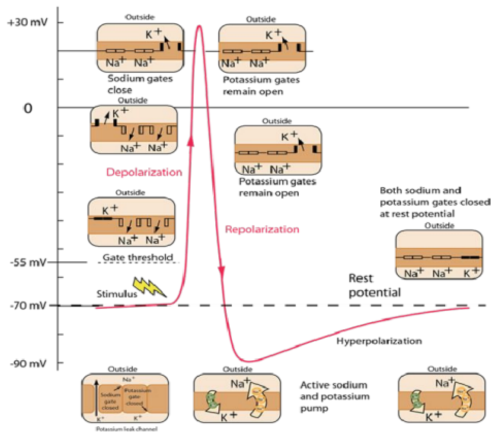
# Chapter 2

# FPGA Implementation of Digital Spiking Neuron

## 2.1 Introduction

Membrane potential in biological neuron rises upon receiving a stimulus as Na+ gate opens and sodium ions rushes into the neuron and voltage increases till it reaches a peak potential then k+ gate open and Na+ closes, k+ ion now move out of the membrane resulting in decrease in potential. This causes spiking in neuron after receiving a stimulus.



+30 mV
0
-55 mV
-70 mV
-90 mV
Sodium gates close
Potassium gates remain open
Depolarization
Repolarization
Potassium gates remain open
Both sodium and potassium gates closed at rest potential
Gate threshold
Stimulus
Rest potential
Hyperpolarization
Active sodium and potassium pump

Figure 2.1: Spiking in biological Neuron [1]

Traditional artificial neural network and convolutional neural networks are unable to simulate the dynamic behaviour of biological neuron. Spiking Neural Network(SNN) can better emulate the behaviour of biological Neuron.

Various spiking model have been proposed to mimic the biological neuron. There are two models at two extremes :

1. Hodgkin-Huxley(HH) model : Model is computationally expensive but can reproduce various spiking pattern as seen in biological Neuron.
2. Leaky integrate and Fire (LIF) model is computationally simple but can simulate the simplest spiking dynamic

Then there is Izhikevich (IZH) model which uses only two equations to model the neuron dynamic yet is able to reproduce a variety of spiking patterns. Therefore it is considered a computationally efficient and biological plausible neuron.

## 2.2 Modelling IZH for Digital Design

IZH model[2] proposed was following:

$$v' = 0.04v^2 + 5v + 140 - u + I$$

$$u' = a(bv - u)$$

$$\text{if } v \geq v_{\text{th}}, \text{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases}$$

where $v$ is membrane potential, $u$ is recovery variable, and I is the synaptic current.
$v' = \frac{dv}{dt}$ and $u' = \frac{du}{dt}$
a, b, c, d is are parameters that decide the spiking patterns of neurons.
Membrane potential is in mV, time scale is ms.

The problem with these equation for digital implementation of system is that it involve quadratic equation, and to model that will require multiplier which increases power consumption and critical delay path, thereby reducing the overall throughput of the design.

The digital design discussed here for IZH modelled is taken from[3], paper avoids the use of multiplier by estimating quadratic function with help of cosh which further can modelled using $2^x$ function as :

$$cosh(x) = \frac{2^{1.44x} + 2^{-1.44x}}{2}$$

This approximation of $v^2$ is valid can be understood using following graphs

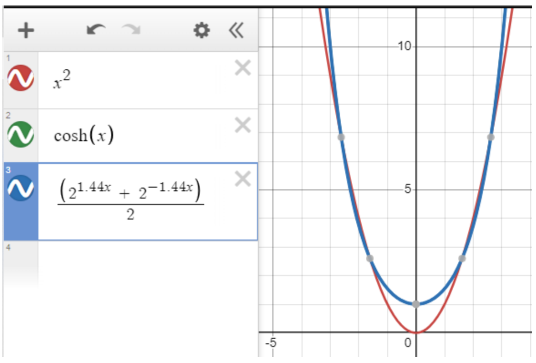


Figure 2.2: Comparison of $x^2$, $cosh(x)$, $\frac{2^{1.44x}+2^{-1.44x}}{2}$

This approximation is usefull because in the design pipeline as x will get values ranging from $\frac{-80}{2^6}$ to $\frac{80}{2^6}$ this allows following:

- x has integer part($x_i$) and fractional part($x_f$) so,

$$x = x_i + x_f$$

$$\therefore 2^x = 2^{x_i} \times 2^{x_f}$$

- x has 1, -1, 0 as integer part($x_i$), so this integer part make $2^{x_f}$ multiply or divide by 2 or multiply by 1. Now this multiplication by 2 can be achieved by single left shift and division by 2 can be achieved by single right shift operation on a given number.
- x has fractional part, and this $2^{x_f}$ can be estimated using
  if $-1 < x_f < 0$ : $2^x_f = 1 + 0.5x_f$
  else $0 < x_f < 1$ : $2^x_f = 1 + x_f$
- So $2^x$ for given range can be approximated by calculating $2^{x_f}$ as mentioned above and shifting $2^{x_f}$ by left shift operation or right shift to multiply or divide by 2 to model effect of $2^{x_i}$. So just by shift and addition operation we avoided using a complete multiplier.

