encoder for a full wave period. In black, input clock signal. In red, output spike train of latency encoder. In blue, input analog current.

the minimum capacitor size possible to be used in the technology used, the size of the neuron will increase significantly. Because of this, the switching delays were increased instead at the expense of power consumption. It should be noted that at higher input current amplitudes the power consumption is reduced but it was kept at an offset of 120nA for easier comparison with the LIF-based encoders. Because the IF neuron is better at producing longer integration times, the current can be increased to lower integration time and achieve similar results to the LIF-based encoder with a smaller noise margin.

4.4.2 LIF-based Latency Encoder

This latency encoder also uses only one neuron, and the external point of reference is the clock signal as well. Figure 4.15 shows the simulation result of the LIF-based latency encoder. The spike's integration time changes based on the input current amplitude which fulfils the encoding functionality of this implementation. It's noticeable in Figure 4.15 that the spikes are generated slower and have a minimum width of 19.35ns which fulfills the spike detectors requirement and is greater than the spike widths achieved by the IF-based latency encoder. The integration times varied between 1.1us to 3.64us for a small input current

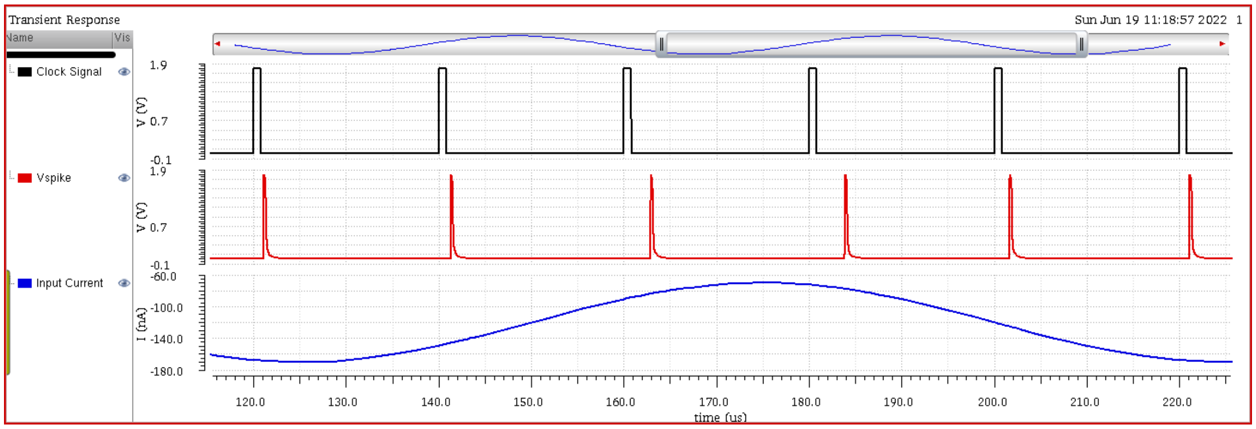


Figure 4.15 Transient simulation of LIF-based Latency encoder for a full wave period. In black, input clock signal. In red, output spike train of ISI encoder. In blue, input analog current.

amplitude which is smaller than the integration time range achieved by the IF-based latency encoder but still makes it sensitive to small signal changes. This is due to the spike-width modulation function block in the LIF neuron circuit. Post-layout simulations showed an energy consumption of 316pJ per spike which is three decimal points smaller than the energy consumption of the IF-based latency encoder. During post-layout simulations, the LIF-based encoder needed a 5mV decrease of its Vleak as can be seen in Table 4.4. This is due to the layout of the circuit changing the capacitance value of the membrane capacitor due to the current flows of the devices within the capacitor. This changed the neuron's threshold voltage slightly. Power simulations showed that this encoder is less prone to noise effects than the IF-based latency encoder.

This encoder was more complex to design because of the added leakage current component. The leakage current is specifically set for each change in the neuron's threshold voltage and should be kept as a low voltage source. During the design process, Vleak never went below 340mV or above 380mV. To understand Vleak modulation better, a design example is presented. When changing V_{ref}, the neuron's threshold voltage changes and therefore, V_{leak} must be adjusted. The LIF neuron design for the latency encoders used a V_{ref} of 0.9V and during pre-layout simulations used a V_{leak} of 0.335V. Two cases will be presented to provide an example of pre-layout input parameter changes that's also useful for post-layout input parameter adjustments. In case 1, V_{ref} reduces to 0.7V. In case 2, V_{ref} increases to 1.2V. In case 1, to start adjusting Vleak it was increased by 20mV to 0.355V. This resulted in a circuit operation similar to the one in Figure 4.16. The highlighted signal is the membrane voltage of a neuron. The rest of the signals are not relevant and can be ignored except for the square wave resulting from the clock signal. It's noticeable that the membrane voltage beings charging at the trigger of the clock pulse but does not trigger a spike generation once it reaches the neurons threshold voltage and doesn't discharge until after the next clock pulse triggers a new sampling period. To fix this operation fault, since V_{ref} was decreased, V_{leak} was decreased to 0.340mV. This resulted in correct spiking at all sampling period but did decrease the integration which allowed for the fastest generated spike to fall within the clock pulse and fire a second spike.

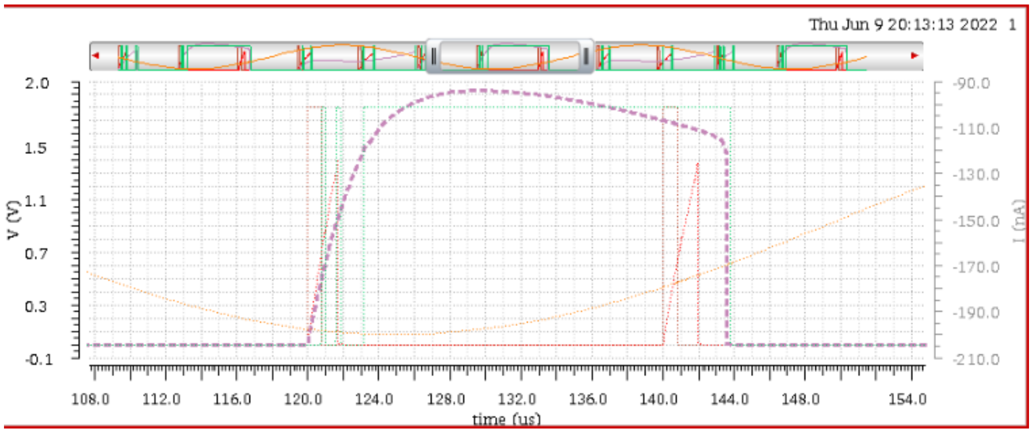


Figure 4.16 Case 1 Vleak adjustment: Purple signal represents overcharged membrane voltage without triggering a spike; needs Vleak adjustment.

