Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Bandwidth-Efficient Emulator of Biologically-Relevant Spiking Neural Networks on FPGA

GIANLUCA LEONE[1], LUIGI RAFFO[2], PAOLO MELONI.[3]
[1]Università degli Studi di Cagliari, Cagliari, Italy (e-mail: gianluca.leone94@unica.it)
[2]Università degli Studi di Cagliari, Cagliari, Italy (e-mail: raffo@unica.it)
[3]Università degli Studi di Cagliari, Cagliari, Italy (e-mail: paolo.meloni@unica.it)

Corresponding author: Paolo Meloni (e-mail: paolo.meloni@unica.it).

ABSTRACT Closed-loop experiments involving biological and artificial neural networks would improve the understanding of neural cells functioning principles and lead to the development of new generation neuroprosthesis. Several technological challenges require to be faced, as the development of real-time spiking neural network emulators which could bear the increasing amount of data provided by new generation High-Density Multielectrode Arrays. This work focuses on the development of a real-time spiking neural network emulator addressing fully-connected neural networks. This work presents a new way to increase the number of synapses supported by real-time neural network accelerators. The proposed solution has been implemented on the Xilinx Zynq 7020 All-Programmable SoC and can emulate fully connected spiking neural networks counting up to 3,098 Izhikevich neurons and 9.6e6 synapses in real-time, with a resolution of 0.1 ms.

INDEX TERMS APSoC, fixed-point, FPGA, neural emulator, hardware accelerator, neural engineering, real-time, spiking neural network.

I. INTRODUCTION

DURING the past decades the comprehension of biological neural network phenomena has been at the center of researchers' interest in the medical and biomedical communities. Countless software and hardware instruments have been developed to enhance the understanding of neural cells' working principles [1]. Some tools can simulate biological neural networks by relying on a wide range of mathematical models having a different level of detail [2]. These kinds of tools can help the investigation of how neurons interact with each other, even though more and more often they are also exploited to address completely different problems, such as neuromorphic computing [3].

New generation High-Density Multielectrode Array, scaled from hundreds to thousands of recording sites [4], pushing for the development of signal processing systems capable of sorting order of magnitude more neural data in real-time than in the past [5], and artificial neural networks capable to keep up and process the incoming data. This translates into an imminent demand for bigger and more-connected neural networks. As a result, during the last years, the development of neural networks accelerator has increased consistently [6]. Moreover, networks of neural units are innately parallel, which means, standard Von Neumann architectures are not the best fit to simulate such networks. Therefore, in a so fickle and constantly-evolving environment, programmable accelerators, such as Field Programmable Gate Array (FPGA) based accelerators are best suited to the parallel and ever-changing demands nature of the experiments. Such hardware tools not only permit scaling down simulation time, but also make possible real-time interactions between artificial and biological neural networks in a closed-loop fashion.

In this work, it is proposed a new method to increase the maximum number of synapses that can be emulated in real-time, without sacrificing the physiological dynamics and latency of biological neural networks. The method takes advantage of a physiological delay that affects the spike propagation along the cell's axon. This phenomenon, called axonal delay [7], makes possible to exploit the off-chip memory to store the synaptic weights. Furthermore, we applied the proposed method during the design of an FPGA-based hardware accelerator targeting fully connected neural networks of Izhikevich spiking neurons [8]. Indeed, on one hand, spiking neural networks encode information in the

temporal domain, as biological neural cells do, emulating de facto more accurately the dynamics of biological cells, on the other hand, their implementation is more memory demanding than non-spiking neural networks ones. Therefore, in the case of low-end FPGA implementing spiking neural networks, as the FPGA embedded into the Xilinx All-Programmable SoC (APSoC) XC7Z020 used in this work, where is not possible to store more than 5 Mb of data relying on the on-chip memory only, the sizes of the network cannot grow above a certain limit.

The off-chip DDR memory has been used in other works that utilize spiking neural networks [9][10][11], however, the focus of these works was on image classification, rather than in the study of biological phenomena, therefore their architecture is not meant to be interfaced with a biological system which can require continuous interaction with a 0.1 ms resolution.

The main findings of this work can be summarized as follows:

- We demonstrate the physiological spike propagation delay present in biological neural networks can be exploited in the real-time emulation of spiking neural networks, guaranteeing a higher number of synaptic connections than by only using on-chip memory;
- We demonstrate Xilinx's APSoCs are eligible to apply the presented method, as their off-chip DDR memory has an adequate bandwidth to transfer the synaptic weights, and their use allows to increase the number of synapses that can be emulated in real-time;
- We demonstrate the Izhikevich neuron model equations [8] can be integrated into fixed-point arithmetic by relying on a few FPGA resources, such as DSP and LUT, without consistent behavioral variations.

The remainder of this article is organized as follows. Section II is an overview of existing FPGA-based neural network accelerators. Section III describes the utilized neuron model and his fixed-point arithmetic. Section IV is an overview of the hardware architecture. Section V presents the results in terms of accuracy and performances achieved by this work. Section VI is a comparison with the state of the art. Section VII is left to conclusions and future works.

II. RELATED WORK

A wide scope of software and hardware tools addressing spiking neural network emulation have been developed in the last few years. Software tools such as Nest [2], Neuron [12], and Brian [13] are well suited for biologically realistic simulation of spiking neural networks. They are flexible, and widely used by the scientific community for a wide range of experiments. However, they require larger and larger computer clusters for simulating high-count neural networks [14] and therefore are not the best fit for embedded applications.

Alternatively, parallel computing systems, implemented on a wide range of different platforms, such as CPU, GPU, and FPGA clusters, can achieve high throughput either. SpiNNaker [15] is a multiprocessor chip organized in a mesh of 48 neural computational cores, each made by 18 ARM968 processors. A board equipped with 4 SpiNNaker chips is capable of emulating in real-time a range of synapses going from 8e5 to 1.6e7 and a number of neurons ranging from 1,600 to 16,000, depending on the complexity of the neuron model used. NeuroFlow [16] is an FPGA-based spiking neural network simulation platform capable of emulating both integrate-and-fire and Izhikevich neurons. When hosted by a cluster of 6 FPGAs it can simulate about 600,000 neurons, and from 1,000 to 10,000 synapses per neuron. The total amount of neurons decreases to 400,000 when the emulation is in real-time.

Moreover, at the state of the art, exists a broad collection of real-time FPGA-based spiking neural network accelerators more suited for embedded applications, having different scales, architectures, and use cases. Some work aims to implement low-power solutions, such as [17], where a neural network of 800 neurons and 12,544 synapses is implemented on a Xilinx Virtex-6 FPGA. The system implements a simplified Leaky Integrate-and-Fire (LIF) model [18] with a time resolution of 1 ms, and embeds real-time learning capabilities by integrating a simplified version of the Spike-Time Dependent Plasticity (STDP) algorithm [19]. Other works make use of the reprogrammable feature of FPGA and present configurable designs which could be exploited for a wider range of experiments, such as the work Snava [20]. Snava is a real-time programmable multi-model spiking neural network emulation system, capable of hosting up to 12,800 neurons and 20,000 synapses. The system, implemented on a Xilinx Kintex-7 FPGA, guarantees a resolution of 1 ms. The Snava system, employing a Graphical User Interface (GUI), permits to monitor the spiking activity, to configure the neuron, the synapse model, and the interconnections.

The hardware implementation presented in [21] focuses on studying fully-connected neural networks; their real-time emulator targets closed-loop experiments, and it is hosted by a Xilinx Virtex-6 FPGA. The system implements 1,440 Izhikevich neurons with a resolution of 0.1 ms and a spike latency of 1 ms.

Other studies focus on more specific problems, such as minimizing the neurons emulation latency down to 8 ns to increase the maximum number of neurons that can be emulated in a single FPGA chip, at the expense of the biological meaning [22]. This result has been achieved by designing a systolic array to integrate a simplified version of the Izhikevich neural model. By following the considerations found in [23], it is possible to decrease the computational load without renouncing the main emulating features of the Izhikevich model.

Conversely, Luo et. al [24] presented a bio-realistic cerebellum model, and propose it as the first step for the realization of neuroprosthesis systems with the purpose of substituting damaged motor control units in the brain. Luo et. al [24] propose a Network on Chip (NoC) hardware architecture, implemented on a Xilinx Virtex-7 FPGA, capable of emulating 101,000 LIF neurons [20] and 100,000 synapses in closed-

TABLE 1. Real-time FPGA-based neural network emulators comparison

Work	Year	Target FPGA	Neurons	Synapses
this work	2021	Xilinx Virtex-6	3,098	9.6e6
Gupta [17]	2020	Xilinx Virtex-6	800	1.25e4
Sripad [20]	2018	Xilinx Kintex-7	12,800	2.00e4
Pani [21]	2017	Xilinx Virtex-6	1,440	2.07e6
Bandeira [22]	2017	Altera Stratix IV	364	3.64e2
Luo [24]	2016	Xilinx Virtex-7	101,000	1.00e5
Ambroise [26]	2013	Xilinx Virtex-4	117	1.37e4
Han [9]	2020	Xilinx Kintex-7	2,842	1.86e6
Panchapakesan [10]	2020	Xilinx ZCU102	2,410	-

loop experiments.

In Khodamoradi et. Al [25] is proposed an architectural solution to support several axonal delays without using extra FIFOs, schedulers, and separate routing networks for spiking feedforward neural networks.

In Ambroise et. al [26], a folded low-resources architecture capable of emulating 117 Izhikevich neurons in real-time with a time resolution of 1 ms is presented. The system is implemented on a Xilinx Virtex-4 chip, and the interconnections of the neuron are configurable, ranging from zero to a fully connected network.