Next we discretize the system of IZH model using euler method so model presented in the paper is

$$v[n+1] = v[n] + dt\,(k_1 \cosh(k_2(v[n] + k_3)) - k_4 - u[n] + I[n]) \tag{2.1}$$
$$u[n+1] = u[n] + dt\,(a(bv[n] - u[n])) \tag{2.2}$$

Values for different parameter as given by author
$k_1 = 380$
$k_2 = 0.0145$
$k_3 = 62.5$
$k_4 = 396.25$
$a = 0.01953125 = 2^{-6} + 2^{-8}$
$b = 0.203125 = 2^{-3} + 2^{-4} + 2^{-6}]$

Table 2.1: c, d Parameters for different Neuron Types

| Neuron Type | c | d |
|---|---|---|
| Regular Spiking | -80 | 15 |
| Fast Spiking | -80 | 7.25 |
| Chattering | -56 | 4.75 |
| Intrinsically Bursting | -61 | 8.5 |

## 2.3 Digital Implementation of Model

### 2.3.1 Number Representation

We will using 2's complement representation for signed numbers.
Secondly we will use fixed point arithmetic to represent floating point number. Total size of register for storing a number will be 21 bits of which 11 bits for integer part and 9 bits for fractional part.

### 2.3.2 Block Diagram

Block diagram given in the paper was modified as:

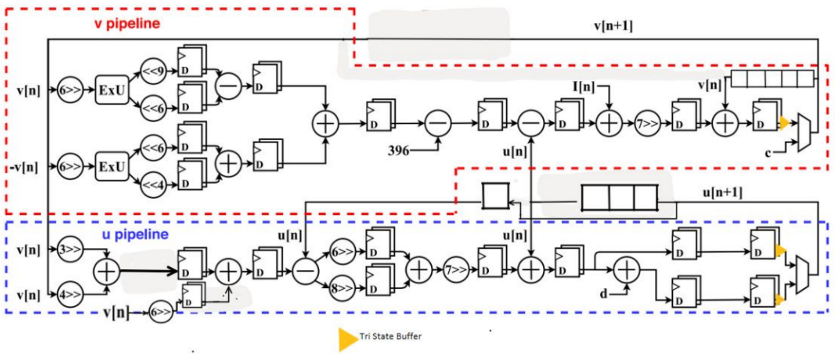


Figure 2.3: Modified Block Diagram for Digital Design

We introduced buffers to avoid the mismatch of v and u value in pipeline. Secondly we introduced tri-state buffer which results in v taking value only when one complete cycle of producing v has taken place. Thirdly we corrected the u pipeline. Exponential Block

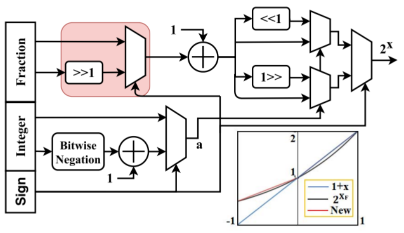


Figure 2.4: Exponential Block

### 2.3.3 Difference in Actual vs Modelled values

Table 2.2: Parameters : Actual vs Modelled

| | Actual Values | Modelled Values |
|---|---|---|
| k1 | 380 | 380 |
| k2 | 0.0145 | 0.015625 |
| k3 | 62.5 | 70 |
| k4 | 396.25 | 396 |

## 2.4 Hardware Implementation

Note : Below in text v, finalv, voltage refers to membrane potential.

### 2.4.1 Code

Code for the project can be accesed using following link to github repository :
Link to Github Repository

### 2.4.2 Simulation Plots

Every plot is have vertical axis in the range -80 to 30 mV
Sampling frequency is $\frac{1}{70ns} = 14MHz$