Case 1 would need further adjustments to reduce power consumption by only allowing one spike to generate.

In case 2, the same procedure was applied and the original Vleak was increased by 20mA to 0.370V, since V_{ref} was increased. This had the same result as case 1 and the membrane voltage behaved similar to the one in Figure 4.16. To fix this, Vleak was increased further to 0.375mV and this resulted in the best possible spiking functionality based on the parameters used, shown in Figure 4.17. The blue signal is the membrane voltage functionality that resulted in the latest V_{leak} adjustment, and the red signals are the spikes generated from this voltage signal. The rest of the signals are not relevant and can be ignored. Case 2 could not be enhanced further using solely V_{leak} modulation, so the integration time was increased while modulating V_{leak}. This can be done by increasing the membrane capacitance or reducing the input current followed by a small reduction in V_{leak}. Since the membrane capacitance was fixed for this design, the input current was changed. To be able to make informed V_{leak} modulations, the input current offset was reduced by 10nA at a time and Vleak was adjusted at each offset reduction. 10nA of current reduction requires a reduction of around 3mV to Vleak. So, to fix the functionality in Figure 4.17, the input current offset was reduced by 10nA and V_{leak} was

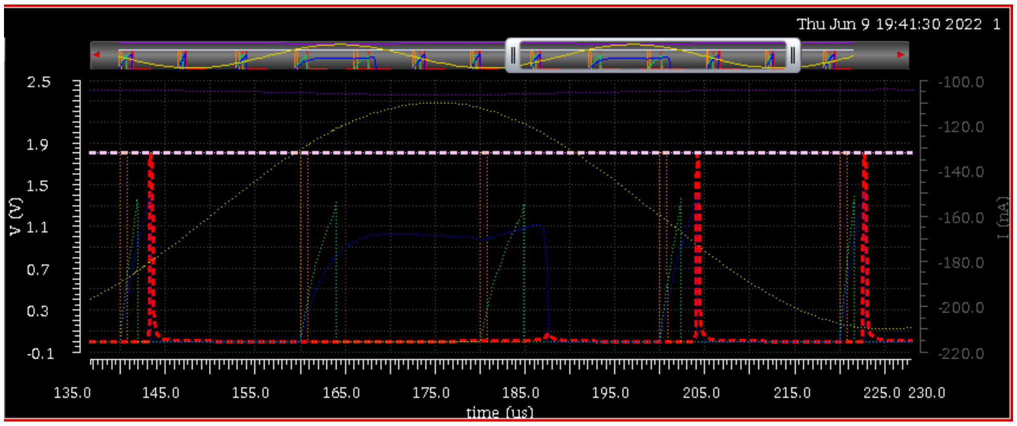


Figure 4.17 Case 2 Vleak adjustment 1: Blue signal represents overcharged membrane voltage. Red signal represents respective output voltage spikes. Vleak adjustments needed.

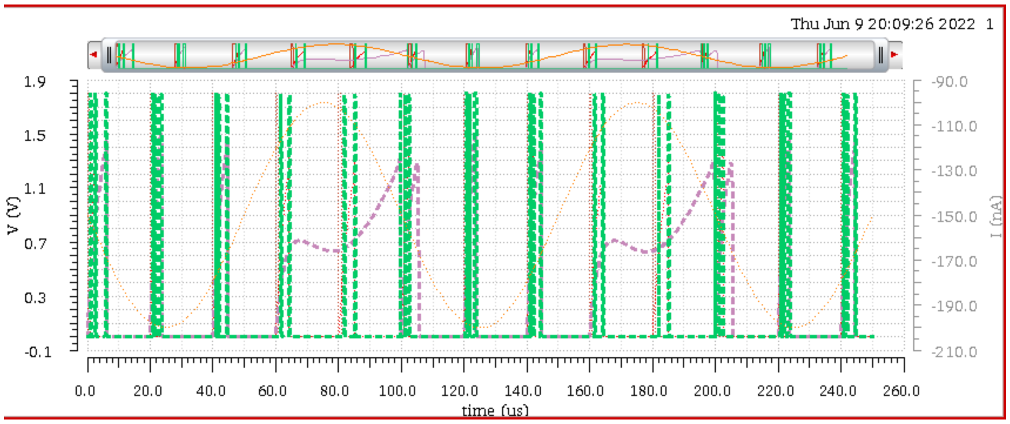


Figure 4.18 Case 2 Vleak adjustment 2: Purple signal represents overcharged membrane voltage. Green signal represents a three-neuron ISI spike train output. Vleak adjustments needed since purple signal doesn't trigger spikes at low input current amplitudes.

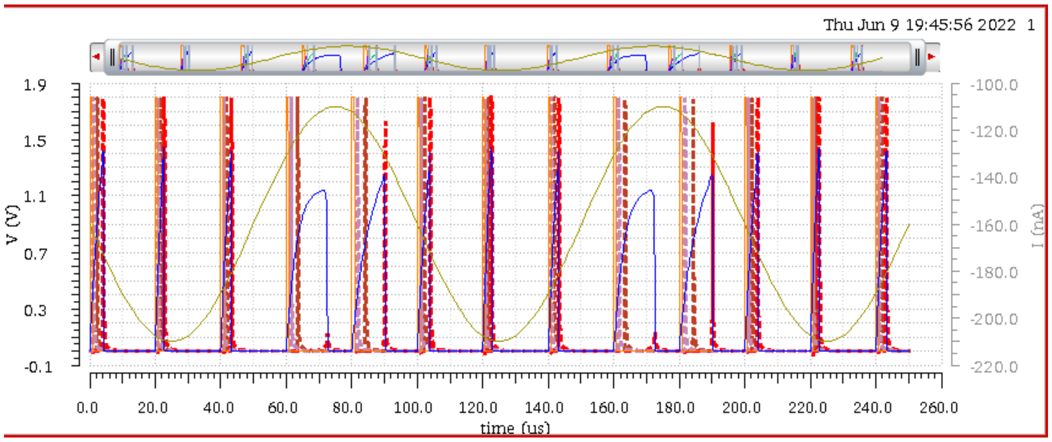


Figure 4.19 Case 2 Vleak adjustment 3: Blue signal represents overcharged membrane voltage. Red signal represents respective output voltage spikes.

reduced by 3mV. This resulted in a circuit functionality similar to Figure 4.18. On this figure, the pink signal is the membrane voltage of this case, and the green signal can be ignored as they are the spikes generated by an ISI encoder. After this V_{leak} modulation, V_{leak} was reduced by 1mV until full functionality was achieved. The first 1mV reduction resulted in a circuit functionality similar to Figure 4.19 showing it was closer to full circuit functionality. On this figure, the blue signal is the membrane voltage for this case and the red spikes are the corresponding spikes generated by this neuron.