Finally, in Han et. Al [9] and Panchapakesan et. Al [10][11] Leak Integrate and Fire and Integrate and Fire based spiking neural networks are used to address image classification tasks on the MNIST and CIFAR-10 datasets on Xilinx Zynq devices, chip provided with both FPGAs and ARM processors. Their approaches take advantage of the off-chip DDR memory to store the weights of the network, however, not being designed as a biological relevant neural network emulator, it is not applied any method to guarantee the physiological dynamics of biological neural networks are respected.

Table I summarizes the main characteristics of the above-mentioned FPGA works. It is possible to notice how the presented architecture, being fully connected, owns a different balance between its number of neurons and synapses than most of the works in Table I [9][10][17][20][22][24], a significantly higher number of synapses and neurons than the other works addressing fully connected neural networks [21][26], and a higher number of synapses in general. In fact, the presented work has been conceived as a tool to study the behaviors of biological neural networks, rather than a machine learning accelerator. Therefore, willing to allow the emulation of arbitrarily connected population of neurons, being the connectivity of the neurons still a topic of interest in neuroscience research, it has been designed a system where the neurons can be connected without restrictions, up to be completely connected.

III. IZHIKEVICH NEURON MODEL

The Izhikevich model [8] permits the emulation of a large set of biological behaviors at a low computational cost. The model is composed by a two dimensional system of ordinary differential equations 1, 2, plus a reset condition 3.

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

$$v > v_{th} \rightarrow \begin{cases} v = c \\ u = u + d \end{cases} \tag{3}$$

v is the membrane potential of the neuron, and u is the membrane recovery variable, both measured in mV. The term v_{th} is the threshold above which the modeled neuron fires a spike. When it happens, both the membrane potential and the membrane recovery variable are reset. The dimensionless parameters a, b, c, and d permit tuning the model in order to emulate properly the behaviors of neocortical and thalamic neurons. I is the synaptic current, it permits taking into account the synaptic connection among neurons. Indeed, each synapse can be described as an oriented and weighted connection between two neurons. When a neuron fires, its post-synaptic neurons counts the spike by adding to I the weight of its interconnection.

A. THE QUANTIZATION PROBLEM

The simplest and most common way to evaluate the Izhikevich model, nevertheless the way used in [8], is the one-step forward Euler scheme, described by 4, 5, and 6.

$$v_{k+1} = v_k + h(0.04v_k^2 + 5v_k + 140 - u_k + I + I_e) \tag{4}$$

$$u_{k+1} = u_k + ha(bv_k - u_k) \tag{5}$$

$$v_{k+1} > v_{th} \rightarrow \begin{cases} v_{k+1} = c \\ u_{k+1} = u_k + d \end{cases} \tag{6}$$

Where h is the time step, equal to 0.1 ms, chosen to be compatible with most High-Density Multielectrode Arrays based data acquisition systems, and I_e is a parametric DC offset.

The above equations are solved by using fixed-point arithmetic so that a considerable amount of FPGA's resources could be saved. However, we found out the accuracy and the convergence of the model, when operating in fixed-point, are not to be taken for granted. In order to investigate the behavior of the fixed-point implementation of the model, two MATLAB scripts have been developed. The former is used to provide a trustworthy ground truth for the experiments, which has been obtained by making use of floating-point arithmetic. The latter script is used to test the accuracy of the fixed-point solution at different levels of quantization.

IV. HARDWARE SPIKING NEURAL NETWORK

The spiking neural network emulator architecture is shown in Figure 1. The Potential modules integrate the Izhikevich equations, updating both the membrane potentials of the neurons and the spike conditions. The neural potentials and the spikes conditions are stored in two BRAM-based memories called Potential mem and Spike mem. The Izhikevich equations' parameters are stored in the BRAM-based memory

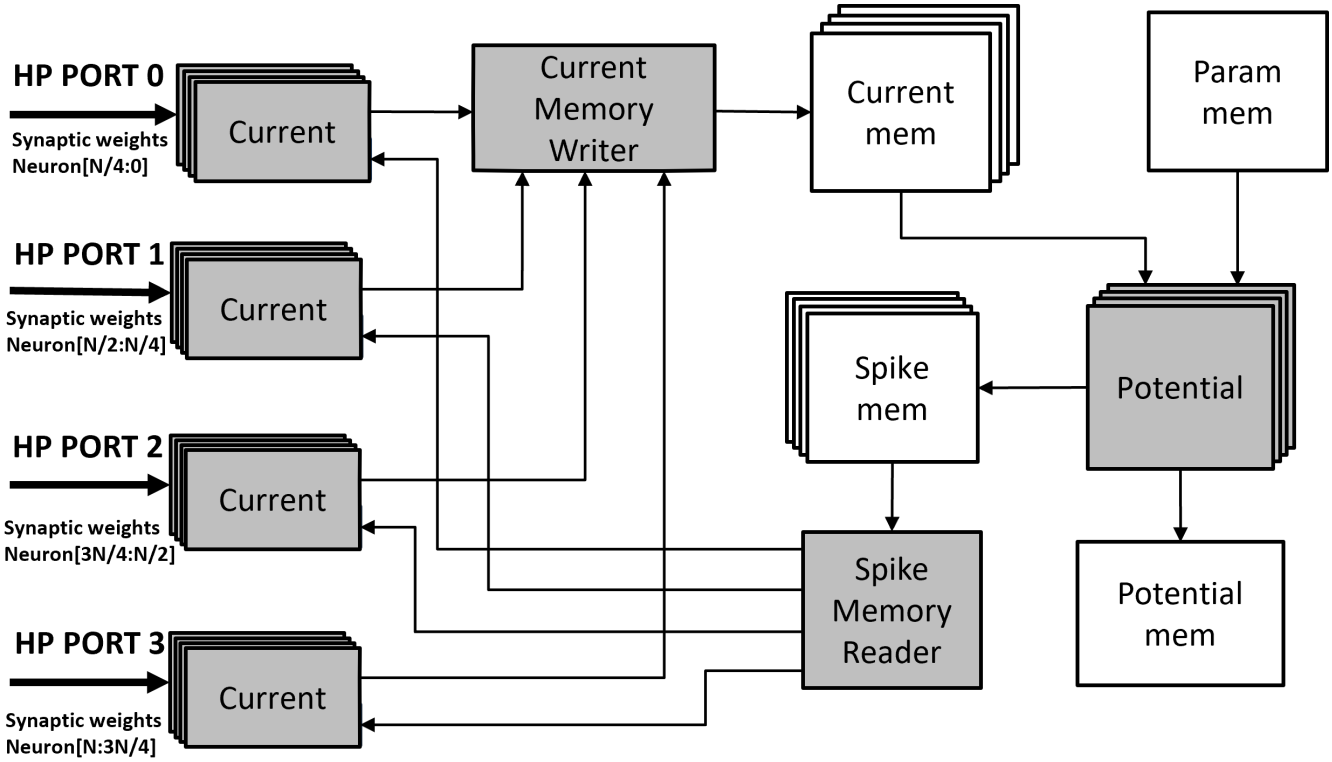

FIGURE 1. The block diagram of the neural network emulator.

Param mem.

The synaptic current is stored in an additional BRAM-based memory called Current mem, updated by the Current modules. In order to update the Current memory, the Current modules read the stream of synaptic weights coming from the off-chip DDR through four axi-stream interfaces and the spike conditions from the Spike mem. The system is designed to exploit the shared characteristics of the Xilinx Zynq-7000 family APSoC devices. The architecture is parametric, so the netlist can be generated to fit in different devices of the family and to emulate neural networks of different sizes. The system setup presented in this paper is implemented on a Z-7020 chip.

The number of synaptic weights grows quadratically with the number of neurons in fully connected neural networks, and the Block-RAM (BRAM), which are the internal memories embedded in Xilinx's FPGA, are usually the bottleneck that prevents to increase the number of synapses over a certain limit. In the fully-connected neural network implemented in [21], the synaptic weights are stored on-chip, in the BRAMs, and the largest possible network which fits in is of about 1,440 neurons, obtained using 392 36 kb BRAM tiles in a Xilinx Virtex-6 XC6VLX240T chip. Indeed, if on one hand, the Programmable Logic (PL) is capable of performing heavy parallel computations, on the other hand, the available memory space is not enough to host larger fully-connected neural networks. Willing to overcome this result, it is possible to exploit the off-chip DDR memory, which is the largest memory available in the Zedboard development board used in this work. The DDR is 512 MB large, and it can be accessed concurrently from 4 High-Performance AXI ports (HP AXI ports), by using 4 AXI DMA operating at their maximum speed of 150 MHz [27], with an overall theoretical bandwidth of 4.8e9 B/s[27]. Keeping the same sampling frequency of [21], which is 10 kHz, it would be possible to move about 4.8e5 B in 0.1 ms, which by using synaptic weights of 8 bits each, would correspond to a fully-connected neural network of about 692 neurons. However, taking into consideration the biological delay which exists between the generation of a spike in the soma, and the propagation of the spike through the axon, towards the post-synaptic neurons, called Axonal Delay (AD), it is possible to relax the 0.1 ms deadline in favor of a looser one. By using an axonal delay of 1 ms, as in [21], it would be possible to transfer the whole set of weights every millisecond instead of every tenth of a millisecond and reuse them 10 times to solve the Izhikevich equations. In this way, it would be possible to transfer 10x weights, which correspond to a fully connected neural network of 4.8e6 synapses, and therefore 2.191 neurons. The computational load increases consistently, as a matter of fact, in the case of an axonal delay of 1 ms, and an integration frequency of 10 kHz, the throughput of the digital system should increase by a factor of 10. However, this is a mandatory requirement if more neurons and synapses need to be emulated in the same amount of time. Figure 2 (a) shows how without taking advantage of the axonal delay the neurons are updated at