**Regular Spiking**
Time period of Spikes is $48\mu s$

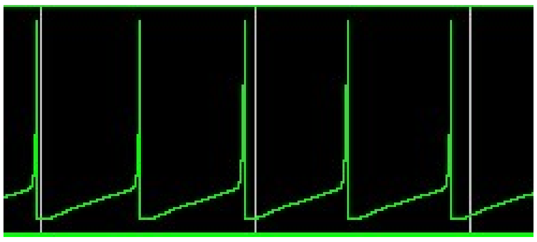


Figure 2.5: Regular Spiking

**Fast Spiking**
Time Period is $25\mu s$

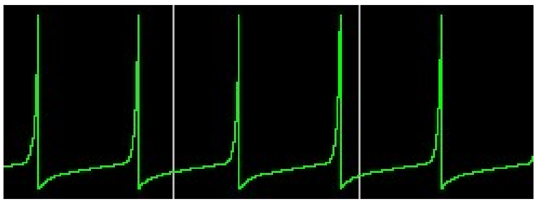


Figure 2.6: Fast Spiking

**Chattering**

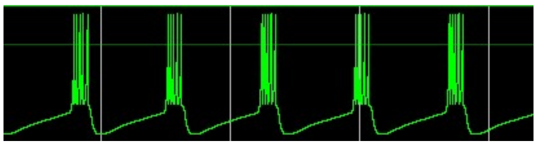


Figure 2.7: Chattering

**Intrinsically Bursting**

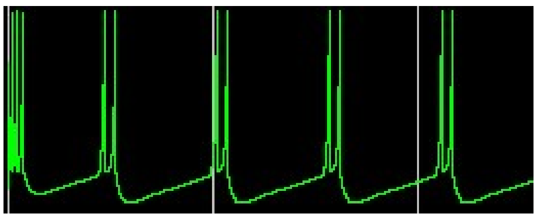


Figure 2.8: Enter Caption

### 2.4.3 Hardware Verification

To verify that the design is synthesizable and shows expected behaviour we can use VIO(Virtual Input and Output) and ILA(Integrated Logic Analyzer). VIO and ILA are available as IP block in Vivado.

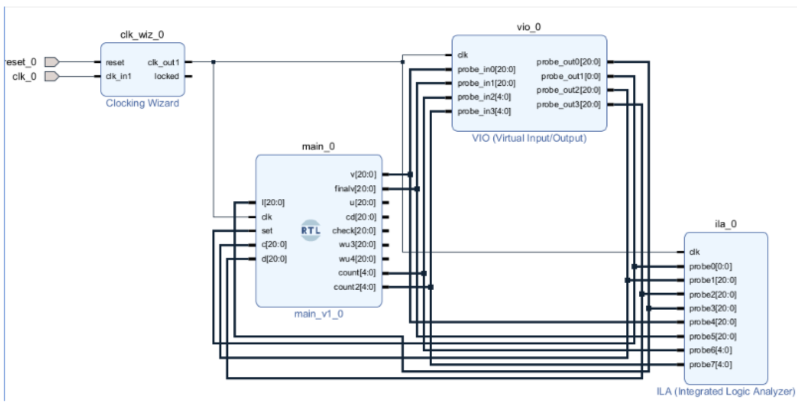
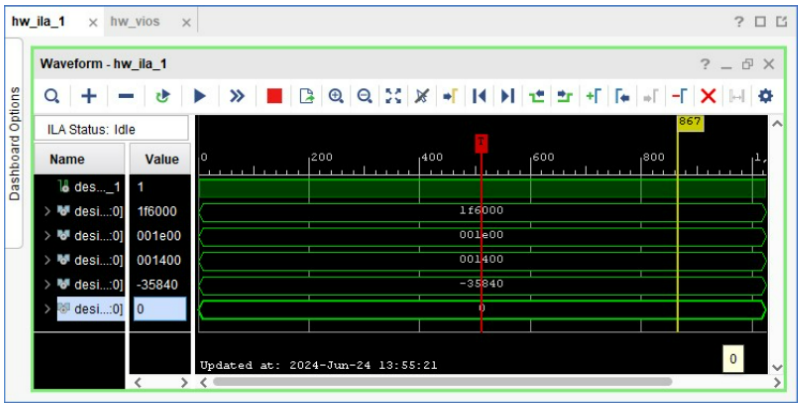


Figure 2.9: Final block diagram with VIO and ILA for testing

Verification observation

- Hardware is initialized properly when set = 1, when set = 1 v(membrane potential) is initialized with -70 mV.



hw_ila_1 × hw_vios ×

Waveform - hw_ila_1

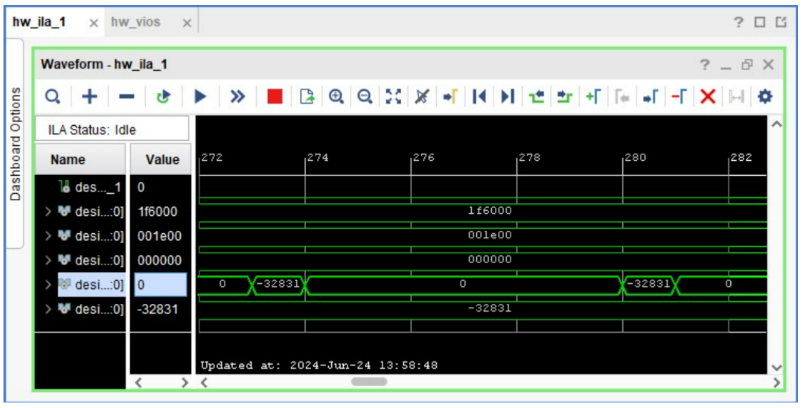
ILA Status: Idle

| Name | Value |
|---|---|
| des..._1 | 1 |
| desi...:0] | 1f6000 |
| desi...:0] | 001e00 |
| desi...:0] | 001400 |
| desi...:0] | -35840 |
| desi...:0] | 0 |

```
Updated at: 2024-Jun-24 13:55:21
```

