FPGA realization of SNN Model for Opportunistic Spectrum Access

Md Mehedi Hassan Galib

***Abstract*—In this project, we implement biologically plausible spiking neural network on field programmable gate array (FPGA) platform addressing model parameters and neural dynamics. FPGAs are an attractive choice for SNN application since they offer a programmable substrate for acceleration, however, obtaining both performance and energy efficiency with FPGAs is a laborious task even for expert hardware designers. Hence, we use high level synthesis compiler (Vitis) to convert our well-trained software model to register transfer level (RTL) and implement that RTL model into the target FPGA (Virtex-7).**

***Index Terms*—Spike neural network, FPGA implementation, Opportunistic access, area, power, performance.**

I. Introduction

Non-Von Neumann architectures have gained popularity in recent years to meet the requirement for high throughput through parallel processing of information located in distributed memories [1]. In traditional neural networks, despite of implementation in a variety of applications [2], [3], requires a tremendous amount of computational resources owing to intense matrix multiplications and memory access. In contrast, spiking neural networks (SNNs) emulate and simulate the structure of nervous systems, which sends information in the form of action potentials (or spikes). Information transmission in the network is often triggered by events that occur irregularly in time [4]. Consequently, because to the event-driven aspect, SNNs perform energy-efficient computation and have shown great promise for real-time applications. SNNs have been investigated for a variety of applications, including image classification, object recognition, navigation, and motor control, because of their functional similarities to the human nervous system.

Different supervised and unsupervised learning algorithms can be used to train SNNs. However, SNNs can be directly trained with biologically plausible unsupervised learning principles, such as spike-timing-dependent plasticity (STDP), which updates synaptic weights using local spiking data between a presynaptic neuron and postsynaptic neuron. Additionally, the STDP learning rule has been shown to be quite helpful in resolving challenging computational issues and is capable of analyzing spatio-temporal data. Neuroscientists are very interested in STDP because it offers a biologically plausible explanation for the learning process taking place in human brains. Additionally, the STDP learning rule may be successfully implemented on hardware and allows for scalable online learning in a variety of neuromorphic systems . A voltage-based STDP model is taken into consideration in this project due to its biological plausibility and simplicity.

SNNs need to be built as a highly parallel hardware system in order to mimic the structure of a human brain, which enable them to be able to execute computations in the time domain. There have been several proposals for hardware systems that make use of analog as well as digital circuits in order to accomplish this goal. Analog systems suffer from a lack of flexibility, noise, device variabilities, and space-efficient storage components [35], [36], despite the fact that they only need a tiny amount of space and use a relatively small amount of power. Digital systems, on the other hand, provide remarkable reconfigurability, stability, and effective embedded memory. Specifically, the field-programmable gate array (FPGA) offers a low-cost reconfigurable platform that supports the design of parallel processing architectures. Because of this, it is an excellent choice for the functional and topological exploration of large-scale SNNs because it is so well suited for the task.

Hence, in this project, I have implemented SNN model through high level synthesis (HLS) compiler, which give estimation of required resources, power consumption, and computation gain compared to software implementation as well as conventional ML approach.

II. Background

A. Spike Neural Network

SNN is a third generation neural network employing spiking neurons. In comparison to traditional neural networks, it provides a more accurate representation of the human brain. Spiking neural networks contain membrane potential, and their neurons utilize spikes to convey or receive information across layers of the network. Moreover, synaptic weights are computed by SNN in an event-driven sequence due to the membrane potential. Because of this property, SNN is superior than conventional neural networks in terms of its energy efficiency. This event-driven facility makes SNN feasible for edge devices. In SNN, when bio-inspired neurons take in new information in the form of spikes, their membrane potentials go up. When the membrane potential of a neuron hits a certain threshold, the neuron will fire, and it will release a spike. After then, its membrane potential is reset to its resting voltage during a refractory period. The membrane potential of the neuron does not increase during the refractory period, despite the fact that the neuron may receive spikes during this time. After the refractory period has passed, the membrane potential will begin to increase again whenever it is subjected to spikes. The threshold is a precise value that determines whether a neuron will fire or not, and it adjusts itself in an adaptive manner based on how intense the input stimulus is.