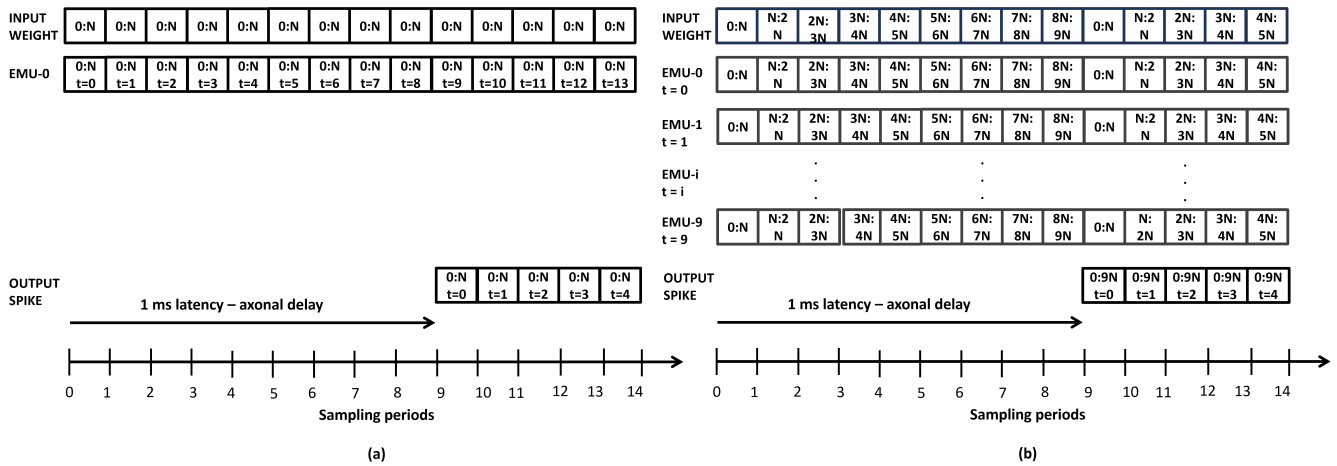

FIGURE 2. (a) The execution flow of a network which takes into account the axonal delay. (b) The execution flow of a network which takes advantage of the axonal delay to emulate more synapsis in real-time.

every integration step, whereas the spikes are forwarded 1 ms later their computation to the output, because of the axonal delay. By looking at Figure 2 (b) it is possible to observe the synaptic weights can be transferred during a longer period of time, 10 integration steps in the example, and the neuron emulation can be spread along this period, by increasing by a factor of 10 the operations per byte.

In the work [7], typical axonal delay values are reported for different mammalian species and neural tissue areas. Values range from 0.3 ms for fast-conducting axons, such as cat visual thalamocortical axons, up to 130 ms, required to reach axon terminals in monkeys' visual cortex.

A. ARCHITECTURAL OVERVIEW

The main blocks of the biological neural network emulator and their interconnections are shown in Figure 1. The main actors are the Potential modules, which integrate the Izhikevich Equations 4, 5, and 6, and the Current modules which compute the synaptic current. Moreover, the spikes, the synaptic currents, the parameters of the Izhikevich model a, b, c, d, the membrane potentials, and the membrane recovery variables are stored inside BRAM-based memories, called after their contents, as shown in the schematic depicted in Figure 1.

The Potential modules are fully pipelined and have a throughput of one integrated neuron per clock cycle. By taking advantage of the parametric port size of Xilinx's BRAMs, it is possible to instantiate more potential modules in parallel when higher throughput is required. Moreover, the current modules are fully pipelined, this allows to achieve a throughput of 8 summed synaptic weights per clock cycle. The synaptic weights are stored in the off-chip DDR and streamed through four AXI High-Performance Ports to the programmable logic. The stream is handled by four DMAs. A different current module is instantiated to handle each of the four streams of synaptic weights. In this way, it is possible to reach the maximum throughput possible, since the four streams of weights are not processed in time-multiplexing by the same current module, but in parallel from four different modules. With this setup, there is no need to store the synaptic weights on-chip, since they are processed as they arrive, and the BRAMs can be saved to store the neurons' model parameters. If the axonal delay is set to a number higher than 0.1 ms (the time resolution chosen in this work), more current modules can be placed in parallel, and the current of multiple time steps can be computed at once. Multiple instances of the Current mem and the Spike mem are required too. The modules Spike Memory Reader and Current Memory Writer are used by the Current modules as interfaces to write the computed synaptic currents into the Current memory and read the spike conditions from the Spike memory. These interface modules permit sharing a single memory port between the four sets of current modules in a time-multiplexed fashion, without creating any bottleneck, as explained in Section IV-B

1) Data transfer

The synaptic weights are moved from the DDR to the Programmable Logic (PL). The transaction is entrusted to 4 Xilinx AXI DMA IPs, each one connected to a different AXI High-Performance port. The response channels of the AXI buses are used to write back the spikes of the network in the DDR.

2) Potential

The Potential module implements the Izhikevich Equations 4, 5, and 6, and it has been designed by using the Verilog hardware description language, as the other modules. Equations 4, 5, and 6 are implemented by means of a systolic array, shared among the neurons in a time-multiplexed fashion, which guarantees a throughput of one integrated neuron per clock cycle. Multiple Potential modules can be instanced in the design.

The membrane potential pipeline makes use of three DSP and

a LUT-based multi-inputs adder, its architecture is shown in Figure 3. The first DSP is used to multiply the membrane

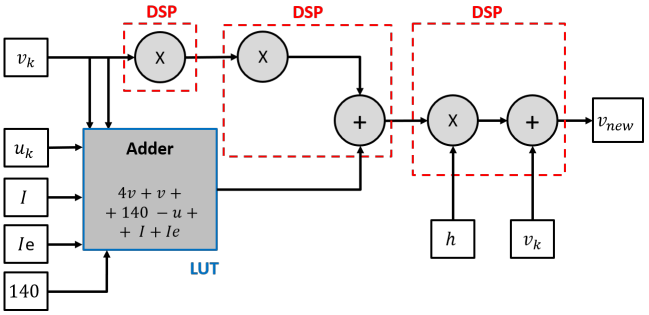

FIGURE 3. The block diagram of the membrane potential pipeline.

potential v_k to the constant 0.04. The product feeds the input of the second DSP block which multiplies $0.04v_k$ again by the membrane potential, obtaining $0.04v_k^2$. Concurrently, the addition $sum_v = 5v + 140 + u_k + I + I_e$ takes place by means of a LUT-based multi-inputs adder. The sum goes in input to the post-multiplication adder embedded in the second DSP, so that the second DSP's output could be $\delta v = 0.04v_k^2 + sum_v$. The third DSP implements the operation $v_{new} = h\delta v + v_k$.

The membrane recovery variable pipeline can be mapped into two additional DSPs. The block diagram, implementing Equation 5 is shown in Figure 4. The former DSP implements

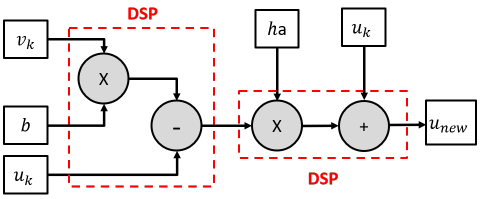

FIGURE 4. The block diagram of the membrane recovery variable pipeline.

the operation $sum_u = bv_k + u_k$, and feeds the latter DSP, which multiplies sum_u by the pre-computed parameter ha, and add u_k to it, obtaining $u_{new} = u_k + ha(bv_k + u_k)$.

The reset or spike condition stated by equation 6 is verified by a comparator, which in turn controls two multiplexers as shown in Figure 5. When $v_{new} > v_{thr}$ the reset condition is activated, the second inputs of the multiplexers are chosen, therefore $v_{k+1} = c$ and $u_{k+1} = u_{new} + d$. Otherwise $v_{k+1} = v_{new}$ and $u_{k+1} = u_{new}$.

Once evaluated, the membrane potential v_{k+1}, the recovery variable u_{k+1}, and the spike condition, are stored in the Potential memory and in the Spike memory.

3) Current

The Current module evaluates the synaptic current of every neuron of the network, so that Equation 4 could be integrated. The Current module architecture is shown in Figure 6. The synaptic weights are transmitted from the DDR through

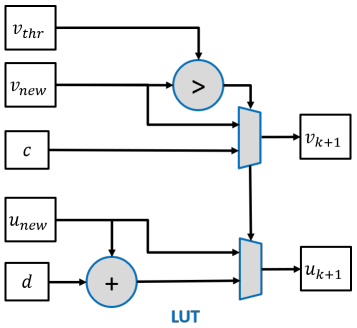

FIGURE 5. The block diagram of the reset condition architecture.

the AXI High-Performance ports, and processed on the fly, without the need to buffer them. Every synaptic weight is counted in the evaluation of the synaptic current if the pre-synaptic neuron is active. The spike conditions are read from the Spike memory as the weights come, in order not to count the weights of the inactive neurons. The weights of the inactive neurons are excluded from the addition by means of a *logical-and* involving both the weights and the spike conditions. Each AXI High-Performance port transmits 64 bits per clock cycle, and since the synaptic weights are 8 bits long, the Current module processes 8 synaptic weights per clock cycle. Four clusters of Current modules are instanced in the design, one for each AXI High-Performance port, and every cluster is made by R Current modules, where R is the ratio between the selected axonal delay (AD) and the integration step, so that R synaptic currents could be evaluated in parallel, without retransmitting or storing the synaptic weights. The weights are added by means of a LUT-based adder, whose result drives a DSP-based accumulator, which permits computing the synaptic current during multiple clock cycles. Once evaluated, the synaptic current is stored in the Current memory.

