Chapter 3

Design and Implementation in FPGA

3.1 Overview

The design and implementation of SNN in VHDL is a modified version of the implementation by Eduard-Guillem Merino Mallorquí including its references, my designs are a modified version of the reference. In this section, the designs will be compared to the reference and some discussion will be made about matching the HW simulation with the software one.

Generally speaking, the software simulation used more precise implementation of the Izhikevich neuron and the STDP synapse. since float/fixed-point operations are computationally expensive it is preferred to use binary operations for HW implementation. I will argue why it is not possible to compare the software simulations with the HW due to the simplifications and approximations made in the HW simulation.

3.2 Network Design

For the most simple form of SNN implementation, there are three components, an AER(Address-Event Representation), the Izhikevich Neuron, and its RAM. For now, the STDP implementation is ignored since it is possible to run a simulation using only static synapses.

The AER takes a spikes signal which is a vector of the same length as the number of neurons in the network, in the default state there are no spikes, but when there is a spike detected in the vector, the AER broadcasts the spike's index number in the AER_Bus signal.

In case more than one neuron fires a spike at the same time the AER disables the neuron's functions until it finishes broadcasting the addresses of all the neurons that fired.

An important addition to the design is the DT signal which is a signal that turns on every $0.5ms$ for one clock cycle. DT is the time interval at which the SNN receives a signal from the spike encoder from outside.

3.2.1 AER

The AER is responsible for managing spikes and turning on the functions of the neurons, it receives 3 signals and outputs 3 signals Fig.(3.1). The default state for EN_neuron signal is ON, and for AER signal is $int(-1)$.

- CLK - [bit] clock signal.

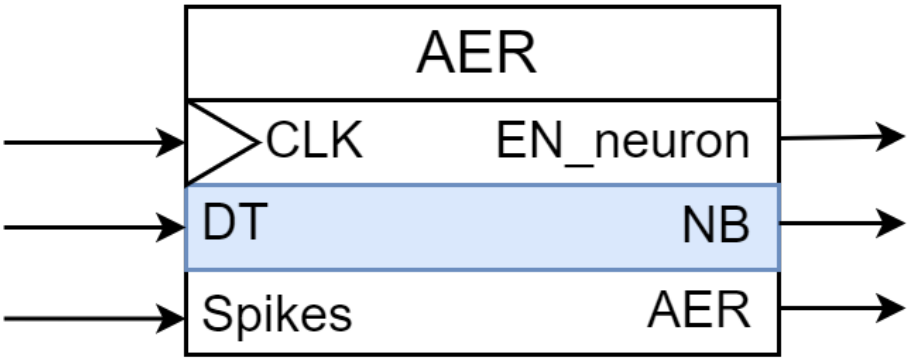

FIGURE 3.1: Block Diagram of AER.

- DT - [bit] simulation resolution signal, *dt* of the differential equation (3.1).
- Spikes - [bit vector] contains as many bits as neurons form the SNN.
- EN_neuron - [bit] always on, when off the neuron stops its functions.
- NB - [bit] only ON when the AER is broadcasting an address.
- AER - [int] defaults to -1, communication bus with neurons.

First, the module waits for the DT signal to be activated. Then it reads the Spikes signal. The Spikes signal is a vector of bits with length = number of neurons in the SNN. each bit represents a neuron's Spike Out at the time of DT ON signal.

Meaning When the DT signal is ON and the Spikes signal has a 1 bit at index 30, it means that the 30th neuron has spiked.

When neuron 30 spikes, then during the clock cycle after the DT signal was activated, the AER module will broadcast 30 in the AER signal for one clock cycle. The NB signal will be active for one clock cycle as well.

If more than one neuron spiked eg. 30 and 35. The NB signal will be turned on for two clock cycles. During the first cycle, the AER signal broadcasts 30 and the EN_neuron signal will remain active. In the second clock cycle, the AER broadcasts 35 and the EN_neuron will be deactivated. Read more about the EN_neuron signal in section (3.2.2). the same rule applies to all present neurons

In case there are no spikes when DT is ON, the NB signal will be ON for one cycle. Read more about the DT signal in section (3.2.2)

In the reference design of the AER, There is a memory buffer to store the state of Spikes in each cycle in case there are neurons still firing while the AER signal is still broadcasting, after the introduction of the DT, this segment is completely ignored as it is no longer of use.

