Posit-based Spiking Neuron in an FPGA

Victor H. L. Silva[*], Jeferson F. Chaves[*], Rogério M. Gomes[*], Bruno A. Santos[*]
[*]Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG

***Abstract*—In this work, we designed a novel spiking neuron circuit in an FPGA. Our low latency circuit is based on posit arithmetic, an emerging format intended to succeed IEEE 754. By proposing this hardware implementation using a new format for floating point representation, we managed to balance accuracy and the number of resources in circuitry. Furthermore, we successfully validated our design reproducing neuron's firing patterns while reducing memory needs by 25% in contrast with IEEE 754 based solutions. To the best of our knowledge, this is the first design of this kind. We believe that this novel design could offer new ground for computational neuroscience experiments.**

***Index Terms*—Posit arithmetic, Izhikevich neuron model, FPGA**

I. Introduction

Computational Neuroscience has been yielding essential insights into how the brain works. Models typically based on neural networks are built to understand how higher functions arise from interacting neurons [1]. A significant challenge in the field is ensuring reproducibility in simulating complex neural networks [2]–[4]. This situation led to recent efforts by the research community towards proposing a rigorous methodology of increasing the correctness of simulation results in the absence of biological experimental validation data [2]–[6]. As Pauli et al. pointed out [7], this issue goes more profound than the model specification itself. For example, the choice of compiler, the order in which numerical operations are executed, or the underlying hardware running the model can lead to rounding errors. Hence, these options affect numerical accuracy, which is critically important as even slight deviations in the dynamics of individual neurons are expressed in the dynamics at the network level [5]–[7].

Typically, there is a trade-off in simulations between accuracy, network size, and simulation time. When accuracy is more critical, simulations are commonly done in software with the IEEE 754 floating point numerical format, which restricts network size and/or simulation time. However, if network size or simulation time is the primary concern, the solution is typically a hardware implementation with a fixed point numerical format, which compromises accuracy.

To address those issues, i.e., to circumvent the trade-off above, we propose a hardware design of a biologically plausible neuron model using a new floating point format. By designing a hardware solution, we could explore parallelism favoring network size and/or simulation time. Additionally, using the Posit format as an IEEE 754 replacement for floating point representation, we could maintain accuracy while having a hardware friendly numerical pattern [8]. Finally, we choose to use the Izhikevich neuron model [9], [10] because it is the mainstream choice for simulating large-scale spiking neuron networks.

To the best of our knowledge, this is the first design of this kind, i.e., our effort is the only one to combine the most computational efficient neuron model and the promising new numerical format in an FPGA. Thus, we believe that this novel design could offer new ground for computational neuroscience experiments. It is also worth noting that previous implementations typically use fixed-point representations and implement different neural networks. However, as this work proposes to use floating-point representation to simulate a neuron model, the comparison with other works may not be fair.

The remaining of this paper is organized as follows. In Section II, we describe the Izhikevich neuron model and the posit numerical format. Section III presents our proposed design, its design process, and its validation. Finally, Section IV summarizes and concludes the work.

II. Background

In this section, we present a summary of the Posit floating point format [8] and a brief introduction to the Izhikevich neuron model [9].

A. Posit Arithmetic

Posit arithmetic or type-3 universal number (unum) is a new format intended to replace IEEE 754 standard for representing real numbers [8]. Posit numbers were designed to provide a better dynamic range and more accuracy than IEEE standard over the same bit field.

A posit format is defined as the pair (N, ES) composed by the word size (N) and exponent size (ES), which actually means the maximum exponent size. The Posit pattern has the format as shown in Figure 1. Although the Posit format has four components within the representation, e.g., Sign, Regime, Exponent, and Mantissa, their length could vary at run-time. The Sign bit, s, operates regularly as in conventional floats, i.e., 0 for a positive number, 1 for negative numbers.

Regime bits are a sequence of either all 0 or 1, terminated by an opposite bit. These bits and the exponent bits perform the

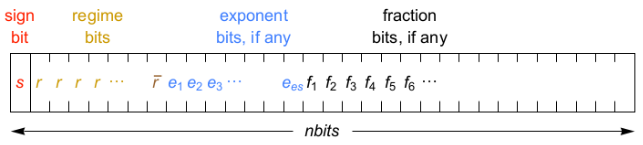
sign bit
regime bits
exponent bits, if any
fraction bits, if any
s r r r r $\cdots$ $\bar{r}$ e_1 e_2 e_3 $\cdots$ e_{es} f_1 f_2 f_3 f_4 f_5 f_6 $\cdots$
nbits