Figure 2.10: ILA outputs when set = 1

- When current is 0, voltage(membrane potential) is constant, that is $(-32831 \times 2^{-9}) = -64.1mV$.



hw_ila_1 × hw_vios ×

Waveform - hw_ila_1

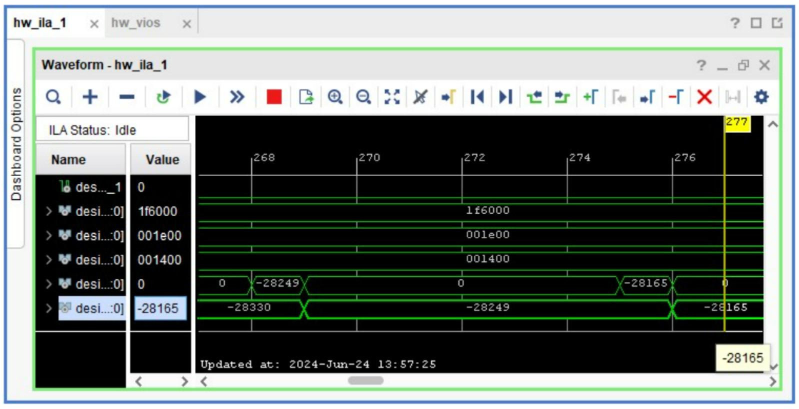
ILA Status: Idle

| Name | Value |
|---|---|
| des..._1 | 0 |
| desi...:0] | 1f6000 |
| desi...:0] | 001e00 |
| desi...:0] | 000000 |
| desi...:0] | 0 |
| desi...:0] | -32831 |

```
Updated at: 2024-Jun-24 13:58:48
```

Figure 2.11: ILA output when current is 0

- When there is some current the data between v(membrane potential) and finalv(another variable in code to record v continuously) flows as expected and in the expected range.



| Name | Value |
|---|---|
| des..._1 | 0 |
| desi...:0] | 1f6000 |
| desi...:0] | 001e00 |
| desi...:0] | 001400 |
| desi...:0] | 0 |
| desi...:0] | -28165 |

```
 268    270    272    274    276    277
                1f6000
                001e00
                001400
 0  -28249       0           -28165  0
 -28330        -28249           -28165
Updated at: 2024-Jun-24 13:57:25        -28165
```

Figure 2.12: ILA output when there is current

- Another way to verify is to collect data from ILA over a time in csv and then plot but problem with this method is it looses data samples between two instances of ILA as it take some time to run ILA and then download the data. This can be seen in plot below.

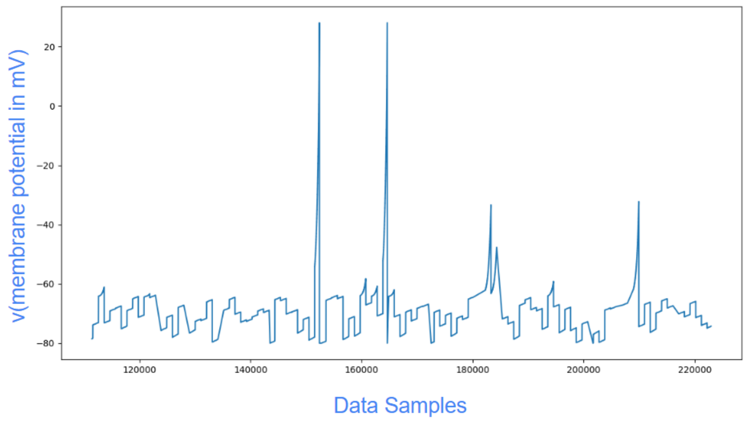


Figure 2.13: ILA data collected over time and plotted

# References

[1] Dan F. M. Goodman and Marcus Ghosh. *Neuroscience for Machine Learners.* https://doi.org/10.5281/zenodo.10366802, 2023.

[2] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003.

[3] Junran Pu, Wang Ling Goh, Vishnu P. Nambiar, Yi Sheng Chong, and Anh Tuan Do. A low-cost high-throughput digital design of biorealistic spiking neuron. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(4):1398–1402, 2021.