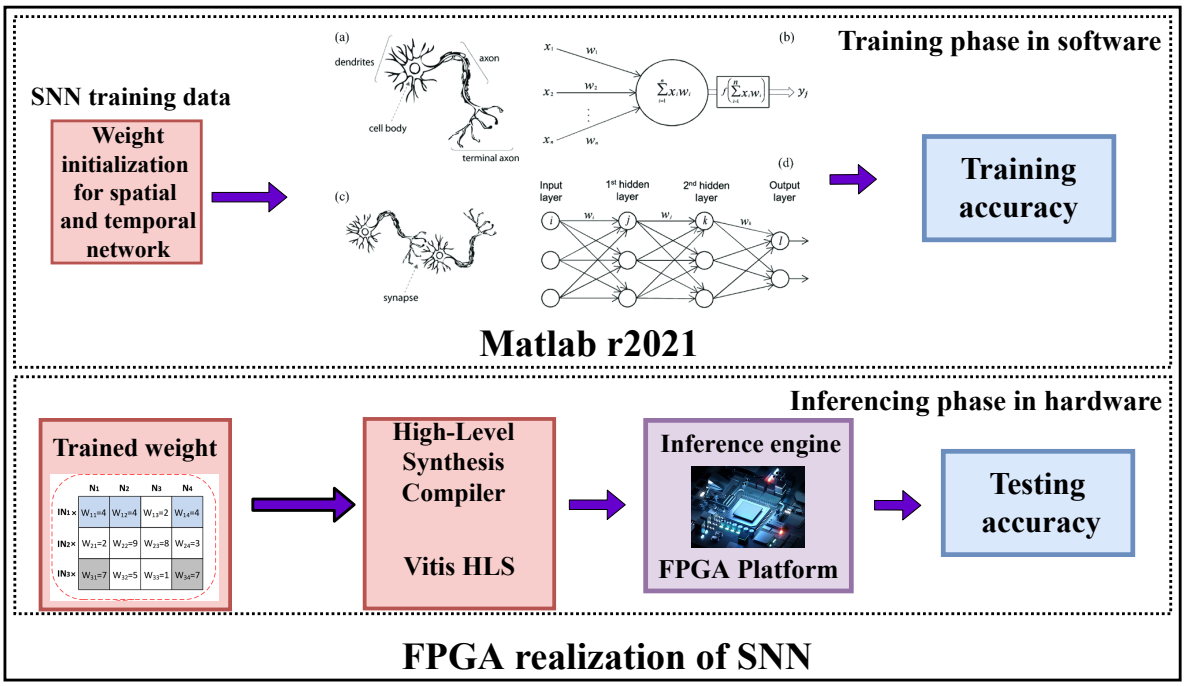

Fig. 1: ST-SNN system deployment for opportunistic spectrum access in CRN

The potential that neurons have before they receive spikes from other neurons is known as their resting potential. A neuron is said to be in a refractory phase when it does not raise its membrane potential, despite the fact that it may be receiving spikes throughout this time. Like biological neurons, neurons in SNN are connected to one another through synapse. These weights, known as synaptic weights, are assigned to each synaptic connection. When synaptic weights increase, the strength of the synaptic connections that exist between neurons increases as well. On the other hand, as synaptic weights decrease, the strength of the connections between neurons decreases. Synaptic weights are only updated in SNN when the underlying neurons activate. Because of this, the synaptic weights of the SNN are not necessarily modified every time it receives new input data. SNN is event-driven because of the synaptic update sequence that it uses. In terms of biology, neurons that are firing produce spikes that are then used to communicate with other neurons through synapses. The membrane potentials of the neurons in the layer below are increased as a result of these spikes. Many other neuron models, such as the Integrate and Fire model, the Leaky Integrate and Fire model, the Spike Response Model, the Hodgkin-Huxley model, and the Izhikevich model, have been presented in order to remember this process [5], [6], [7], [8]. Among them, the Leaky Integrate and Fire (LIF) neuron is the model that is being used in SNN the most frequently at the moment.

B. Spike time dependent plasticity

A biological process known as STDP is responsible for regulating synaptic connections [8]. It does this by analyzing the temporal fore-and-aft associations between spikes produced by pre-synaptic neurons and spikes produced by post-synaptic neurons. This allows it to either strengthen or decrease synaptic connections. Because of this, using STDP with error backpropagation might be challenging. As a result of this limitation, researchers often use STDP for the purpose of unsupervised learning. Long Tern Potential (LTP) is the name given to the phenomenon that occurs when a pre-synaptic neuron activates before a post-synaptic neuron fires. This phenomenon serves to reinforce the synaptic connection between the neurons. Long Term Depression (LTD) is the name given to the phenomena that occurs when a post-synaptic neuron activates before a pre-synaptic neuron fires. This occurrence has the effect of weakening the synapse connection between neurons. This process is used as a significant learning rule in order to bring the synaptic weights of SNN up to date. Synaptic weights between neurons in an SNN grow when long-term potentiation (LTP) takes place; this increase in synaptic weights serves to reinforce the link between neurons. The frequency of firing in the post synaptic neuron of a connection increases as that connection is reinforced. In a similar manner, as LTD takes place, the synaptic weights between neurons decrease. This, in turn, causes the synaptic connection to become less robust. The frequency of firing in the post synaptic neuron of a connection decreases as that connection's synaptic link is weakened.