B. EXECUTION FLOW

The system execution flow repeats every time the selected axonal delay period expires. Each cycle can be described as follow:

- The Processing System (PS) enables four AXI DMA, which handle the transmission of the synaptic weights from the DDR to the PL, through 4 High-Performance AXI-Stream buses.
- Four clusters of R Current modules process the synaptic weights transmitted from the four DMAs through the four AXI High-Performance ports, meanwhile, the Spike memories are accessed not to count the weights of the inactive synapses. A module called Spike Memory Reader arbitrates the access to the spike memory through a single port, by allowing only a cluster of Current modules per clock cycle to access, in the meantime the others wait. The AXI-Stream transactions of the waiting clusters of current modules are paused. Every

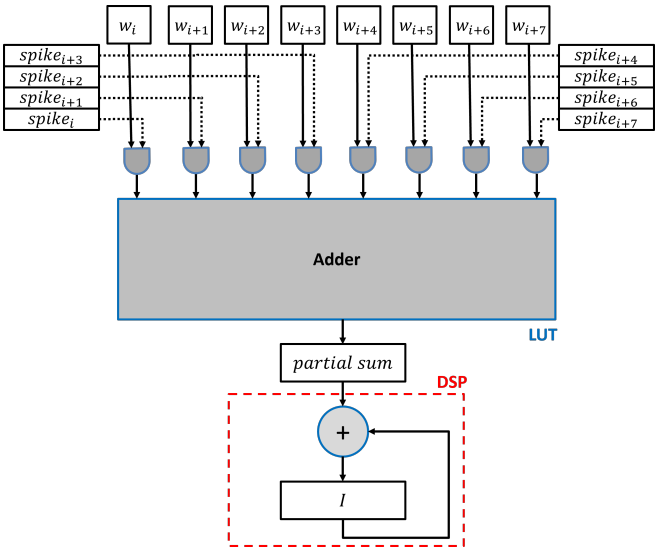

FIGURE 6. The block diagram of the Current module architecture

time the Spike Memory Reader gets access to the spike memory on behalf of a cluster of Current modules, it reads in advance the spikes needed by that cluster for the next 4 clock cycles, taking advantage of the configurable width of the BRAM ports. Therefore, since 8 weights are transmitted per clock cycle, 32 spikes are read each time. By doing so, during steady-state processing, access conflicts do not take place.

- Once a stream of weights starts, each Current module in the same cluster computes the synaptic current of the same neuron, just at a different point in time. This is possible because the computation of the synaptic currents requires the synaptic weights and the spike conditions of AD ms before, and therefore they have just been computed and stored in the Spike memory. Once evaluated, the synaptic currents are stored in the Current memory. The Current memory is organized as a multi-banked memory of R banks, where R is the ratio between the axonal delay and the integration step. Each bank has an entry per neuron, whereas different banks host currents of different integration steps.
- As soon as the synaptic currents are available, it is possible to integrate the Izhikevich Equations. In order to keep the potential fetching logic simple, four Potential modules integrate the Izhikevich equations of four different neurons concurrently. Once finished, the membrane potentials of the same neurons in the next integration steps are evaluated. When all the integration steps of those neurons are evaluated, the Potential modules start integrating 4 new neurons. The results are stored in the Potential and in the Spike memories.
- The evaluated spikes are written into the DDR. The spikes are moved by using the response channels of the AXI High-Performance ports.

V. RESULTS

In this section, the performance, the hardware resource utilization, and the accuracy of the presented work are analyzed.

A. SYSTEM PERFORMANCE

The design presented in this paper is implemented on a ZedBoard, a low-cost development board for the Xilinx Zynq Z-7020 All-Programmable SoC. The architecture is parametric along multiple axes, as the number of neurons, synapses, and the axonal delay. The system setup chosen is the one that permits the emulation of the maximum number of fully connected neurons, which is 3,098, with 9.6e6 synapses and an axonal delay of 3 ms. To achieve this result, instead of storing the synaptic weights into the chip's internal BRAMs, which are not enough to memorize 9.6e6 bytes, the synaptic weights are stored in the off-chip DDR and transferred through the 4 AXI High-Performance ports present in every Zynq device. Four DMAs take care of the transmission of the weights, clocked at their maximum speed of 150 MHz [27].

Taking into account that the chosen emulation step is 0.1 ms, which is a common value in neuro-engineering applications, the system should process the whole set of synaptic weights every 0.1 ms. However, taking advantage of the axonal delay, a physiological latency that exists between the generation of a spike in the soma, and its propagation through the axon, towards the post-synaptic neurons [7], it is possible to generate the outputs with a certain latency. Taking advantage of this, it becomes possible to spread the transmission of the synaptic weights into more than a tenth of a millisecond, and then have more weights transmitted without violating the physiological dynamics of the neuronal cells. Increasing the axonal delay, from the performance point of view, permits to increase the operational intensity, i.e. the number of operations per byte, and therefore to enhance the FPGA throughput, at the expense of instantiating multiple current modules. The Roofline model, shown in figure 7, helps understand how the operational intensity and the performances of the architecture change depending on the Axonal Delay (AD) value. The x-axis is the operational intensity measured in operations per byte ($ops/byte$), and the y-axis represents the overall performance in terms of the number of operations per second ($Gops$). It can be observed how the operational intensity rises up to 30 $ops/byte$ as the axonal delay increases. It is not possible to go beyond this limit, reached for an axonal delay equal to 3 ms, because of the saturation of the BRAM tiles required to store the Izhikevich parameters, among which are the synaptic currents.

The computational bound of 240 Gops, represented by the upper horizontal dotted line in Figure 7, would be reached with an axonal delay equal to 5 ms, which would permit an operational intensity of 50 ops/byte. To achieve such performances, it would be necessary to instantiate 50 current modules per axi port, for an overall number of 200 current modules, any of which would compute 8 additions per clock cycle at 150 MHz.

In the case of a 3 ms axonal delay: the theoretical number

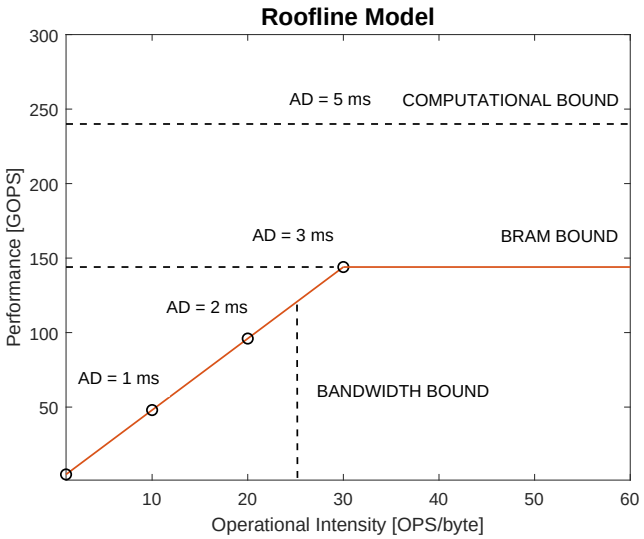

FIGURE 7. The Roofline model at the varying of the Axonal Delay (AD).

of weights that can be transmitted in real-time in 3 ms is 1.4e7 [27]. However, we experimented that is not possible to transmit more than 9.6e6 synapses. To process the 4 streams of weights coming from the 4 AXI High-Performance ports, the PL has been clocked at the same frequency of 150 MHz used to stream the weights. Thirty instances of the Current module have been placed per each AXI High-Performance port, for an overall number of Current modules equal to 120. Moreover, four Potential modules read the synaptic currents once computed, integrate the Izhikevich equations, and evaluate the spike conditions. With this setup, the presented system is capable to emulate in real-time a fully connected neural network of 3,098 neurons and 9.6e6 synapses, with a resolution of 0.1 ms and an axonal delay of 3 ms.

B. HARDWARE RESOURCE UTILIZATION

The Zynq 7020 hosts 106,400 Flip-Flops (FFs), 53,200 Look Up Tables (LUTs), 140 36Kb BRAMs tiles, and 220 DSP48E1 slices.

The Current module is implemented using an array of and-gates, a LUT-based eight-inputs adder, and a DSP-based accumulator. To meet the timing constraints of 150 MHz, two pipeline stages have been introduced inside the cascade of the and-gates array and the multi-addends adder. The pipeline stages have been properly placed along the combinational paths by enabling the *retiming* option in the settings panel of the Vivado synthesizer. Table 2 shows the post-implementation resource requirement of a single Current module, obtained utilizing Vivado 2019.2. Note that 120 Current module instances are necessary to properly work in real-time since the selected axonal delay is 3 ms.

The potential module is mapped in hardware by the use of 5 DSP, a LUT-based multi-inputs adder, and a LUT-based comparator. One DSP is used as a multiplier only, whereas the remaining four DSP are configured to use both the multiplier and the post-multiplier adder. To meet the timing constraints of 150 MHz, the Potential module has been pipelined by adding three pipeline stages for each multiplication and multiply-and-accumulate operation, one for each addition, and one for the threshold comparison, for a total of 10 pipeline stages. Table 2 shows the post-implementation resource requirement of a single Potential module, four of them have been instanced in the presented design.