4.5 ISI Encoder results

The ISI encoder requires three spiking neurons integrated together through a NOR gate and subsequently a final inverter to produce a spike train of three spikes per sampling period. A spike train is a series of spikes with differing intervals as shown in Figure 4.13. As mentioned in Section 4.3, the intervals are calculated using the integration times of the spikes within one sampling period. The integration time of each neuron is a function of their capacitance and

Table 4.5 ISI encoders input parameters

ISI Input Parameters			
IF Neuron Based		LIF neuron Based	
Vdd	1.8 V	Vdd	1.8V
Vref 1	600 mV	Vref1	500 mV
Vref2	1.2 V	Vref2	1.1 V
Vref3	1.5 V	Vref3	1.2 V
Input Current Frequency	10 kHz	Input Current Frequency	10 kHz
Iput Current Offset	150 nA	Iput Current Offset	150 nA
Input Current Amplitude	50 nA	Input Current Amplitude	50 nA
Clock Period (Frequency)	20 us (50 kHz)	Clock Period (Frequency)	20 us (50 kHz)
Clock Pulse Width1	800 ns	Clock Pulse Width1	800 ns
Clock Pulse Width2	1.5us	Clock Pulse Width2	1.5us
Clock Pulse Width3	2us	Clock Pulse Width3	2us
Cmem1	180 fF	Cmem1	90 fF
Cmem2	220 fF	Cmem2	130 fF
Cmem3	250 fF	Cmem3	200 fF
		Spike Width Capacitance	49 fF
		Vleak1	355m
		Vleak2	356m
		Vleak3	362m

threshold voltage. All three neurons need their own input current and would require a current mirror at the input in hardware implementations. The input parameters of the IF-based and LIF-based ISI encoders can be seen in Table 4.5. Highlighted in red are the design parameters most useful to reduce power consumption. Input parameter modulation for ISI encoders have the same design considerations and effects mentioned in Section 4.4. As can be seen in Table 4.5, the three neurons in both the IF-based and LIF-based ISI encoder have differing capacitances and V_{ref}'s to result in differing integration times to create better interval ranges as input current changes. This is important because if the spikes overlap each other at the same time, the NOR gate only outputs 1 spike instead of two or three. Different clock pulse widths were used for each neuron as well to decrease power consumption by increasing the pulse width to the integration time of the fastest generated spike of each neuron, as mentioned in Section 4.4. Since each neuron employs differing capacitances and V_{ref}'s, the LIF-based ISI encoder needed differing V_{leak}'s as well.

4.5.1 IF-based ISI encoder

Figure 4.20 shows the circuit schematic of the ISI encoder. As can be seen it is made up of three spiking IF neurons, a NOR gate, and an inverter. Figure 4.21 shows the simulation results of the IF-based ISI encoder. We can see that the spikes are generated quickly and that the largest intervals occur when the current is at the lowest amplitude, similar to the IF-based latency encoder spikes. This proves true the ability of the ISI encoder to map the amplitude of an analog input current to the intervals between a spike train. The spikes generated within one sampling period from this neuron are proportional to the number of neurons used. The spikes are generated quickly and the closest they are to the clock pulse the narrower they become. Because of this, most spikes had a minimum width above the spike detector's requirement of 10ns but the fastest generated spike by neuron 1 had a width of 9.1ns. This means that neuron

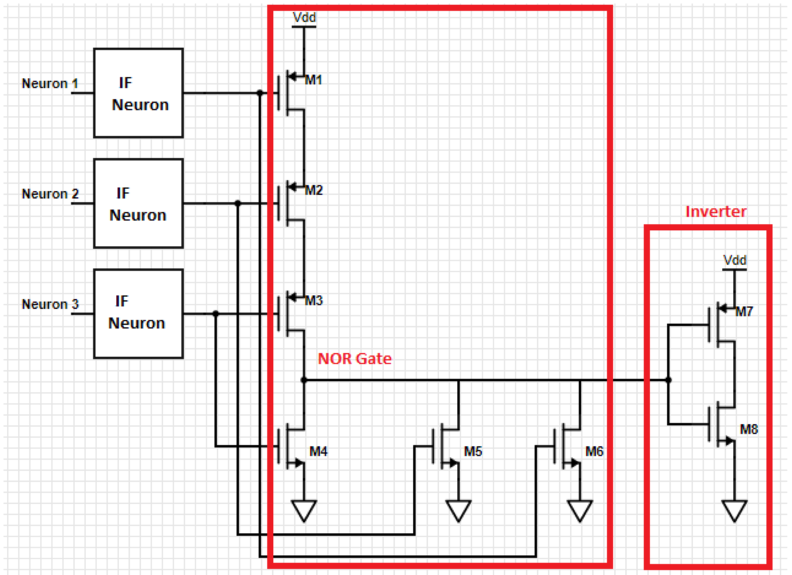


Figure 4.20 IF-based ISI encoder circuit schematic

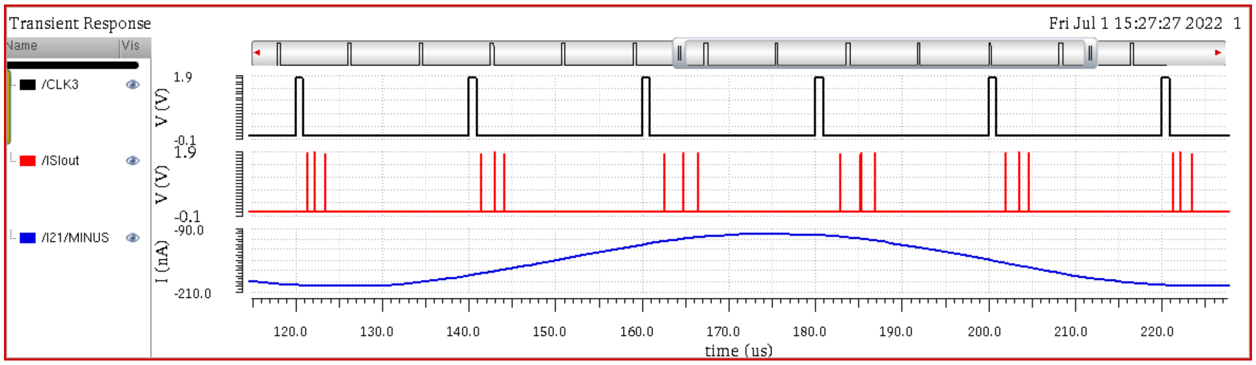


Figure 4.21 Transient simulation of IF-based ISI encoder for a full wave period. In black, input clock signal. In red, output spike train of ISI encoder. In blue, input analog current.