3.2.2 Izhikevich Neuron

Adaptaion

In order to reduce the complexity of the functions, divisions were approximated to use binary shifts, and since the lack of float points, the equation was multiplied by

10. The Izhikevich functions in eq. (3.1) and eq.(3.2) and the reset condition (3.3). Were simplified into eq.(3.4), eq.(3.5) and (3.6).

$$dv/dt = 0.04v^2 + 5v + 140 - u + I \tag{3.1}$$

$$du/dt = 0.02(0.2v - u) \tag{3.2}$$

$$\begin{aligned} &\text{if} \quad v \geq 30mV: \\ &\quad v \text{ is set to } -65 \\ &\quad u \text{ is incremented by } 8 \end{aligned} \tag{3.3}$$

$$v[n+1] = 2v[n] + \frac{1}{2^8}v^2[n] + 1400 - u[n] + I[n] \tag{3.4}$$

$$u[n+1] = u[n] + \frac{1}{2^6}(\frac{1}{2^2}v[n] - u[n]) \tag{3.5}$$

$$\begin{aligned} &\text{if} \quad v[n+1] \geq 300mV: \\ &\quad v[n+1] = -650 \\ &\quad u[n+1] = u[n+1] + 80 \end{aligned} \tag{3.6}$$

In the reference design, a registry store was used for I, v, and u Fig.(3.2), it is easy to realize the functions were delayed by $n = 3$. Since this is a differential equation the function should be multiplied by dt which was ignored in this implementation. Finally, since one of the functions is polynomial and in addition to all the simplifications, even if we match the timing of the first spike with the one from the software simulation assuming both have the same incoming spike train, it is not possible to match it at any time forward, the behavior of this implementation should not be expected to match the software simulation implementation for any network.

In my redesign of the Neuron, I removed the stores removing the delay of $n = 3$ and used fixed-point representation for the neuron's functions. Even then I was able to match the timing of the neuron spiking after using 16 bits for the fraction part of the equation Fig.(3.3).

Design and architecture

The design module of the Neuron has 9 entries Fig.(3.4), 2 of which i added to the reference design. The neuron has a RAM attached to it Fig.(3.5).

- CLK - the clock signal.
- EN - activation signal turns off the execution of the functions.
- WE - write enable signal, passed on to the RAM.
- Addr - address, passed on to the RAM.
- Weight - synaptic weight, passed on to the RAM.
- AER_BUS - number of the neuron that spiked in int, passed on to the RAM.
- NB - neuron broadcasting signal, enables the neuron to receive synaptic weights from the RAM.

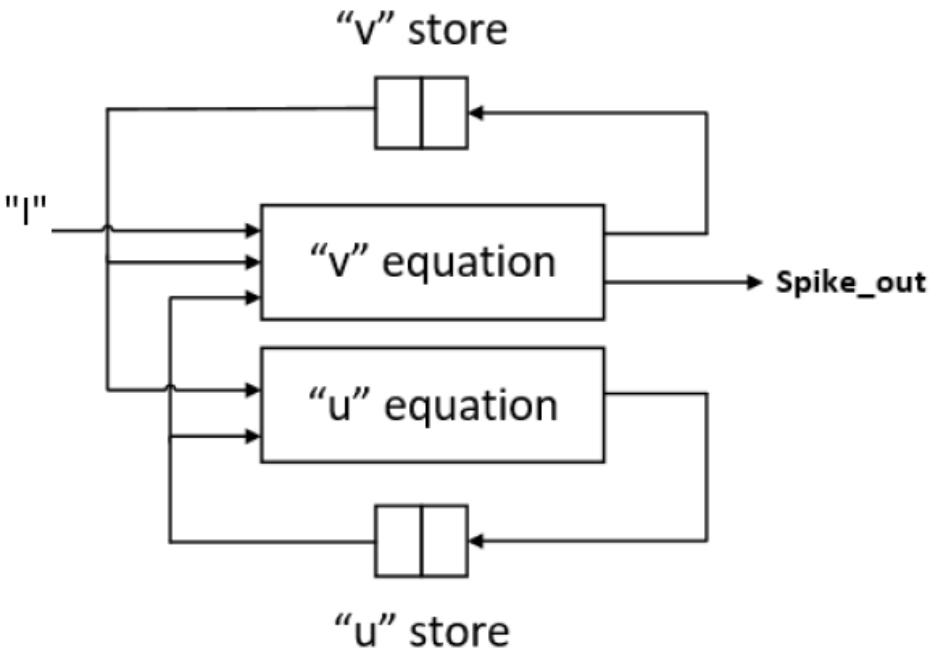

FIGURE 3.2: Neuron Function stores.