TABLE 2. Hardware resources distribution among the main modules

	Processing Elements		Memory			
Resource	Current	Potential	Current	Potential	Param	Spike
INSTANCE	120	4	30	1	1	60
LUT	45	173	0	0	0	0
LUTRAM	0	0	0	0	0	0
FF	78	452	0	0	0	0
BRAM	0	0	2	5	13.5	0.5
DSP	1	5	0	0	0	0

The Current memory has an entry of 15 bits per neuron and two 60 bits ports; 30 instances of the Current memory are placed since the currents of 30 consecutive integration cycles are computed at the same time, for a total amount of 60 BRAMs as shown in Table 2.

The Potential Memory has an entry of 42 bits per neuron; one instance of the Potential memory requires 5 BRAMs, as shown in Table 2. A single instance of the Potential memory is sufficient in the design, since the old potential values, once used, can be overwritten. Moreover, 13.5 BRAM are required to store the neuron parameters (a, b, c, d, I_e, h).

The Spike memory contains a bit per neuron, and 60 instances are required, two per each integration step, since it is not possible to overwrite the entries of the Spike memory while computing the new spike conditions. In fact, the synaptic currents of the following neurons should be computed by relying on the old spike conditions. Every Spike memory requires a single 18Kb BRAM, for a total amount of 30 36 KB BRAM tiles as shown in Table 2.

The post-implementation resource utilization report of the whole system is shown in Table 3. The Zynq 7020 chip is only partially used. As expected the BRAM tiles are almost fully occupied, as the 93.21% is utilized. Since most of the data-crunchy computational logic has been mapped into the DSPs, 63.64% of the DSPs are used. The 66.96% of the FFs are free, as well as more than half of the LUTs (53.98%), and only 4.5% of the LUTRAMs are used.

TABLE 3. Resource utilization table

Resource	Utilization	Available	Utilization %
LUT	24,480	53,200	46.02
LUTRAM	802	17,400	4.61
FF	35,158	106,400	33.04
BRAM	130.5	140	93.21
DSP	140	220	63.64

C. ACCURACY EVALUATION

Equations 1, 2, and 3 are solved by using fixed-point arithmetic, so that a considerable amount of FPGA's resources could be saved. However, we found out the accuracy and the convergence of the Izhikevich model, when operating in fixed-point, are not to be taken for granted. In order to investigate the behavior of the fixed-point implementation of the model, two MATLAB scripts were implemented. The former is used to provide trustworthy ground truth for the experiments, generated from a floating-point implementation of the Izhikevich model. The latter script is used to analyze the fixed-point accuracy and understand how many bits are needed to smoothly move towards a fixed-point representation. To assess the behavior of the fixed-point neural network a set of 1024 fully-connected Izhikevich neurons were simulated, of which 768 are of the excitatory type, and the remaining 256 are of the inhibitory type. Even if it is possible making use of regular spiking and fast-spiking cells to model the whole set of excitatory and inhibitory neurons respectively, to simulate a more heterogeneous network, the directives proposed in [8] had been followed. The excitatory neurons are modelled by setting $a_i = 0.02$, $b_i = 0.2$, $c_i = -65.0 + 15r_i^2$ and $d_i = 8.0 + 6r_i^2$, with r_i a random variable uniformly distributed on the interval [0,1], and i the neuron index. On the value of r_i depends the kind of neuron dynamic obtained. With $r_i = 0$ the cell dynamic is the one of a regular spiking cell, with $r_i = 1$ is obtained the dynamic of a chattering cell, and with r_i around the value 0.8, is emulated the dynamic of an intrinsically bursting neuron. In a similar way all the inhibitory cells parameters are randomly assigned by using the following rules $a_i = 0.02 + 0.08r_i^2$, $b_i = 0.25 - 0.05r_i^2$, $c_i = -65$ and $d_i = 2$; so that for $r_i = 1$ is obtained the dynamic of a fast spiking cell, and for $r_i = 0$ is simulated the dynamic of a low-threshold spiking cell. For all the excitatory cells the DC offset I_e is set at 4 pA, whereas for all the inhibitory cells the DC offset is set at 2 pA.

The functionality of the fixed-point network has been assessed at both the single-cell and the network levels. The spike jitter, or spike lag, has been verified neuron by neuron between the floating and fixed point networks and used as a comparison metric such in [28], as long as the mean firing rate, and the interspike interval. Moreover, the networks bursts have been analyzed: the mean bursting rate, the burst duration, and the interburst interval of the networks have been compared. The bursts are identified as a sequence of more than 4 spikes with an interspike interval of less than 100 ms.

1) Custom Data Width Selection

The data width of every input, output, and internal signal was chosen to optimize at the same time the emulation accuracy and the resource utilization. We analyzed the dynamics of the membrane potential, the recovery variable, and the synaptic current by relying on the floating-point Matlab simulation of the Izhikevich model. We found out the membrane potential reaches maximum values which fit into 8 integer bits, the recovery variable into 6 integer bits, and as regards the synaptic current, it fits into 8 bits. In order to optimize the data mapping into Xilinx's DSP48E1, the DSP's input data width has not to be exceeded. Conversely, to obtain the best possible accuracy with such processing elements, once placed the integer part of the data in input to the DSP, the remaining bits of the DSP's inputs can be filled with fractional bits. The DSP48E1 contains a multiplier with two input ports of 25 and 18 bits and an adder with three 48-bits input ports, two of which are used to carry out the multiplication. Therefore, in the case of the membrane potential, which goes in input to the DSP multiplier, it is possible to choose 18 bits or 25 bits data widths, corresponding to the formats $< 8.10 >$ or $< 8.17 >$ bits. We did not find any significant difference in accuracy between the two formats, therefore, we chose $< 8.10 >$ to save BRAM tiles. As regards the recovery variable, it does not go directly inside a multiplier, therefore its data width can go up to 48 bits, being 48 bits the input data width of the adder embedded in the DSP. However, 24 bits, with the data format $< 6.18 >$, are a good trade-off between accuracy and BRAM utilization. When it comes to the synaptic current, since its integer part fits into 8 bits, and the synaptic weights have the format $< 1.7 >$ bits, the format $< 8.7 >$ bits can be used.

The constant 0.04 can be represented with the format $< 1.24 >$. The first multiplication $0.04v_k$ requires fewer integer bits than the sum of the two integer parts of the factors involved in the multiplication. In fact, 0.04 is smaller than one. Since $1/16$ is bigger than 0.04, and it would be a 4 digits shift to the right, it can be inferred the integer part will lose 4 bits. Therefore, the new format for $0.04v_k$ will be $< 4.21 >$. The fractional part size has been selected to fit the next multiplier input width, and maximize the emulation precision. The addition $sum_v = 4v_k + v_k + 140 - u_k + I + I_e$ is implemented by means of a multi-addend LUT-based adder. The addends sizes are listed in Table 4. The fractional bits of the term sum_v are the same of the membrane recovery variable u_k, whereas its integer part fits into ten bits. The product $0.04v_k$ is multiplied by v_k and added to sum_v employing a DSP. The integer part of the product fits in eleven bits, whereas the fractional part size is truncated to fourteen bits before being multiplied by the integration step h in the next DSP. The integration step is equal to 0.1, therefore the output of the multiplication $h(0.04v_k^2 + sum_v)$ will require fewer integer bits than the input factors. The power of two 2^{-3} corresponds to a three digits right shift, and it is bigger than 0.1. Then, the integer part of the product will fit into 8 bits. The term $h(0.04v_k^2 + sum_v)$ is added to v_k by means of the post-multiplication adder of the same DSP. Since the dynamic of v_k does not exceed 8 integer bits, as observed in the floating-point simulation, this sum will still fit into 8 integer bits.

The membrane recovery variable pipeline is composed of two DSPs which implement the operations $bv_k - u_k$ and $ha(bv_k - u_k) + u_k$. The 25 bits input port of the first DSP are used for the parameter b, with the format $< 1.24 >$. This term's maximum value is 0.25, which corresponds to

TABLE 4. Potential module signals format

Data	Format	Data	Format
v_k	8.10	0.04	01.24
$0.04v_k$	4.21	$4v_k$	10.10
I	8.7	I_e	5.7
140	09.00	u_k	6.18
sum_v	10.18	$0.04v_k^2 + sum_v$	11.14
h	01.17	v_{new}	8.10
v_{thr}	08.10	c	08.10
b	01.24	$bv_k - u_k$	6.19
ha	01.17	u_{new}	6.18
d	06.18	$weight$	1.7

a two-digits right-shift, therefore bv_k will require 6 integer bits at most. The multiplier output is subtracted by u_k using the adder embedded in the same DSP. The output, in the format $< 6.19 >$, fits into the 25 bits input of the second DSP implementing the operation $ha(bv_k - u_k) + u_k$. The term ha, directly stored pre-computed instead of a (to save a multiplier), can reach the maximum value of 0.01. Therefore it reduces by at least 6 bits the dynamic of $(bv_k - u_k)$. In any case, being $ha(bv_k - u_k) + u_k$ the new value of the membrane recovery variable, it cannot have a dynamic that goes above 6 integer bits, as observed during the floating-point simulation.

2) Single Neuron Behaviour

This section shows the behavior of the Izhikevich neuron obtained by using the custom data width presented in V-C1. Moreover, the single-cell behavior at the varying of the fixed-point data width is shown as well. We selected 10 bits for the integer part. Using less than 10 bits causes data overflows, as pointed out in [26]. On the other hand, using more than 10 bits for the integer part does not provide any accuracy benefits.