1 requires a slight change in input parameters or membrane capacitance to produce wider spikes by increasing its integration time to occur further away from the clock trigger. Only one spike out of the entire spike trains produces by each sampling period had this challenge so overall, the IF-neuron is capable of making an ISI encoder with detectable spikes. The interval times, D1 and D2, varied between 0.88us to 2.34us and 1.14us to 1.65us for a small input current amplitude which makes it sensitive to small signal changes. D1 varied a lot more than D2 and this is evident in Figure 4.21. D1 varied more than D2 when the current amplitude was the highest and D1 varied less than D2 when the current amplitude was the lowest. Post-layout simulations showed an energy consumption of 1.07uJ per spike train and needed no alterations to produce good results. This is proportional to the energy consumption of the IF-based latency encoder and proves that power is proportional to the number of neurons used. Power simulations showed that this encoder is prone to noise effects, a lot more than the latency encoder. Ideally, the same neuron would be used but only the V_{ref} values would get changed so that no extra designing or layout is needed. But based on the need to change capacitor values, each neuron would need to be designed separately and integrated together during layout.

4.5.2 LIF-based ISI encoder

The LIF-based ISI encoder has the same circuit schematic as the IF-based ISI encoder schematic shown in Figure 4.20 but uses LIF neurons instead of IF neurons. Figure 4.22 shows

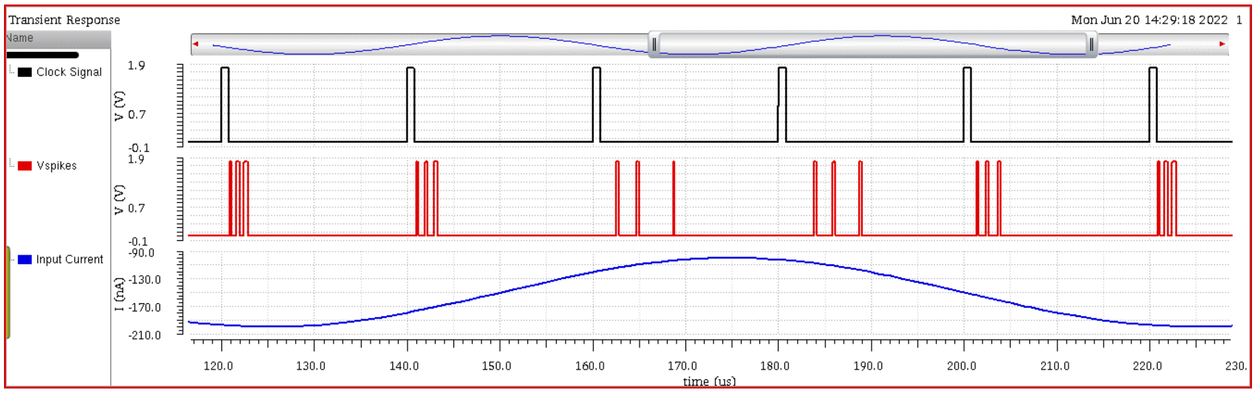


Figure 4.22 Transient simulation of LIF-based ISI encoder for a full wave period. In black, input clock signal. In red, output spike train of ISI encoder. In blue, input analog current.

the simulation results of the LIF-based ISI encoder. Similar to the LIF-based latency encoder, in this simulation it is visually evident that the spike widths are bigger than 10ns. The generated spikes have a minimum width of 11.69ns which fulfill the ability to encode analog signals into the intervals between spikes with detectable spikes. The interval times, D1 and D2, varied between 0.33us to 1.53us and 1.74us to 2.75us for a small input current amplitude which makes it sensitive to small signal changes. The variation times are pretty similar but when looking at Figure 4.22, it is evident that at the highest current amplitudes, the intervals are similar in range but when the input current is at the lowest amplitude, D2 begins to vary more than D1. Post-layout simulations showed an energy consumption of 901nJ per spike train which is similar to the energy consumption of the IF-based ISI encoder but not proportional to the energy consumption of the LIF-based latency encoder. Since encoder energy consumption is proportional to the number of neurons used the expected energy consumption of the LIF-based ISI encoder that uses three neuron is much lower than the energy-consumption achieved. The simulation results for this encoder didn't use neuron layouts and suggest that the LIF neuron layout has the ability to reduce power consumption significantly. Power consumption simulations also showed that the LIF-based ISI encoder is much more prone to noise that the latency encoders but less prone than the IF-based ISI encoder. As mentioned in 4.4, the biggest challenge with using LIF neurons is needing to adjust leakage current at every change in the neuron's threshold voltage.

4.6 Comparisons

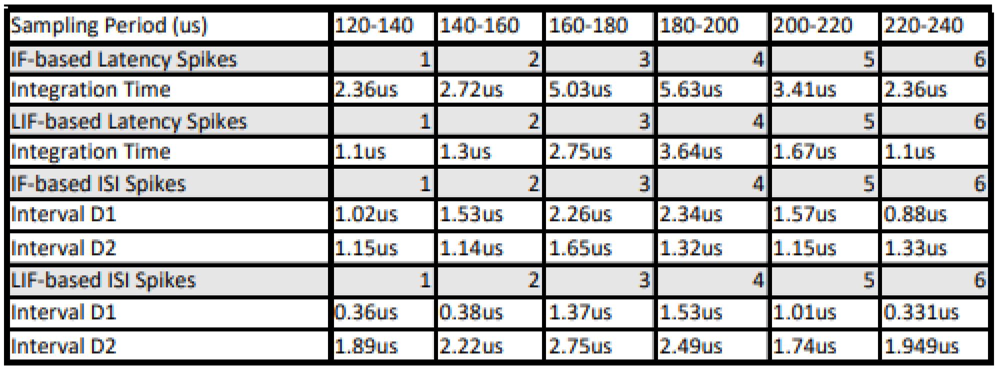
For easier comparison, the input parameters of all four designed encoders were kept relatively the same. For the latency encoders, the input parameters are the same for both except that the LIF-based encoder allowed for a smaller membrane capacitor and had two extra components, the spike-width capacitor and the V_{leak} voltage source. The ISI encoders' input parameters varied more because they had to integrate three neurons with differing spiking times to create a three-spike spike train. Since the amplitude of the current signal is small, these neurons are sensitive to small input changes making them better able to distinguish small signal changes at the input. Neuron sizing was mainly to increase delay and input parameters can be used to increase integration time and reduce power consumption. The encoder best suited to handle a larger noise margin is the LIF-based latency encoder. The most prone to noise was the IF-based ISI encoder. In general, the LIF neuron was better suited to minimize noise effects and the latency encoder was the least affected scheme. The proposed encoders are designed with the Global Foundries 180nm standard CMOS process. Table 4.6 shows design comparisons with similar designs for differing and similar CMOS technologies used.