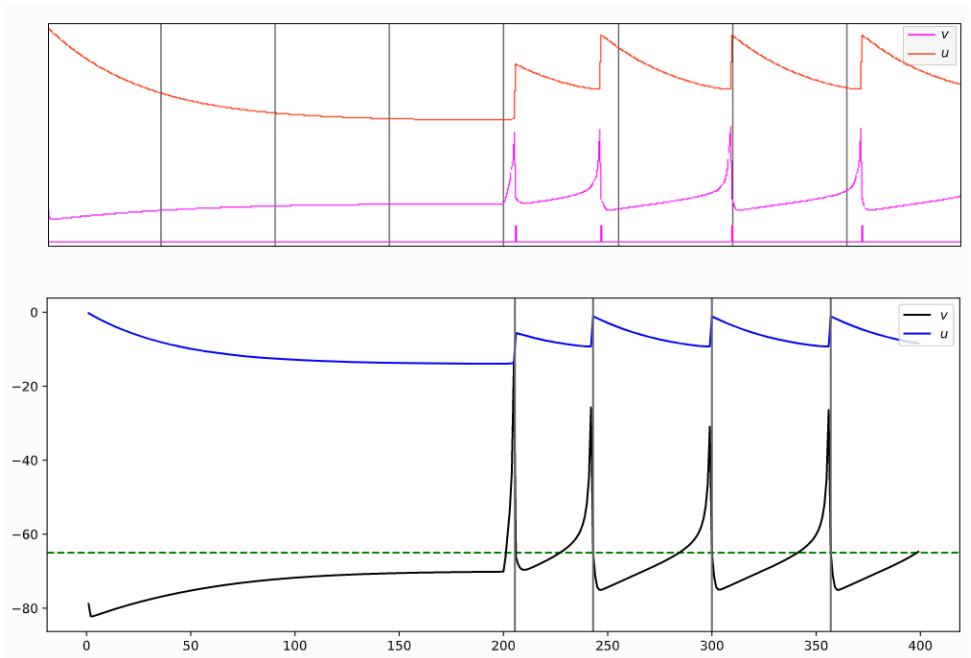

FIGURE 3.3: The membrane potential and membrane recovery of the VHDL Neuron, UP. and the Software simulated neuron, Down. the x axis is the time in *ms*.

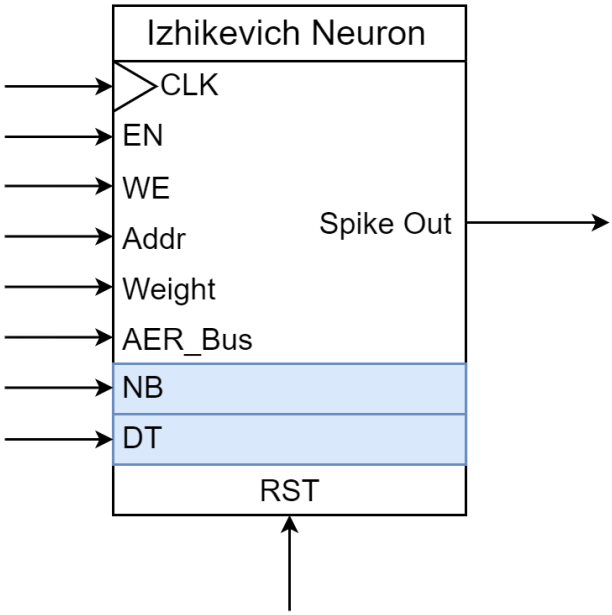

FIGURE 3.4: Block Diagram of the Izhikevivh neuron.

- DT - on for one CLK cycle every dt. In simulations $dt = 0.5ms$ but it is scaled down in VHDL simulations.
- RST - reset signal.
- Spike Out - [bit] connected to "Spikes[n]" signal where n is the neuron's own number.