Figure 8 shows the superimposition of the fixed- and the floating-point simulation of the membrane potential for several fractional bits widths, within a time window of 200 ms. From left to write regular spiking, chattering, intrinsically bursting, fast-spiking, and low-threshold spiking cells. The parameters used to simulate each neuron are listed in Table 5. In the first row are used 10 fractional bits. The membrane potential superimposition shows evident differences with the floating-point model. Both the number and the timing of the spike differ. In the second row, the number of fractional bits is increased to 16. Starting from this format the spikes count between the floating- and the fixed-point cells is the same, for all the cell types. However, it is still possible to observe a significant timing lag among the spikes. In the third row, 22 fractional bits are used, for a total data size of 32 bits. This format permits obtaining two perfectly superimposed simulations. The last row shows the case where the custom format described in V-C1 is used. There are no significant behavioral differences between using 32 bits fixed-point arithmetic and the custom data width proposed in this work. However, the custom data-width permits to map efficiently the computations into Xilinx's DSPs and LUTs. Moreover, the overall memory required per neuron is 371 bits, about the 61% required if 32 bits data-width is used.

TABLE 5. Single Cell Simulation Parameters Values

Type	a	b	c	d	I_e
Regular Spiking	0.02	0.2 7	-65	8	4
Chattering	0.02	0.2	-55	4	4
Intrinsically Bursting	0.02	0.2	-50	2	4
Fast Spiking	0.1	0.2	-65	2	4
Low-Threshold Spiking	0.02	0.25	-65	2	4

3) Neural Network behavior

The firing patterns of the hardware fully-connected neural network of 1024 Izhikevich neurons were compared to the floating-point Matlab model. The network had been compared along two seconds of activity.

The spike timing for each neuron of the network is shown in Figure 9. The x-axis represents the time in samples (the integration step is 0.1 ms, so 20k samples correspond to 2 seconds), whereas the y-axis has an entry for each neuron of the network; neuron identifiers from 1 to 768 are of excitatory neurons, neuron identifiers from 769 to 1024 are of inhibitory neurons. The firing activity superimposition of Figure 9 shows a match between the hardware and the Matlab reference models. Within 2 seconds of activity, a total amount of 20,874 spikes were fired from the Matlab reference model, whereas 20,886 were fired from the presented hardware implementation. The total number of spikes fired by the networks differs by about 0.06%. The mean firing rate of the two networks is shown in Figure 10; in the case of the hardware network, the MFR is 10.1924 spikes per second, whereas it is 10.1982 spikes per second in the case of the Matlab reference model. Among the spikes fired from the Matlab reference model, 20,620 were correctly replicated from the hardware network, with a maximum timing jitter of 2 ms, which corresponds to the 98.78% of the spikes fired. The 1.27% of the fired spikes are instead false positives, and the 1.22% are false negatives.

The spike jitter distribution is shown in Figure 11, the 98.78% of the spikes are correctly reproduced with a maximum jitter of 2 ms, of which the 89.68% have a time jitter less or equal to 1 ms.

The Inter-Spike interval (ISI) values are shown in Figure 12. The first and the second columns depict respectively the excitatory and inhibitory neurons of the networks, whereas the first and the second rows show the Inter-Spike interval of the Matlab and the Hardware networks. There are not significant difference in the interspike time distributions of the hardware and Matlab networks, for both the inhibitory and the excitatory neurons.

The analysis of the bursting activity shows how the hardware network behavior still retraces the firing pattern of the Matlab

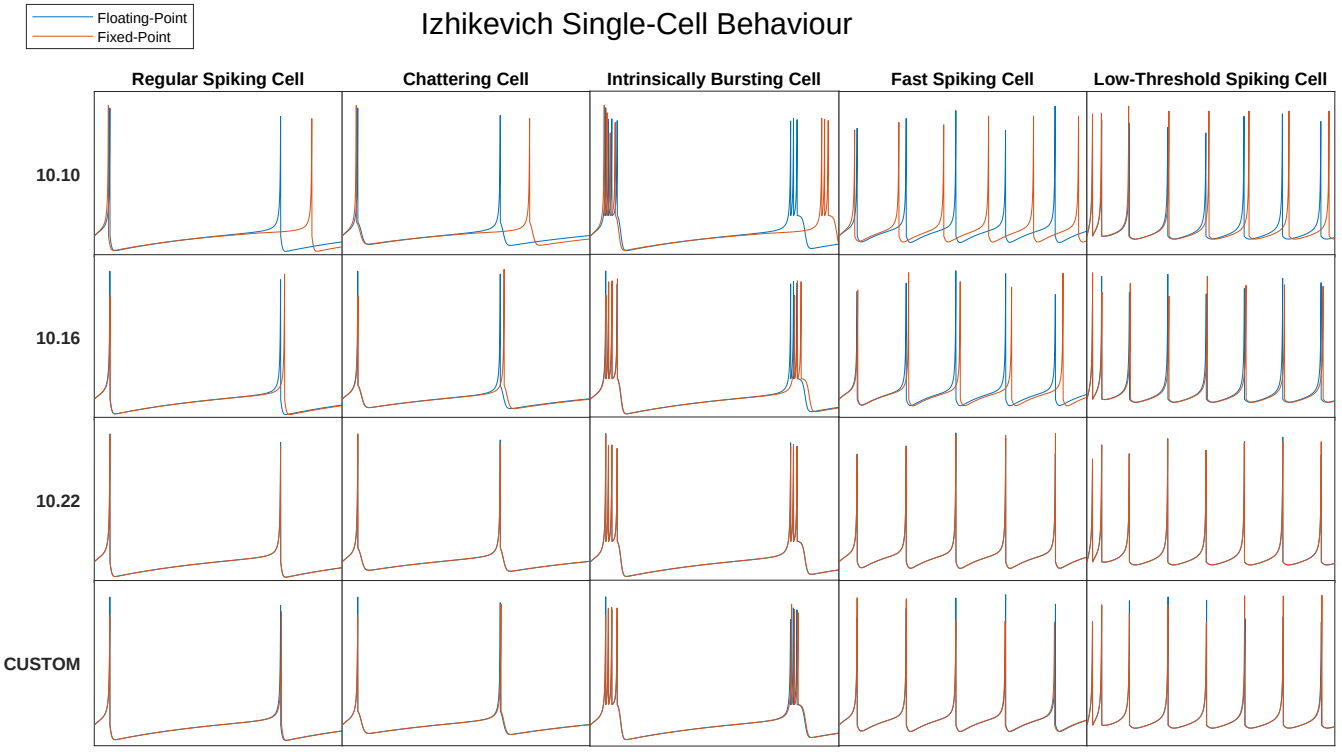

FIGURE 8. Fixed- and floating-point Izhikevich single-cell membrane potential superimposition at the varying of the fixed-point data width. The waveforms have been captured within a time window of 200 ms. On the left is indicated the fixed-point data format. Each column depicts a different kind of neuron, from left to right regular spiking, chattering, intrinsically bursting, fast-spiking, and low-threshold spiking cells.

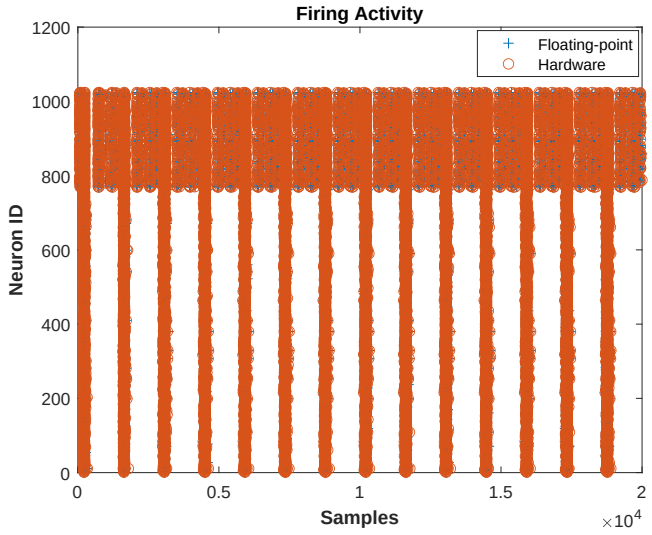

FIGURE 9. Floating-point vs Hardware neural networks firing activity

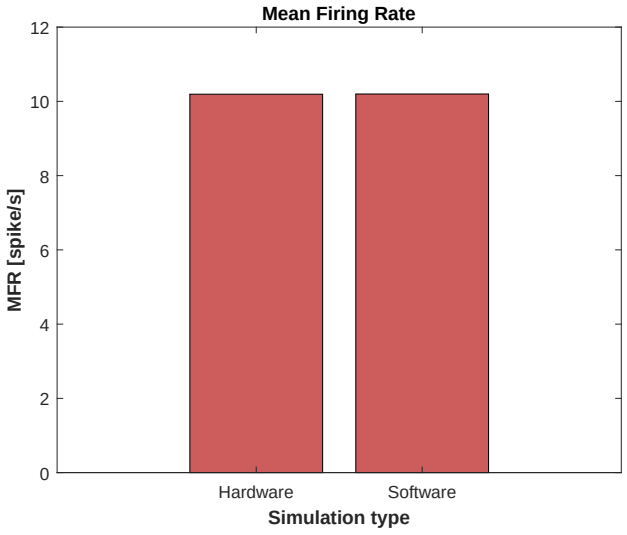

FIGURE 10. Mean firing rate comparison

reference model. Figure 13 shows the Mean Bursting Rate (MBR) of the two networks, the Burst Duration (BD), and the Inter-Burst Interval (IBI). The mean bursting rates are identical, in fact, the numbers of bursts of the two networks are the same. The average burst duration of the hardware network is 149.95 ms, whereas it is 150.55 ms for the reference network. The mean interburst interval of the hardware network is 123.49 ms, the one of the software network is 123.03 ms.