During spike generation the power dissipation becomes a function of the firing rate, V_{ref}, and vdd. So, power consumption will vary based on the CMOS technology used. The smaller the CMOS technology used, the smaller the power consumption will be. As can be noticed from Table 4.6, the lowest power consumption was achieved by the LIF-based latency encoder and is higher than most of the designs compared in the figure. The only higher power consumption was produced by the rate encoder in [17] since rate encoding requires a lot more spikes to be generated. In order to produce detectable spikes while keeping a small area, the switching delays of transistors were increased which increased power consumption significantly. A range of variation between integration times was prioritized as well for this design to be able encode small input amplitude changes so this increased power consumption further. The LIF-based latency encoder had the best results using post-layout simulations. It was noticeable that creating the layout of M9 away from the capacitors' layers had an effect in dropping the power consumption post-layout. Although the power consumption was high, the proposed neurons achieved the smallest area in the 180nm CMOS technology. The only comparable size is achieved by the proposed IF neuron in [17] using 130nm CMOS technology, achieving a very low area consumption for the single neuron. Although this neuron is comparable in size, it consumes much less power. This is due to having a faster sampling frequency and much smaller integration times per spike. The IF neuron in [17] also produces spikes much smaller than the ones in the proposed designs; meaning that it would need additional circuit components to produce detectable spikes.

Table 4.6 Design comparisons

Encoding Scheme	Power	Energy per spike	Area (m^2)	# of Neurons	Spike Width	Sampling Period	Tech Used
Latency - IF based	73,500uW	277nJ	186.25u	1	9.8ns-18.2ns	20us	180nm
Latency - LIF based	155uW	316pJ	182u	1	11.69ns-19.35ns	20us	180nm
ISI - IF based	214,000uW	1.07uJ	-	3	16.6ns-26.9ns	20us	180nm
ISI - LIF	158,000uW	901nJ	-	3	19.39ns-30ns	20us	180nm
Other Schemes							
IF neuron [6]	1.22uW	-	924u	1	1.5ns	3us	180nm
IF neuron [17]	-	0.48pJ	175.3u	1	2ns	0.03us	130nm
LIF analog neuron [18]	-	2pJ	120u	1	-	5us	65nm
LIF digital neuron [18]	78.2uW	41pJ	538u	1	-	5us	65nm
LIF neuron [19]	55uW	-	28,900u	1	-	-	130nm
LIF neuron [20]	1.1uW	-	-	1	-	-	22nm
Column ADC [21]	9,460uW	-	660,000u	256	-	-	90nm
Rate encoder [17]	577uW	-	1,350u	2	2ns	0.03us	130nm
Latency Encoder [22]	3.5uW	-	-	1	-	-	180nm
ISI encoder [6]	2.5uW	-	4,416u	2	1.5ns	3us	180nm
Parallel ISI encoder [20]	370uW	-	2,560,000u	12	-	-	22nm
Multiplexer 1 [6]	1.3uW	-	12,120u	2	15ns	3us	180nm
Multiplexer 2 [6]	2.6uW	-	21,760u	2	15ns	3us	180nm

Table 4.7 Latency encoder integration times for a full wave period & ISI encoder intervals D1 and D2 for a full wave period



Sampling Period (us)	120-140	140-160	160-180	180-200	200-220	220-240
IF-based Latency Spikes	1	2	3	4	5	6
Integration Time	2.36us	2.72us	5.03us	5.63us	3.41us	2.36us
LIF-based Latency Spikes	1	2	3	4	5	6
Integration Time	1.1us	1.3us	2.75us	3.64us	1.67us	1.1us
IF-based ISI Spikes	1	2	3	4	5	6
Interval D1	1.02us	1.53us	2.26us	2.34us	1.57us	0.88us
Interval D2	1.15us	1.14us	1.65us	1.32us	1.15us	1.33us
LIF-based ISI Spikes	1	2	3	4	5	6
Interval D1	0.36us	0.38us	1.37us	1.53us	1.01us	0.331us
Interval D2	1.89us	2.22us	2.75us	2.49us	1.74us	1.949us

The IF neuron was better at producing a bigger range of integration times for small current amplitudes. It was better at creating a longer integration time variation in the latency encoder implementation with a maximum variation of 3.27us whereas the LIF neuron had a one microsecond of difference at 2.54us. For the ISI encoder, the IF-based encoder achieved the bigger integration time variation for interval D1, but the LIF-based encoder achieved a bigger interval D2, as can be seen in Table 4.7. Overall, all encoders achieved the functionality of encoding a current amplitude into the timing of generated spikes. The IF neuron achieved longer integration times but required more switching delay increases to produce detectable spikes, increasing power consumption. The LIF neuron achieved detectable spikes with less dependence on switching delay which achieved significantly lower power consumption at post-layout simulations.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

Through the inspiration of biological neurons, four temporal encoders were designed and achieved high sensitivity to small current amplitudes producing a very low area design at comparable power consumption. The sampling frequency of the temporal encoders is 50kHz for a speed reduction of the overall neuromorphic computing system. The neurons were optimized to produce detectable spikes (width>10ns) using transistor size modulations for the IF neuron and additional circuit components (spike-width module) for the LIF neuron. The LIF-based latency encoder produced the lowest power consumption comparable to current designs with very low area usage at post-layout simulations. Suggestion that the layout of this design reduced power consumption. The low design area was achieved by making the layout of active devices within the outer capacitor layers and the power consumption reduction of the LIF-based latency encoder was achieved by creating the layout of M9 away from the effects of the capacitor layers. The parameters that could be changed to improve power consumption are the excitation current, sampling period, threshold voltage, or internal transistor sizes. High sensitivity to small current amplitude changes would produce more accurate encoders and the fast frequency allows to increase the speed of the overall process. This functionality could improve the traditional preprocessing of signals and implement neuromorphic computing in designs like machine learning to provide a more efficient signal processing system.