Signals related to STDP learning are ignored for static SNN which are the WE, Addr, Weight signals, they are explained later in the STDP section. The RAM's only function for a static synapse is to receive a neuron number as int in the DPRA input and return its corresponding weight to the neuron using the DPO signal. In order for the Neuron to start receiving spikes and execute its functions the NB signal should be activated. Only then the neuron module is ready to read the address communicated by the AER module in the AER_BUS input of the neuron Fig.(3.6).

If neuron 30 spiked, the AER will activate the NB signal, the EN_neuron is already active as default and AER_Bus will be 30, The RAM receives the AER_Bus and returns the value stored in its memory at that address to the Neuron's input I.

At the falling edge of NB, The Neuron will execute its differential functions using the input value I. If the functions trigger a spike. the neuron will broadcast a 1 signal to the Spike_out signal for one cycle at the rising edge of the DT signal. The Spike_out signal is connected to the Spikes signal at index n, where n is the neuron's number. The Spikes Signal will be read by the AER module as explained before.

If more than one neuron spike at the same time, the Neuron will receive the first address as normal, but during the consecutive address broadcasts, the EN signal will be deactivated, in this case, the Neuron will continue accumulating values received from the RAM in I on top of the first value received during the first clock cycle of the broadcast. When the AER has finished broadcasting addresses the neuron will behave normally like in the previous case at the falling edge of NB.

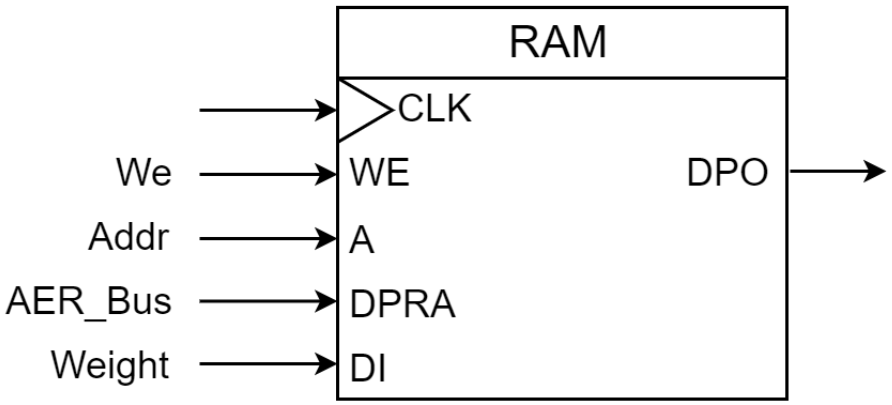

FIGURE 3.5: Block Diagram of the neuron's RAM.

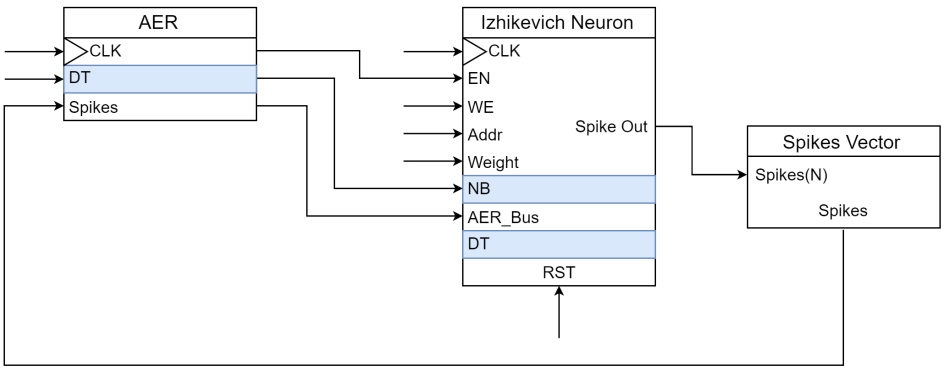

FIGURE 3.6: Block Diagram of AER and Neuron behavior.

In case no neuron spiked, the AER will activate NB for one cycle and the AER_Bus will be -1. In this case, the Neuron will still execute its functions with input $I = 0$.

In case there is no connection between the firing neuron and the neuron receiving the Spike, the weight in the memory's RAM will be zero as it will have no effect on I.

3.2.3 STDP Synapse

In the reference design, the STDP learning rule eq.(3.8), which is a different STDP model [4], the model used in the software simulations.