III. System Model & Approach Overview

The overall system model for developing an FPGA based inference engine using SNN architecture is depicted in Figure 1.

A. Data-set

In this project, we adopt spectrum usage prediction data-set from a commercial LTE network.

B. Software Implementation

1) Neuron Processing Core

a) Excitatory Neuron Model

For stimulating biological excitatory neuron in our temporal network, we adopt an adaptive exponential leaky integrate and fire neuron model, since it expresses time varying neuronal dynamics which is absent in a simple leaky integrate and fire neuron. In the dynamic model of adaptive exponential leaky integrate and fire neuron, we have one differential equation of membrane potential, one differential equation for adaptive sub-threshold voltage, exponential activation function, and time varying resistance. The first order ordinary differential equation for membrane potential V is modeled as

$$C_m \frac{dV}{dt} = \frac{(E-V)}{R_{leak}} + \sum I^{int} + \sum I_{syn} - L \tag{1}$$

Here C_m is neuron membrane capacitance, R_{leak} is neuron leakage resistance, E is neuron reversal potential, I^{int} is intrinsic cell current, which is defined as $I^{int} = \frac{\delta_e exp(V-V_{th})}{R_{Leak}}$, I_{syn} is summation of input current from presynaptic neuron to postsynaptic neuron, and L is excitatory neuron adaptability.

The first order differential equation for excitatory neuron adaptability is described as

$$\frac{dL}{dt} = \frac{\alpha(V-E) - L}{\tau_L} \tag{2}$$

Here, α is the sub-threshold adaptation, τ_L is the adaptive time constant. The neuron experiences a increase in sub-threshold by the factor β after each spikes. For our simulation purpose, we choose $\alpha = 4ns$ and $\beta = 1000pA$.

b) Inhibitory Neuron Model

Since the purpose of inhibitory neurons is to provide homeostatic plasticity to our temporal network, we adopt leaky integrate and fire neuron model (LIF). The advantage, however with simple LIF model, is to deal with a single ordinary differential equation unlike the excitatory neurons. The first order ordinary differential equation for the membrane potential (V) for an inhibitory neuron is described as follows.

$$C_m \frac{dV}{dt} = \frac{(E-V)}{R_{leak}} + \sum I_{syn} \tag{3}$$

2) Memristive Realization

The synaptic current (I_{syn}) of eq. (1) and eq. (3) are modeled using time varying resistance and potential difference between reversal and membrane potential. Since, in our temporal network, we resort to both excitatory and inhibitory neurons, the synapse dynamics is different for these neurons due to various permutation of presynaptic and postsynaptic connections. Hence, I_{syn} for excitatory and inhibitory neurons are expressed using eq. (4) and eq. (5) respectively.

$$I_{syn} = \frac{E_E - V}{R_{EE}(t)} + \frac{E_I - V}{R_{EI}(t)} \tag{4}$$

$$I_{syn} = \frac{E_E - V}{R_{IE}(t)} + \frac{E_I - V}{R_{II}(t)} \tag{5}$$

Here, E_E and E_I are excitatory and inhibitory neuron reversal potential respectively. Moreover, eq. (4) and eq. (5) contain four different time varying resistances such as R_{EE}, R_{EI}, R_{IE}, and R_{II}. For R_{EE} and R_{EI}, postsynaptic excitatory neuron is connected with presynaptic excitatory and inhibitory neuron respectively, whereas for R_{IE} and R_{II}, postsynaptic inhibitory neuron is connected with presynaptic excitatory and inhibitory neuron respectively. The dynamics for the time varying resistance is expressed as follows.

$$R_{XY}(t) = \frac{1}{\frac{e^{-\frac{t}{\tau_d^Y}} - e^{-\frac{t}{\tau_r^X}}}{\tau_d^Y - \tau_r^X} * \left(W_{ex} I_{ex} + \sum W_{XY}(t)) I(t) \right)} \tag{6}$$

Here, X represent the postsynaptic neuron and Y represent presynaptic neuron and possible set of XY is $\{EE, EI, IE, EE\}$. Moreover, I_{ex} and W_{ex} are constant biasing current and weight, W_{XY} is the weight between presynaptic Y neuron to postsynaptic X neuron, τ_d^Y is the decay time for synaptic dynamics of presynaptic Y neuron, and τ_r^X is the rise time for synaptic dynamics of postsynaptic X neuron.