5.2 Future Work

In future applications, a lower power consumption should be prioritized more during neuron sizing since input parameters can increase integration times. The proposed encoders should be tested with a processor to check whether the longer integration times for small amplitude changes leads to an increase in accuracy. Post-layout simulations showed a power consumption drop of three decimal points which suggests that M9 benefits greatly from being placed away from the effects of the electric fields generated by the capacitor layers. To further reduce power consumption, the layout of the neurons should be recreated to have the vdd path fall outside of the capacitor layers to reduce the effects of the capacitor layers' electric field on power consumption. Outside of layout techniques, power consumption can be further reduced by employing more subthreshold techniques. The LIF neuron used more threshold techniques, and its power consumption was significantly reduced at post-layout simulations. The positive feedback module of the LIF neuron could be stronger by employing a bigger M14 transistor to increase the feedback current. The diode connected PMOS M14 was kept at a minimum sized, but this resulted in a weak positive feedback current for the LIF neuron. Although it did reduce power consumption, it can be reduced further by increasing the size of M14. As 3D integration of neuromorphic IC's become more popular, the area of neurons can be bigger to reduce power consumption since 3D integration can reduce overall chip area by stacking circuits and using vias to connect them. This would provide high system speed, high density, low power consumption, and a small footprint, improving neuromorphic computing significantly.

References

1. H. Zheng, N. Mohammadi, K. Bai, Y. Yi, “Low-power Analog and Mixed-signal IC Design of Multiplexing Neural Encoder in Neuromorphic Computing”, *2021 22nd ISDEQ*, pp. 154-159, 2021.
2. B. Razavi, “Design of Analog CMOS Integrated Circuits”, 2nd edition, McGraw-Hill Education, 2017.
3. Y. Yang, “Advanced Analog Integrated Circuit Design Lecture 1: Introduction” Lecture, VT MICS lab, Spring, 2022.
4. C. Zhao and B. Wysocki, “Spike-Time-Dependent Encoding for Neuromorphic Processors,” ACM Journal on Emerging Technologies in Computing Systems, vol. 12, no. 3, September 2015. [DOI: http://dx.doi.org/10.1145/2738040].
5. L. Koutha, (2015). Advanced Encoding Schemes and Their Hardware Implementation for Brain Inspired Computing [MS]. University of Kansas.
6. H. Zheng, “Multiplexing Temporal Neural Encoder & Triplet-Based Recongifugrable STDP Circuit” Lecture, VT MICS lab, Spring, 2022.
7. H. Zheng, J. Anderson and Y. Yi, "Approaching the Area of Neuromorphic Computing Circuit and System Design," 2021 12th International Green and Sustainable Computing Conference (IGSC), 2021, pp. 1-8, doi: 10.1109/IGSC54211.2021.9651627.
8. F. Nowshin, “Advanced Analog Integrated Circuit Design Lecture: Emerging Device Memristor and its application in Neuromorphic Computing” Lecture, VT MICS lab, Spring, 2022.
9. P.E. Allen, D.R. Holberg, “CMOS Analog Circuit Design”, 2nd edition, Oxford University Press, 2002.
10. Y. Yang, “Advanced Analog Integrated Circuit Design Lecture 11: Analog to Digital Converter” Lecture, VT MICS lab, Spring, 2022.
11. Y. Yang, “Advanced Analog Integrated Circuit Design Lecture 8: Neuromorphic Computing” Lecture, VT MICS lab, Spring, 2022.
12. Bio Legend, accessed: June, 2022. https://www.biolegend.com/en-us/synaptic-function
13. Wikipedia, accessed: June, 2022. https://en.wikipedia.org/wiki/Action_potential
14. Wikipedia, accessed: June, 2022. https://en.wikipedia.org/wiki/Biological_neuron_model
15. G. Indiveri, “A low-power adaptive integrate-and-fire neuron circuit”, IEEE, pp. IV820-IV823, 2003.
16. H. Choudhary, A. Kumar, A. Islam, “Propagation Delay and its Robustness Study of Inverter Topologies”, IEEE, 2015.
17. A. Joubert et al., "Hardware spiking neurons design: Analog or digital?", IJCNN, pp. 1-5, June 2012.
18. C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Y. Chen, H. Li, Q. Wu, and H. Jiang, “A spiking neuromorphic design with resistive crossbar,” in 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, Conference Proceedings, pp. 1–6.

19. I. E. Ebong and P. Mazumder, "CMOS and memristor-based neural network design for position detection", Proc. IEEE, vol. 100, no. 6, pp. 2050-2060, Jun. 2012.
20. Y. Kim, Y. Zhang and P. Li, "A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning", Proc. IEEE Int. SOC Conf., pp. 328-333, 2012.
21. C. Zhao, B. T. Wysocki, C. D. Thiem, N. R. McDonald, J. Li, L. Liu, and Y. Yi, "Energy efficient spiking temporal encoder design for neuromorphic computing systems," IEEE Transactions on Multi-Scale Computing Systems, vol. 2, no. 4, pp. 265–276, 2016.
22. C. Zhao, J. Li, and Y. Yi, "Making neural encoding robust and energy efficient: an advanced analog temporal encoder for brain-inspired computing systems," in Proceedings of the 35th International Conference on Computer-Aided Design, Conference Proceedings, pp. 1–6.
23. Electronics Club, accessed: June, 2022. https://www.electronicshub.org/mosfet-as-a-switch/
24. L.M. Chua, P.C. Liu, "Subthreshold current for submicron LDD MOS transistor", *Proceedings of 36th Midwest Symposium on Circuits and Systems*, 1993, pp. 1044-1047 vol.2, doi: 10.1109/MWSCAS.1993.343261.
25. K. Bai*, L. Liu and Y. Yi, "Spatial-Temporal Hybrid Neural Network with Computing-in-Memory Architecture," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2850-2862, July 2021, doi: 10.1109/TCSI.2021.3071956.
26. H. An*, M. S. Al-Mamun, M. K. Orlowski, L. Liu and Y. Yi, "Robust Deep Reservoir Computing Through Reliable Memristor with Improved Heat Dissipation Capability," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 3, pp. 574-583, March 2021, doi: 10.1109/TCAD.2020.3002539.
27. H. An*, Q. An* and Y. Yi, "Realizing Behavior Level Associative Memory Learning Through Three-Dimensional Memristor-Based Neuromorphic Circuits," in *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 4, pp. 668-678, Aug. 2021, doi: 10.1109/TETCI.2019.2921787.
28. S. Liu*, D. Ha, F. Shen, Y. Yi, "Efficient neural networks for edge devices," *Computers & Electrical Engineering*, Vol. 92, 2021, 107121, https://doi.org/10.1016/j.compeleceng.2021.107121.
29. Q. An*, K. Bai*, L. Liu*, F. Shen, Y. Yi, "A unified information perceptron using deep reservoir computing," Computers & Electrical Engineering, Volume 85, 2020,106705,ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2020.106705.
30. K. Bai*, Y. Yi, Z. Zhou*, S. Jere and L. Liu, "Moving Toward Intelligence: Detecting Symbols on 5G Systems Through Deep Echo State Network," in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 10, no. 2, pp. 253-263, June 2020, doi: 10.1109/JETCAS.2020.2992238.
31. H. An*, M. Al-Mamun, Marius K. Orlowski, L. Liu, Y. Yi, "Three-dimensional Neuromorphic Computing System with Two-layer and Low-variation Memristive

Synapses," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, doi: 10.1109/TCAD.2021.3061481.