Fig. 1: Posit pattern

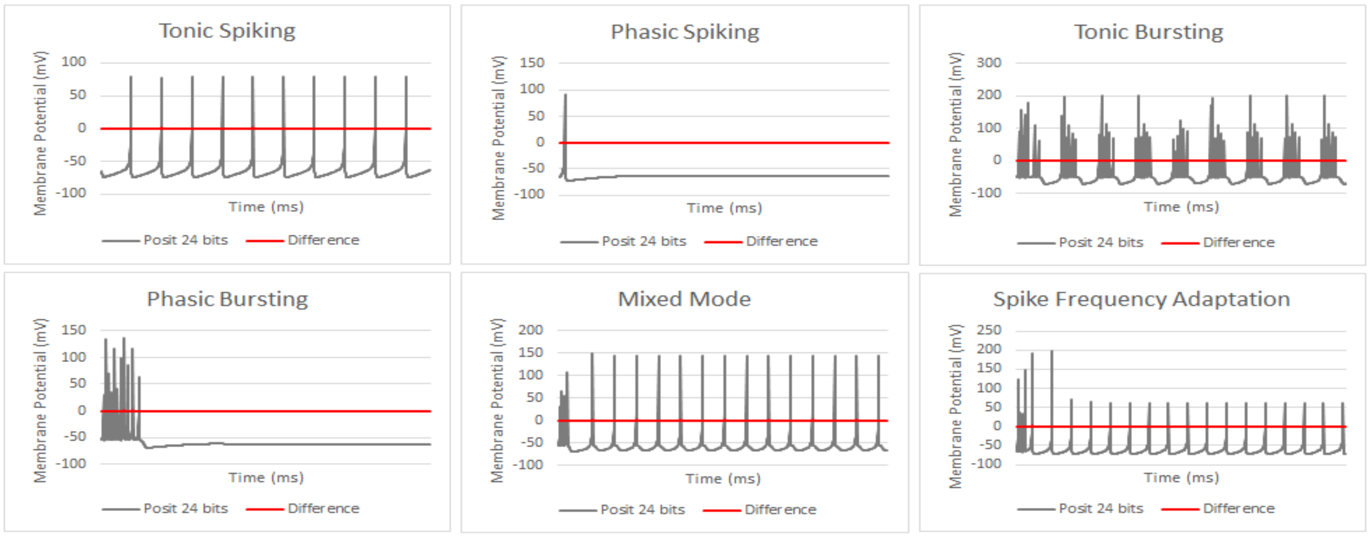

Fig. 2: Firing patterns generated by posit-based neuron circuit (gray curve). The red curve shows the difference from 32-bit IEEE 754 implementation. Simulation for a time interval of 1 second.

same role as the exponent bits in a standard float. However, due to its variable size, it manages to cover a relatively large range of values. This feature allows Posit to have a good precision close to one since few regime bits are used. On the other hand, if the number is extremely large or extremely close to zero many bits for the regime are used.

Finally, the Mantissa Bits, if available, are similar to the normalized floating point. The value of a posit number is given by the expression 1, where k represents a numerical meaning for the regime bits [8]:

$$s * (2^{(2^{ES})})^k * 2^e * f \tag{1}$$

B. *Izhikevich Neuron Model*

The neuron model proposed by Izhikevich is a two-dimension simplification of the Hodgkin-Huxley model and can be described by equations 2 and 3 with an auxiliary after-spike resetting represented by Equation 4:

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

$$\frac{du}{dt} = a(bv - u), \tag{3}$$

$$\text{if } v \geq 30\text{mV, then} \quad \begin{cases} v \leftarrow c \\ u \leftarrow u + d, \end{cases} \tag{4}$$

where v is the membrane potential of a neuron; I is the input current, u is the membrane recovery variable, a is the decay rate of u, b is the sensitivity of u for sub-threshold fluctuations of the membrane potential, c is the reset value of the membrane potential after spike and d is the reset value of the variable u after spike.

Izhikevich developed a model of spiking neuron that reproduces the behavior of real neurons very accurately, but with much simpler equations and low computational cost, allowing the simulation of networks with large numbers of neurons. Izhikevich also showed in [10] the biological plausibility and computational efficiency of his model and presented that different types of cortical neurons can be obtained by tuning the parameters a, b, c, and d of the model. In this work, we approached some types of neurons modeled by Izhikevich, whose behaviors can be seen in Figure 2.

III. Methodology

In this section, we show our proposed neuron circuit developed in three steps. First, we made a software implementation of the Izhikevich neuron using the posit numeric format. Our goal with this step was to verify the feasibility of reproducing a neuron's behavior with this new numerical format. Next, we designed and validated the neuron circuit by comparing its simulation results with the software results. Finally, we synthesized the circuit for an FPGA to evaluate the required resources.