3) Weight update unit

For replicating biological synapses in our temporal network, we incorporate STDP for all the synapses that are connected among excitatory neurons and HP for all the synapses that are connected between inhibitory to excitatory neurons.

a) STDP Unit

Among three different type of STDPs such as calcium ion based STDP, triplet STDP, and voltage based STDP, we adopt voltage based STDP to update synaptic weights between excitatory neurons. The advantage, however with voltage based STDP, is to ensure proper stabilization of all the synaptic weights compared with other STDPs. The dynamics of voltage based STDP composes of several first order differential equations such as a low-pass filtered version of presynaptic input, two sets of low-pass filtered version of post-synaptic voltage, and a weight update equation, which are expresses as

$$\frac{dx}{dt} = \frac{I-x}{\tau_x} \tag{7}$$

$$\frac{dy}{dt} = \frac{V-y}{\tau_y} \tag{8}$$

$$\frac{dz}{dt} = \frac{V-z}{\tau_z} \tag{9}$$

$$\frac{dW}{dt} = A_{LTP} x_j(t) F(V_i(t) - \Delta_{LTP}) F(y_i(t) - \Delta_{LTD}) - A_{LTD} I(t) F(z_i(t) - \Delta_{LTD}) \tag{10}$$

Here, x is low-pass filtered version of input, y and z are low-pass filtered version of postsynaptic voltage, A_{LTP} and A_{LTD} are amplitude and Δ_{LTP} and Δ_{LTD} are threshold of

IP Block	Resource utilization						Frequency	On chip power (w)	
	LUT	LUTRAM	FF	DSP	BRAM	BUFG	MHz	Dynamic	Static
Initial Network Parameter	*6913*	*134*	*4533*	*227*	*22*	*1*	*50.72*	*0.111*	*0.146*
Excitatory Neuron	*39*	*1*	*89*	*0*	*0*	*2*	*76.7*	*0.108*	*0.135*
Inhibitory Neuron	*38*	*1*	*86*	*0*	*0*	*2*	*78.07*	*0.109*	*0.138*

long-term potentiation and depression respectively, and $F(.)$ signifies the activation function.

b) WTA Unit

To update synapses between inhibitory and excitatory neuron and provide winner takes all architecture, a homeostatic plasticity model is considered with three first order differential equations such as a low-pass filtered version of presynaptic inhibitory input, a low-pass filtered version of postsynaptic voltage, and weight update dynamics, which are described as follows.

$$\frac{du}{dt} = \frac{I-u}{\tau} \tag{11}$$

$$\frac{dv}{dt} = \frac{V-v}{\tau} \tag{12}$$

$$\frac{dW}{dt} = A_{inh}(v(t) - 2r_0\tau)I(t) + A_{inh}u(t)V(t) \tag{13}$$

4) Poisson Spike Generator

To encode the spectrum sequence as stochastic spike trains, the Poisson spike generator is employed. Spiking neurons in the subsequent layer receive the generated spike trains. In SNNs, this encoding method is known as rate coding . The 16-bit linear-feedback shift register is used to construct a random number generator (LFSR). Each time step within a given interval generates a random number, which is then compared to the input pixel value. It returns a logic-1 if the random number is less than 1. Whenever this is not the case, a logical 0 is produced. The Poisson spike train is built from such a 0 or 1 time series.

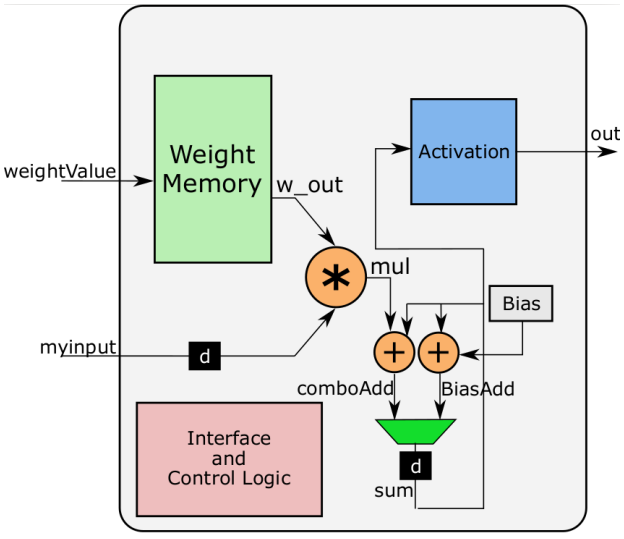

Fig. 2: FPGA realization of a neuron using weight memory, multiplier, adder, and activation function.