32. V. M. Gan*, Y. Liang*; L. Li, L. Liu; Y. Yi, “A Cost-Efficient Digital ESN Architecture on FPGA for OFDM Symbol Detection,” Early Access, ACM Journal on Emerging Technologies in Computing Systems (JETC-2020-0100.R1).
33. K. Bai*, Q. An*, L. Liu, and Y. Yi, "A Training-Efficient Hybrid-Structured Deep Neural Network With Reconfigurable Memristive Synapses," IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Early Access), 2019/10/11.
34. C. Zhao*, Q. An*, K. Bai*, B. Wysocki, C. Thiem, L. Liu, Y. Yi, "Energy Efficient Temporal Spatial Information Processing Circuits Based on STDP and Spike Iteration," IEEE Transactions on Circuits and Systems II: Express Briefs (Early Access), 2019/10/4.
35. H. An*, M. A. Ehsan*, Z. Zhou, F. Shen, and Y. Yi, "Monolithic 3D neuromorphic computing system with hybrid CMOS and memristor-based synapses and neurons," Integration (Elsevier), vol. 65, pp. 273-281, 2019/3/1.
36. C. Zhao*, K. Hamedani*, J. Li*, Y. Yi, “Analog Spike-timing-dependent Resistive Crossbar Design for Brain Inspired Computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 8, no. 1, pp. 38 - 50, 2018.
37. A. Ehsan*, H. An*, Z. Zhou, Y. Yi, “A Novel Approach for using TSVs as Membrane Capacitance in Neuromorphic 3D," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (TCAD), vol. 37, no. 8, pp. 1640 – 1653, 2018.
38. C. Zhao*, Y. Yi, and J. Li*, X. Fu, L. Liu, “Inter-Spike Intervals (ISI) based Analog Spike-Time-Dependent Encoder for Neuromorphic Processors,” *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 25, no. 8, pp. 2193-2205, 2017.
39. H. An*, J. Li*, Y. Li, X. Fu, and Y. Yi, "Three Dimensional Memristor Based Neuromorphic Computing System and its Application to Cloud Robotics," *Computers & Electrical Engineering an International Journal (Elsevier)*, vol. 63, pp. 99-113, 2017.
40. C. Zhao*, B. Wysocki, C. Thiem, N. McDonald, J. Li*, and Y. Yi, “Energy Efficient Spiking Temporal Encoder Design for Neuromorphic Computing Systems,” *IEEE Transactions on Multi-Scale Computing Systems (TMSCS),* vol. 2, no. 4, pp. 265 - 276, 2016.
41. Y. Yi, Y. Liao*, B. Wang*, X. Fu, F. Shen, and H. Hou*, “FPGA based Spike Time Dependent Encoder and Reservoir Design in Neuromorphic Computing Processors,” *Journal of Microprocessors and Microsystems: Embedded Hardware Design (Elsevier)*, vol. 46, Part B, pp. 175-183, 2016.
42. C. Zhao*, B. T. Wysocki, Y. Liu, C. D. Thiem, N. R. McDonald, and Y. Yi, “Spike-Time-Dependent Encoding for Neuromorphic Processors,” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 3, pp. 23-46, 2015.
43. O. Shears*, K. Bai*, L Liu, and Y. Yi, “A Hybrid FPGA-ASIC Delayed Feedback Reservoir System to Enable Spectrum Sensing/Sharing for Low Power IoT Devices, 2021 *IEEE/ACM International Conference On Computer Aided Design (ICCAD).*
44. H. An*, M. S. Al-Mamun, M. K. Orlowski and Y. Yi, "A Three-dimensional (3D) Memristive Spiking Neural Network (M-SNN) System," *2021 22nd International Symposium on*

Quality Electronic Design (ISQED), 2021, pp. 337-342, doi: 10.1109/ISQED51717.2021.9424303.

45. H. An*, K. Bai* and Y. Yi, "Three-dimensional Memristive Deep Neural Network with Programmable Attention Mechanism," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 210-215, doi: 10.1109/ISQED51717.2021.9424331.
46. H. Zheng*, J. Handerson*, and Y. Yi, “Approaching the Area of Neuromorphic Computing Circuit and System Chip Design,” 2021 12th International Green and Sustainable Computing Conference (IGSC).
47. H. Zheng*, N. Mohammadi*, K. Bai* and Y. Yi, "Low-power Analog and Mixed-signal IC Design of Multiplexing Neural Encoder in Neuromorphic Computing," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 154-159, doi: 10.1109/ISQED51717.2021.9424267.
48. S. Liu*, and Y. Yi, “Quantization-Aware Training of Spiking Neural Networks for Intelligent Sensing Systems,” *ISCAS 2022: IEEE International Symposium on Circuits and Systems.*
49. C. Lin, and Y. Yi, “FPGA based Reservoir computing with optimized reservoir node architecture,” Accepted, *ISQED 2022: International Symposium on Quality Electronic Design*.
50. K. Bai*, C. Thiem, N. McDonald, L. Loomis and Y. Yi, "Toward Intelligence in Communication Networks: A Deep Learning Identification Strategy for Radio Frequency Fingerprints," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 204-209, doi: 10.1109/ISQED51717.2021.9424319.
51. K. Bai*, L. Liu, Z. Zhou and Y. Yi, "Detection Through Deep Neural Networks: A Reservoir Computing Approach for MIMO-OFDM Symbol Detection," 2020 *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1-7.
52. F. Nowshin*, Y. Zhang, L. Liu and Y. Yi, "Recent Advances in Reservoir Computing With A Focus on Electronic Reservoirs," 2020 11th *International Green and Sustainable Computing Workshops (IGSC),* Pullman, WA, USA, 2020, pp. 1-8, doi: 10.1109/IGSC51522.2020.9290858.
53. H. An*, D. S. Ha, and Y. Yi. 2020. Powering next-generation industry 4.0 by a self-learning and low-power neuromorphic system. In Proceedings of the *ACM International Conference on Nanoscale Computing and Communication (NanoCom '20).* Association for Computing Machinery, New York, NY, USA, Article 6, 1–6. DOI: https://doi.org/10.1145/3411295.3411302
54. S. Liu*, Y. Liang*, V. Gan*, L. Liu and Y. Yi, "Accurate and Efficient Quantized Reservoir Computing System," 2020 21st International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2020, pp. 364-369, doi: 10.1109/ISQED48828.2020.9136986.
55. S. Liu*; L. Liu; Y. Yi, “Quantized Reservoir Computing on Edge Devices for Communication Applications,” Fifth ACM/IEEE Symposium on Edge Computing (SEC '20).