Regarding the first step, i.e., the proof of concept of the Izhikevich model using posit numbers, we based on the posit arithmetic functions from the SoftPosit library [11], e.g., additions and multiplications, to implement our neuron model in C Language. To solve the model's differential equation, we use Euler's method iterating with a 1 ms time step, i.e., in each iteration, our algorithm simulates 1 ms where new values for v and u are produced from the previous values of these variables. The simulation results reveal that it is possible to reproduce neuron's firing patterns using the posit format.

Figure 2 shows that our design was able to qualitatively reproduce the behavior of real neurons using fewer bits (N=24, ES=2) comparing to IEEE 754. Concerning accuracy, the difference between the posit neuron and the IEEE 754 neuron (32 bits) is negligible, as shown in the red line in Figure 2. We also notice that it is possible to reduce even more the data

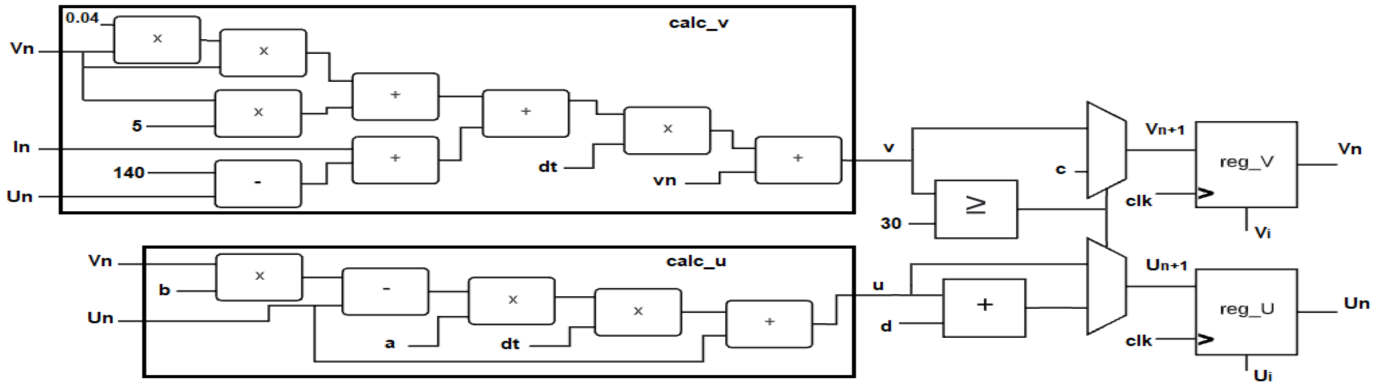

Fig. 3: Proposed circuit

size, i.e., the number of bits in the posit number, while still reproducing the same qualitative behavior as shown in Table I.

In the next step, we designed the digital circuit. For that, we used some posit arithmetic circuits (adders and multipliers) from the PACoGen core Generator [12] to implement our Verilog HDL design. This tool is a hardware generator for Posit, also written in Verilog HDL, which produces blocks that can be parameterized by the number of bits (N) and the size of the exponent bit (ES).

The diagram in Fig. 3 shows our proposed neuron. The circuit primary inputs are the initials values for neuron's state (v and u), the external current (I), and the clock signal, which was used only to register the neuron's state for the next iteration. Everything else in the neuron is pure combinational. Additionally, the design is as parallel as possible with no reuse of any arithmetic block.

To validate our proposed circuit, we carried out a simulation of the circuit with the ModelSim tool. In the testbench, we contrasted the values of v, produced by the circuit, against the software results (our neuron implemented in C Language with SoftPosit library). We noted an unexpected divergence in the results related to the timing of the spikes of the neuron (variable v). After a detailed review of our Verilog and C codes, we noted that the divergence was coming from differences in how SoftPosit and PACoGen round numbers, despite parameterizing both to use the same number of bits.

The cause of this problem is that SoftPosit uses the computer's 64-bit registers to store N-bit data. When performing an arithmetic operation, the SoftPosit functions take advantage of the 64-N extra bits to improve accuracy. On the other hand, the PACoGen blocks are generated with exactly N-bits internal signals. To fix this issue, we chose to modify SoftPosit to use only N-bit data as our goal is to produce smaller circuits.