C. FPGA realization

As we are focusing FPGA implementation, the methodology of implementing a SNN is totally different from the software implementation as shown in Fig. 2. In case of FPGA interference, a microprocessor is used to move all of the input from the personal computer to the external memory on FPGA. The direct memory access (DMA) protocol and the external memory are both under the control of the microprocessor, which manages their communication. As a result, input data are moved from the external memory to the network using DMA and are then turned into Poisson spike trains. The read operations of synaptic weights from BRAMs are controlled by these spike trains, which means that only specific weights are transmitted to the neuron processing centres. The membrane potentials of the neuron processing centers are then checked, and spikes are produced if the values of those potentials are higher than the thresholds. The SNN's generated spikes first activate the STDP learning unit, after which they are transferred to the DDR memory, and finally they are extracted on the PC. The Poisson spikes generator, neuron units, the STDP learning unit, and memory blocks make up the bulk of the SNN. Other components include memory blocks.

IV. Result and Discussion

This section compares the area, power, performance, and accuracy of the software implementation of SNN with our FPGA realization of inference engine. To perform this analysis, we adopt following experimental setup.

A. Experimental setup and data set

As aforementioned, we adopt commercial LTE data which contains dynamics resource block allocations and well as white spaces. Our main motivation is to detect the white spectrum and use for the opportunistic access. Hence, for software training purpose, we chose Matlab@r2021 which run in AMD® Ryzen 7 4210u CPU. For FPGA implementation, we perform HLS simulation using Xilinx® Vitis for targeted hardware Virtex-7. In HLS, we adopt system C with static memory allocation.

B. Software Training

After performing SNN model training in Matlab, we copied the saturated weight for the hardware implementation as shown in Fig. 1. In order to perform software training, we perform simulation over entire data-set for 85 epochs to confirm the removal of all unwanted dynamics.

C. FPGA resource utilization and constraints

We estimate the FPGA resource utilization with maximum clock frequency and total on chip power for all the different combinations of our SNN IP blocks generated by

HLS. Moreover, in order to find the trade-off between power, performance, area, and accuracy, we calculate optimal resource utilization with higher clock speed with sacrificing a bit of on chip power. The resource are calculated in terms of e number of LUTs, number of FF, number of block RAM and DSP block uses as illustrated in Table I.

D. Limitation

Due to the time limit, not all the functions have been implemented using HLS compiler, which are considered as future works.

V. Conclusion

In this project, we build FPGA inference engine based on the weights trained by the software implemented of SNN. Then, we compare the performance and power between FPGA inference engine for different types of generated IP block by HLS.

Acknowledgment

The authors would like to thank Dr. Mohamed Younis, course instructor of CMPE 684 (Wireless Sensor Network), Department of Computer Science & Electrical Engineering, University of Maryland, Baltimore, County, MD-21250, USA.

References

[1] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," 2014.

[2] V. P. Rekkas, S. Sotiroudis, P. Sarigiannidis, S. Wan, G. K. Karagiannidis, and S. K. Goudos, "Machine learning in beyond 5g/6g networks—state-of-the-art and future trends," *Electronics*, vol. 10, no. 22, p. 2786, 2021.

[3] F. Obite, A. D. Usman, and E. Okafor, "An overview of deep reinforcement learning for spectrum sensing in cognitive radio networks," *Digital Signal Processing*, vol. 113, p. 103014, 2021.

[4] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and stdp integrated circuits," *IEEE transactions on biomedical circuits and systems*, vol. 6, no. 3, pp. 246–256, 2012.

[5] K. Xie, Z. Zhang, B. Li, J. Kang, D. Niyato, S. Xie, and Y. Wu, "Efficient federated learning with spike neural networks for traffic sign recognition," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9980–9992, 2022.

[6] A. Destexhe, M. Rudolph, and D. Paré, "The high-conductance state of neocortical neurons in vivo," *Nature reviews neuroscience*, vol. 4, no. 9, pp. 739–751, 2003.

[7] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[8] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.