56. K. Bai*, S. Liu*, and Y. Yi, “High speed and energy efficient deep neural network for edge computing,” in the Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, pp. 347-349, Washington D.C., 2019/11/7.
57. C. Zhao*, L. Liu, and Y. Yi, "Design and Analysis of Real Time Spiking Neural Network Decoder for Neuromorphic Chips," in the Proceedings of the International Conference on Neuromorphic Systems, Knoxville, TN, pp. 1-4, 2019/7/23
58. K. Bai*, Q. An*, Y. Yi, "Deep-DFR: A memristive deep delayed feedback reservoir computing system with hybrid neural network topology," in the Proceedings of 2019 56th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, 2019/6/2
59. Bai*, J. Li*, and Y. Yi, “Enabling a New Era of Brain-inspired Computing: Energy-efficient Spiking Neural Network with Ring Topology,” in *Proceedings of IEEE/ACM Design Automation Conference (DAC),* 2018.
60. J. Li*, K. Bai*, L. Liu, and Y. Yi, “A Deep Learning Based Approach for Analog Hardware Implementation of Delayed Feedback Reservoir Computing System,” in *Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED)*, 2018. *(Best Paper Award)*
61. K. Bai* and Y. Yi, “A Path to Energy Efficient Spiking Delayed Feedback Reservoir Computing for Brain-inspired Neuromorphic Processors,” in *Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED)*, 2018.
62. H. An*, M. S. Al-Mamun, M. K. Orlowski, and Y. Yi, “Learning Accuracy Analysis of Memristor-based Nonlinear Computing Module on Long Short-term Memory,” in *Proceedings of Neuromorphic Computing Symposium*, 2018.
63. A. Ehsan*, H. An*, Z. Zhou, and Y. Yi, “Adaptation of Enhanced TSV Capacitance as Membrane Property in 3D Brain-inspired Computing System,” in *Proceedings of IEEE/ACM Design Automation Conference (DAC),* 2017.
64. H. An*, Z. Zhou, and Y. Yi, “Memristor-Based 3D Neuromorphic Computing System and Its Application to Associative Memory Learning,” in *Proceedings of IEEE Nanotechnology Conference*, 2017.
65. H. An*, Z. Zhou, and Y. Yi, “3D Memristor-based Adjustable Deep Recurrent Neural Network with Programmable Attention Mechanism,” in *Proceedings of Neuromorphic Computing Symposium*, 2017.
66. H. An*, M. Ehsan*, Z. Zhou, and Y. Yi, "Electrical Modeling and Analysis of 3D Synaptic Array using Vertical RRAM Structure,” in *Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED)*, 2017.
67. C. Zhao*, J. Li*, H. An, and Y. Yi, “When Energy Efficient Spike-Based Temporal Encoding Meets Resistive Crossbar: From Circuit Design to Application,” in *Proceedings of Neuromorphic Computing Symposium*, 2017.
68. J. Li*, C. Zhao*, and Y. Yi, “Energy Efficient and Compact Analog Integrated Circuit Design for Delay-dynamical Reservoir Computing System,” Special Session in “Hardware in Reservoir Computing”, in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN)*, 2017.

69. C. Zhao*, J. Li*, and Y. Yi, "Analog Spiking Temporal Encoder with Inter-Spike Intervals with Verification and Recovery Scheme for Neuromorphic Computing Systems," in *Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED)*, 2017.
70. A. Ehsan*, Z. Zhou, and Y. Yi, "3D Integration Meets Neuromorphic Computing: A Novel Way to Reach a High Performance and Energy Efficient Computing System," in *Proceedings of IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, invited paper, 2017.
71. C. Zhao*, J. Li*, and Y. Yi, "Making Neural Encoding Robust and Energy-Efficient: An Advanced Analog Temporal Encoder for Brain-Inspired Computing Systems," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2016.
72. H. An*, A. Ehsan*, Z. Zhou, and Y. Yi, "Electrical Modeling and Analysis of 3D Neuromorphic IC with Monolithic Inter-tier Vias," in *Proceedings of IEEE Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2016.*
73. C. Zhao*, J. Li*, L. Liu, and Y. Yi, "Novel Spike Based Reservoir Node Design with High Performance Spike Delay Loop," in *Proceedings of ACM International Conference on Nanoscale Computing and Communication (NanoCom),* 2016.
74. A. Ehsan*, H. An*, Z. Zhou, and Y, Yi, "Design Challenges and Methodologies in 3D Integration for Neuromorphic Computing Systems," in *Proceedings of IEEE International Symposium on Quality Electronic Design (ISQED)*, 2016.
75. C. Zhao*, W. Danesh*, B. T. Wysocki, and Y, Yi, "Neuromorphic Encoding System Design with Chaos Based CMOS Analog Neuron," in *Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 76-81, 2015.
76. A. Ehsan*, Z. Zhou, and Y. Yi, "Three Dimensional Integration Technology Applied to Neuromorphic Hardware Implementation," in *Proceedings of IEEE International Symposium on Nanoelectronic and Information System (INIS),* 2015.
77. S. Liu*, Y. Yi, "Quantized Neural Networks and Neuromorphic Computing for Embedded Systems," 10.5772/intechopen.91835, 2020.
78. K. Bai*, Y. Yi, "Opening the "Black Box" of Silicon Chip Design in Neuromorphic Computing," Bio-Inspired Technology, IntechOpen, 2019.
79. S. Liu*, Y. Yi, "Quantized Neural Networks and Neuromorphic Computing for Embedded Systems," Intelligent System and Computing, IntechOpen, 2019.
80. H. An*, K. Bai*, and Y. Yi, "The Roadmap to Realize Memristive Three- Dimensional Neuromorphic Computing System," Advances in Memristor Neural Networks-Modeling and Applications, IntechOpen, 2019.