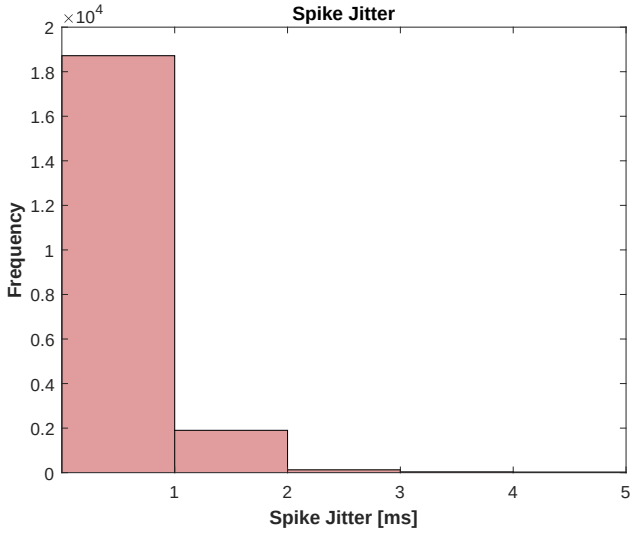

FIGURE 11. Spike time jitter between software and hardware neural networks

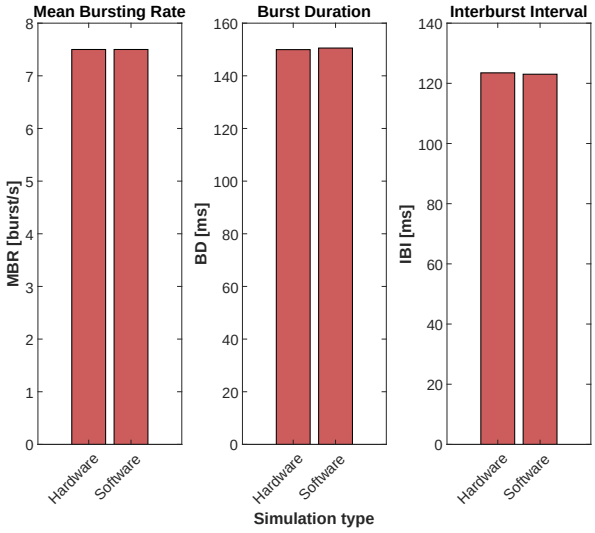

FIGURE 13. Burst metrics comparison

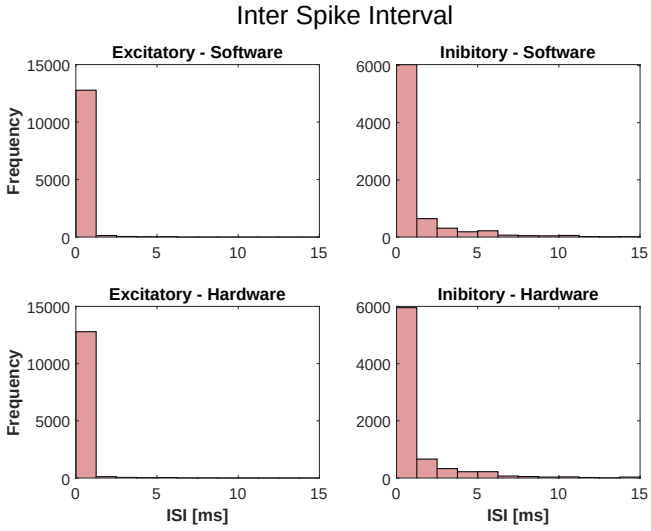

FIGURE 12. Inter spike interval of excitatory and inhibitory neurons

VI. COMPARISON WITH THE STATE OF THE ART

The main characteristics of the FPGA accelerator we target for comparison with our work are shown in Table 6.

The low-power embedded system implementation presented in [17] emulates 800 Leakage Integrate-and-Fire (LIF) neurons and 1.25e4 synapses. Our requirements in terms of DSP and registers are higher, 2.2x more DSP and 1.5x more registers respectively. However, the work [17] uses 2.3x more LUT than the presented system, and emulates only 25.82% of the neurons, and about 0.13% of the synapses of this work. Moreover, the neuron integration frequency is 10 times lower.

The digital system presented in [20] is capable of hosting 12,800 neurons and 20,000 synapses with a time resolution of 1.5 us. The network topology and the resolution used in [20] do not permit a fair comparison with this work. In fact, this work emulates fully-connected neural networks with a time resolution of 0.1 ms, it counts 4.1 times fewer neurons, however, hosting 480 times more synaptic connections than in [20], it allows to emulate arbitrarily connected neural networks.

The system presented in [21] is a fully connected neural network accelerator prototyped into a high-end Virtex-6 FPGA. The time resolution and the neuron model are the same as the presented work. Their digital system can emulate 1,440 neurons and 2.07e6 synapses, which are the 46.48% and the 21.56% of this work's result. Moreover, our work requires only the 43.80% of the LUTs, the 72.49% of the FFs, the 33.29% of the BRAMs, and the 34.31% of the DSPs compared to the implementation in [21].

The low-latency neural network accelerator presented in [22] is implemented on a Stratix-IV device, and can sustain a maximum clock frequency of 250 MHz. This allows to integrate the Izhikevich model in 8 ns and reuse many times the same neuron computational core within their 0.78 ms integrating step, as it happens in the presented work. However, the throughput of our Potential module guarantees an integrated neuron every 6 ns, which is higher than 8 ns, even though our maximum clock frequency is lower. In addition, the neuron interconnections scheme of this work has more biological meaning than in [22], where the neurons can have a single synaptic interconnection. The maximum number of neuron computational cores which fit in the Stratix-IV device is 364, but it is not explicitly declared the maximum number of neurons their architecture can handle in real-time. Moreover, having a single synaptic connection per neuron, the comparison with the other architectures of Table 6 would not be fair.

TABLE 6. Real-time FPGA-based neural network emulators comparison

Work	Year	Family	Part	Neu	Syn	FC	Off-chip mem	Model	Res	Data Format [bit]	LUT	FF	BRAM	DSP
this work	2021	Virtex-6	XC7Z020	3,098	9.6e6	✓	✓	Izhikevich	0.1 ms	custom	24,480	35,158	130.5	140
Gupta [17]	2020	Virtex-6	XC6VLX240T	800	1.25e4	✗	✗	Simp. LIF	1.0 ms	24	56,230	23,238	16	64
Sripad [20]	2018	Kintex-7	XC7K325T	12,800	2.00e4	✗	✗	Izhikevich	1.5 μs	16	148,774	97,824	213	100
Pani [21]	2017	Virtex-6	XC6VLX240T	1,440	2.07e6	✓	✗	Izhikevich	0.1 ms	32	55,884	48,502	392	408
Bandeira [22]	2017	Stratix IV	EP4SGX230	-	-	✗	✗	Izhikevich	0.78 ms	18	84,816*	84,816*	-	-
Luo [24]	2016	Virtex-7	XC7VX485T	101,000	1.00e5	✗	✗	LIF	-	40	268,455	176,424	960	2.304
Ambroise [26]	2013	Virtex-4	SX55	117	1.37e4	✓	✗	Izhikevich	1.0 ms	18	1,598	970	4	1
Han [9]	2020	Kintex-7	XC7Z045	2,842	1.86e6	✗	✓	LIF	-	16	5,381	7,309	40.5	-

*The quantity refers to the number of Adaptive Logic Module of Altera's FPGAs

The bio-realistic cerebellum model presented by Luo et. al [24] emulates 101,000 LIF neurons [20]. So as the presented work, the digital system can be coupled with biological neural networks in closed-loop experiments. Even though the number of neural units is consistently higher compared to this work, it is to be taken into account that the presented system emulates Izhikevich neurons, which are far more computationally expensive than LIF neurons. Moreover, the number of synapses this work supports is 96 times higher than in [24] and this result is obtained by using the 9.11% of the LUT, the 19.93% of the FF, the 13.59% of the BRAM, and the 6.07% of the DSP of the implementation in [24].

The folded architecture presented in [26] permits saving resources by reusing the same processing elements along multiple clock cycles to evaluate the Izhikevich model. They instanced a single neural computational core that can be compared to our Potential module. Their core uses a single DSP, whereas ours makes use of 5 DSP. However, due to this, their architecture requires about x9 more LUT than our Potential module. Moreover, even though in a folded architecture is possible to save registers, by sharing them in time among multiple variables, the folded architecture in [26] still requires x3.5 more registers, probably because most of the registers used by the presented work are the ones embedded into the DSPs. The folded architecture needs 11 clock cycles to integrate the Izhikevich model, whereas our Potential module can integrate a neuron per clock cycle. In addition, the maximum clock frequency of the presented work is higher: 150 MHz against the 85 Mhz of the folded architecture in [26], and summing up, this leads to a throughput 19.4 times higher in favor of the presented architecture.

The image classifier presented in [9] accelerates the execution of spiking neural networks of LIF neurons. Their approach takes advantage of the off-chip DDR memory to store the synaptic weights of the network, as in this work. As a matter of fact, their utilization in terms of BRAMs is lower than most of the other works in Table 6. Moreover, being the LIF model simpler than the Izhikevich model, and counts fewer parameters, their LUTs and FFs utilization is about a fifth of this work, and their BRAM utilization is the 31.03% of the presented work. However, being a hardware accelerator for image classification tasks, even though the number of neurons is almost the same (they emulate about 9% fewer neurons than the presented design), the synapses only serve to connect the neurons among the layers. Therefore, our design counts 5.16 times more synapses than [9]. Finally, not being conceived as a biological-meaningful neural network emulator, [9] is not suited to be interfaced with biological neural networks.