TABLE I: Minimum number of bits needed to represent each simulated neuron type

Neuron	Number of Bits (Posit)
Tonic Spiking	18
Tonic Bursting	24
Phasic Spiking	18
Phasic Bursting	24
Mixed Mode	24
Spike Frequency Adaptation	24

TABLE II: Resources needed to implement the Izhikevich neuron model in a Stratix V 5SEEBF45I4 FPGA

Resources	Number of bits (Posit)		
	18	24	32
Logic utilization (in ALMs)	3,676 (1%)	4,834 (1%)	6,424 (2%)
Total registers	36	48	64
Total pins	57 (7%)	75 (9%)	99 (12%)
Total DSP Blocks	7 (2%)	7(2%)	14 (4%)

Finally, with a validated circuit, we synthesized our design for the Stratix V 5SEEBF45I4 FPGA. For that, we used the Quartus Prime Standart Edition v.18.1.0 software with the area optimization option enabled. Table II presents our results for circuits with 18, 24, and 32 data sizes.

There are seven posit multipliers in this project, as shown in Fig. 3, and the use of DSP blocks is directly associated with PaCoGen's multipliers. For example, for 18-bit and 24-bit implementations, one DSP block was used for each multiplier, thus requiring a total of seven DSP blocks, as shown in Table II. For 32-bit implementation, two DSP blocks were used in each multiplier, resulting in twice the resources used.

The results indicate encouraging perspectives. As shown in Table II, our 24-bit posit neurons require at most 2% of the FPGA resources. This outcome allows us to build a reasonable network even without any unexplored and possible optimizations, e.g., pipelines, time multiplexing neuron's simulations, reducing even further the data size. Hence, there is much room for substantial improvements in future works.

The proposed approach also optimizes memory usage. Considering that the simulation of each neuron results in a time series of the variable 'v', reducing the number of bits by 25% (from 32 to only 24) can increase the length of time of the simulation in the same proportion. On the other hand, if there are no other restrictions (i.e., bandwidth, memory, available resources in the FPGA), this reduction of bits can increase the number of neurons implemented in the same amount of memory.

IV. Conclusion

This work proposed to implement, embedded in FPGA, a biologically plausible neuron model with Posit Arithmetic using a smaller number of bits. The results obtained showed that the model of neurons with Posit arithmetic could reproduce the same results obtained by the IEEE 754 standard with less memory and the same precision.

These results open space for future work, such as optimizing the circuit designed with Posit arithmetic to use fewer resources and reducing DSP blocks' use. In this way, it will be possible to implement spiking neural networks with even more neurons, contributing to Computational Neuroscience applications in a novel way.

References

[1] N. Kriegeskorte and P. K. Douglas, "Cognitive computational neuroscience," *Nature Neuroscience*, vol. 21, no. 9, pp. 1148–1160, 2018.

[2] R. A. McDougal, A. S. Bulanova, and W. W. Lytton, "Reproducibility in computational neuroscience models and simulations," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 10, pp. 2021–2035, 2016.

[3] M. Topalidou, A. Leblois, T. Boraud, and N. P. Rougier, "A long journey into reproducible computational neuroscience," *Frontiers in Computational Neuroscience*, vol. 9, p. 30, 2015.

[4] S. M. Crook, A. P. Davison, R. A. McDougal, and H. E. Plesser, "Reproducibility and rigour in computational neuroscience," *Frontiers in Neuroinformatics*, vol. 14, p. 23, 2020.

[5] G. Trensch, R. Gutzen, I. Blundell, M. Denker, and A. Morrison, "Rigorous neural network simulations: A model substantiation methodology for increasing the correctness of simulation results in the absence of experimental validation data," *Frontiers in Neuroinformatics*, vol. 12, p. 81, 2018.

[6] R. Gutzen, M. von Papen, G. Trensch, P. Quaglio, S. Grün, and M. Denker, "Reproducible neural network simulations: Statistical methods for model validation on the level of network activity data," *Frontiers in Neuroinformatics*, vol. 12, p. 90, 2018.

[7] R. Pauli, P. Weidel, S. Kunkel, and A. Morrison, "Reproducing polychronization: A guide to maximizing the reproducibility of spiking network models," *Frontiers in Neuroinformatics*, vol. 12, p. 46, 2018.

[8] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomputing Frontiers and Innovations*, vol. 4, no. 2, pp. 71–86, 2017.

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

[10] ——, "Which model to use for cortical spiking neurons?" *IEEE transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[11] C. Leong, *Softposit library*, Jun. 2021. [Online]. Available: https://gitlab.com/cerlane/SoftPosit.

[12] M. K. Jaiswal and H. K.-H. So, "Pacogen: A hardware posit arithmetic core generator," *IEEE Access*, vol. 7, pp. 74 586–74 601, 2019.