Considering a pre-synaptic neuron i and a post-synaptic neuron j, the STDP rule characterizes the changes in synaptic strength in eq.(3.7).

$$\Delta w_j = \sum_{k=1}^{N} \sum_{l=1}^{N} W(t_j^l - t_i^k) \tag{3.7}$$

$$\Delta w = \begin{cases} A_+ \times exp(-|\Delta t|\tau) & if \Delta t < 0 \\ -A_- \times exp(-|\Delta t|\tau) & if \Delta t \geq 0 \end{cases} \tag{3.8}$$

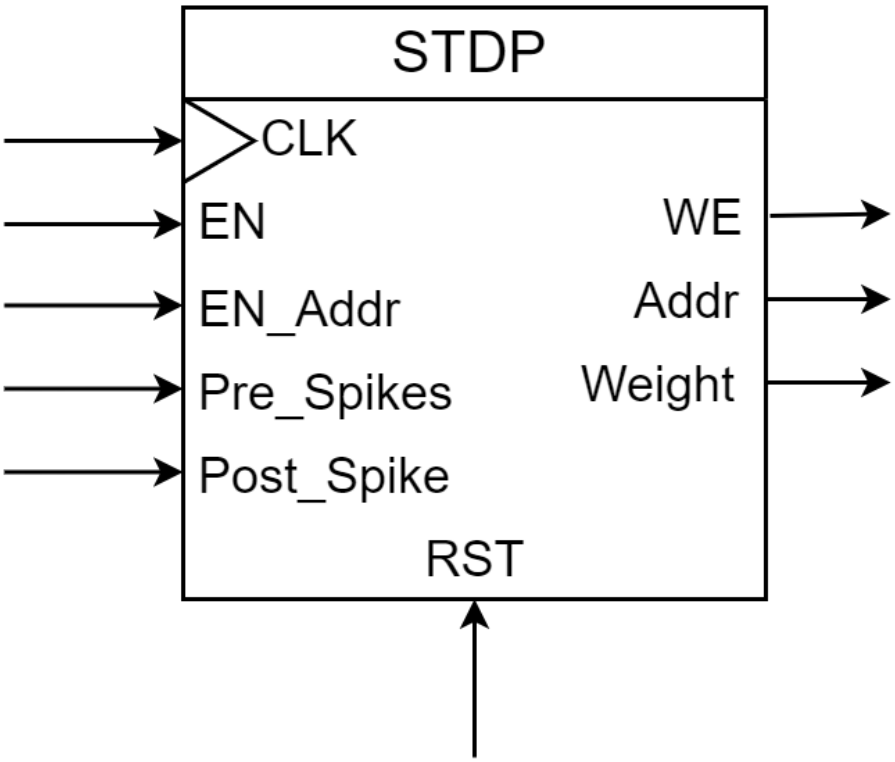

FIGURE 3.7: Block Diagram of STDP.

To reduce the complexity of handling all the neurons connections a solution was implemented, creating a single training module to handle all connections of a single neuron Fig.(3.7), First I will describe the output signals of the STDP module.

- write enable (WE) - The RAM reads it and understands it needs to update the weight of a connection.
- address (Addr) - The address of the connection to the pre-synaptic neuron that needs updating.
- synaptic weight (Weight) - The new weight that needs to be inserted.

Now to understand how the STDP module applies the learning I will describe the Input signals and discuss why the module needs to be changed.

- EN - neuron activation signal.
- EN_Addr - when activated the STDP module will change the address of the connection in which it is applying the learning.
- Pre_Spikes - the spikes vector.
- Post_Spike - the spike of neuron it is attached to.

The behavior of the STDP synapse is that it will only apply the learning on a single connection with the neuron n until it gets a signal from EN_Addr to switch

to the next connection with the neuron $n + 1$, The STDP module was designed for a Feedforward network consisting of only two layers. the Pre_Spikes vector-only containing the neuron spikes of the first layer then only training the connections between the first and second layer. The STDP module will treat the connection rule between the layers as an "All to All" rule.

In case this module is used for an LSM network the Pre_Spikes vector will represent all neurons in the network. The STDP module is not designed to take into consideration how the network is connected, for its attached neuron it will apply the STDP learning on the connections with all neurons in the network ie: changing the values stored in memory for non-connected neurons from zero to the updated weight based on the learning rule. as explained in (3.2.2) when there is no connection between the firing neuron and the neuron receiving the spike the synapse weight will be zero in the RAM's memory at that neuron's address.