VII. CONCLUSION

We have presented a new method for increasing the synapses count of real-time neural network accelerators. We demonstrated the feasibility of the method by implementing a real-time neural network accelerator counting up to 3,098 neurons and 9.6e6 synapses into a Xilinx Zynq 7020 All-Programmable SoC, with a resolution of 0.1 ms. We showed the DDR memory provides enough storage capability and I/O bandwidth to transfer the synaptic weights in real-time, and that by relying on the DDR, it is possible to overcome the number of synapses that a real-time spiking neural network emulator can store inside its BRAM. This work demonstrates it is possible to emulate highly-connected neural networks in real-time and paves the way to closed-loop experiments addressing biological and artificial neural network interaction, with the aim of increasing the actual comprehension of biological neural network functioning principles and neuroprosthesis development.

We studied how to map the Izhikevich neuron model in fixed-point arithmetic so as to simultaneously find a good map into Xilinx's DSP and LUT, and degrade the accuracy of the network the least possible. We found a difference of 0.06% in the total amount of fired spikes by the proposed fixed-point neural network and the floating-point reference model, with 98.78% of the spikes having a time jitter less than 2 ms.

A long-term purpose for our work is interfacing biological and artificial neural networks in real-time. By relying on the support of a multielectrode array and a neural processing interface, it could be possible to provide input and output data exchange between the networks, paving the way to bio and artificial neural network cooperation.

ACKNOWLEDGMENT

thank you all.

BIBLIOGRAPHY

[1] Robert C Froemke and Yang Dan. "Spike-timing-dependent synaptic modification induced by natural spike trains". In: *Nature* 416.6879 (2002), pp. 433–438.

[2] Marc-Oliver Gewaltig and Markus Diesmann. "NEST (NEural Simulation Tool)". In: *Scholarpedia* 2.4 (2007), p. 1430.

[3] Daria Lisitsa and Anton A. Zhilenkov. "Prospects for the development and application of spiking neural networks". In: *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. 2017, pp. 926–929. DOI: 10.1109/EIConRus.2017.7910708.

[4] Nick Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, Susu Chen, Jennifer Colonell, Richard Gardner, Bill Karsh, Dimitar Kostadinov, Carolina Lopez, Junchol Park, Jan Putzeys, Britton Sauerbrei, and Timothy Harris. "Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings". In: (Nov. 2020). DOI: 10.1101/2020.10.27.358291.

[5] Gianluca Leone, Luigi Raffo, and Paolo Meloni. "ZyON: Enabling Spike Sorting on APSoC-Based Signal Processors for High-Density Microelectrode Arrays". In: *IEEE Access* 8 (2020), pp. 218145–218160. DOI: 10.1109/ACCESS.2020.3042034.

[6] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. "Spiking neural networks hardware implementations and challenges: A survey". In: *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 15.2 (2019), pp. 1–35.

[7] H. A. Swadlow and S. G. Waxman. "Axonal conduction delays". In: *Scholarpedia* 7.6 (2012). revision #125736, p. 1451. DOI: 10.4249/scholarpedia.1451.

[8] E.M. Izhikevich. "Simple model of spiking neurons". In: *IEEE Transactions on Neural Networks* 14.6 (2003), pp. 1569–1572. DOI: 10.1109/TNN.2003.820440.

[9] Jianhui Han, Zhaolin Li, Weimin Zheng, and Youhui Zhang. "Hardware implementation of spiking neural networks on FPGA". In: *Tsinghua Science and Technology* 25.4 (2020), pp. 479–486.

[10] Sathish Panchapakesan, Zhenman Fang, and Nitin Chandrachoodan. "EASpiNN: Effective Automated Spiking Neural Network Evaluation on FPGA". In: *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE. 2020, pp. 242–242.

[11] Sathish Panchapakesan, Zhenman Fang, and Jian Li. "SyncNN: Evaluating and Accelerating Spiking Neural Networks on FPGAs". In: *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE. 2021, pp. 286–293.

[12] Nicholas T Carnevale and Michael L Hines. *The NEURON book*. Cambridge University Press, 2006.

[13] Dan Goodman and Romain Brette. "The Brian simulator". In: *Frontiers in Neuroscience* 3 (2009), p. 26. ISSN: 1662-453X. DOI: 10.3389/neuro.01.026.2009. URL: https://www.frontiersin.org/article/10.3389/neuro.01.026.2009.

[14] Alexander D Rast, Xin Jin, Francesco Galluppi, Luis A Plana, Cameron Patterson, and Steve Furber. "Scalable event-driven native parallel processing: the SpiNNaker neuromimetic system". In: *Proceedings of the 7th ACM international conference on Computing frontiers*. 2010, pp. 21–30.

[15] Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R. Lester, Andrew D. Brown, and Steve B. Furber. "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation". In: *IEEE Journal of Solid-State Circuits* 48.8 (2013), pp. 1943–1953. DOI: 10.1109/JSSC.2013.2259038.

[16] Kit Cheung, Simon R. Schultz, and Wayne Luk. "NeuroFlow: A General Purpose Spiking Neural Network Simulation Platform using Customizable Processors". In: *Frontiers in Neuroscience* 9 (2016), p. 516. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00516. URL: https://www.frontiersin.org/article/10.3389/fnins.2015.00516.

[17] Shikhar Gupta, Arpan Vyas, and Gaurav Trivedi. "FPGA Implementation of Simplified Spiking Neural Network". In: *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294790.

[18] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[19] Henry Markram, Wulfram Gerstner, and Per Jesper Sjöström. "Spike-timing-dependent plasticity: a comprehensive overview". In: *Frontiers in synaptic neuroscience* 4 (2012), p. 2.

[20] Athul Sripad, Giovanny Sanchez, Mireya Zapata, Vito Pirrone, Taho Dorta, Salvatore Cambria, Albert Marti, Karthikeyan Krishnamourthy, and Jordi Madrenas. "SNAVA—A real-time multi-FPGA multi-model spiking neural network simulation architecture". In: *Neural Networks* 97 (2018), pp. 28–45. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2017.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0893608017302150.

[21] Danilo Pani, Paolo Meloni, Giuseppe Tuveri, Francesca Palumbo, Paolo Massobrio, and Luigi Raffo. "An FPGA Platform for Real-Time Simulation of Spiking Neuronal Networks". In: *Frontiers in Neu-*

roscience 11 (2017), p. 90. ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00090. URL: https://www.frontiersin.org/article/10.3389/fnins.2017.00090.

[22] Vitor Bandeira, Vivianne Costa, Guilherme Bontorin, and Ricardo Reis. "Low Latency FPGA Implementation of Izhikevich-Neuron Model". In: Jan. 2017, pp. 210–217. ISBN: 978-3-319-90022-3. DOI: 10.1007/978-3-319-90023-0_17.

[23] Andrew Cassidy and Andreas G Andreou. "Dynamical digital silicon neurons". In: *2008 IEEE Biomedical Circuits and Systems Conference*. IEEE. 2008, pp. 289–292.

[24] Junwen Luo, Graeme Coapes, Terrence Mak, Tadashi Yamazaki, Chung Tin, and Patrick Degenaar. "Real-Time Simulation of Passage-of-Time Encoding in Cerebellum Using a Scalable FPGA-Based System". In: *IEEE Transactions on Biomedical Circuits and Systems* 10.3 (2016), pp. 742–753. DOI: 10.1109/TBCAS.2015.2460232.

[25] Alireza Khodamoradi, Kristof Denolf, and Ryan Kastner. "S2n2: A fpga accelerator for streaming spiking neural networks". In: *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2021, pp. 194–205.

[26] Matthieu Ambroise, Timothee Levi, Yannick Bornat, and Sylvain Saighi. "Biorealistic spiking neural network on FPGA". In: *2013 47th Annual Conference on Information Sciences and Systems (CISS)*. 2013, pp. 1–6. DOI: 10.1109/CISS.2013.6616689.

[27] Xilinx. "AXI DMA v7.1". In: *LogiCORE IP Product Guide* (2019), p. 8.

[28] Michael Hopkins, Mantas Mikaitis, Dave R Lester, and Steve Furber. "Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations". In: *Philosophical Transactions of the Royal Society A* 378.2166 (2020), p. 20190052.

GIANLUCA LEONE is a Ph.D. student in Electronic and Computer Engineering at the University of Cagliari since 2019. He received the B.S. degree in Electronics Engineering from the University of Cagliari, Cagliari, Italy, in 2016, and the M.S. degree in Electronics Engineering from the Politecnico di Torino, Turin, Italy, in 2019. His research activity concerns the development and optimization of hardware accelerators on FPGA.

LUIGI RAFFO is full professor of Electronics at University of Cagliari (ITALY) since 2006. In 1994 he joined the Department of Electrical and Electronic Engineering of University of Cagliari (ITALY). He teaches courses on system design, digital and analog electronics design and processor architectures. Since 2012 he has been Rector's delegate for International Research Projects. He has been coordinator of the Course of Studies in Biomedical Engineering from 2006 to 2012 and from 2017 to 2018. His research topics are in the field of the study, design, development of systems and micro-systems for application where high performance, high efficiency, low power are required. In such a filed he is author of more than 200 scientific papers.

PAOLO MELONI is assistant professor at University of Cagliari since 2012. His research activity is on the development of advanced digital systems, on the application-driven design and programming of multi-core on-chip architectures and FPGAs. He is author of a significant track of international research papers. He teaches Advanced Embedded Systems at the University of Cagliari and is currently scientific coordinator of the ALOHA (www.aloha-h2020.eu) H2020 project.