Biorealistic Spiking Neural Network on FPGA

Matthieu Ambroise, Timothée Levi, Yannick Bornat, and Sylvain Saïghi
Univ. Bordeaux, IMS, UMR 5218, F-33400 Talence, France.

Abstract—**In this paper, we present a digital hardware implementation of a biorealistic spiking neural network composed of 117 Izhikevich neurons. This digital system works in hard real-time, which means that it keeps the same biological time of simulation at the millisecond scale. The Izhikevich neuron implementation requires few resources. The neurons behavior is validated by comparing their firing rate to biological data. The interneuron connections are composed of biorealistic synapses. The architecture of the network implementation allows working on a single computation core. It is freely configurable from an independent-neuron configuration to all-to-all configuration or a mix with several independent small networks. This spiking neural network will be used for the development of a new proof-of-concept Brain Machine Interface, i.e. a neuromorphic chip for neuroprosthesis, which has to replace the functionality of a damaged part of the central nervous system.**

I. Introduction

Millions of people worldwide are affected by neurological disorders which disrupt connections between brain and body causing paralysis or affect cognitive capabilities. Such a number is likely to increase in the next years and current assistive technology is still limited. Since last decades Brain-Machine Interfaces (BMIs) and generally neuroprostheses [1-3] have been object of extensive research and may represent a valid treatment for such disabilities. The development of these devices has and will hopefully have a profound social impact on the quality of life. The realization of such prostheses implies that we know how to interact with neuronal cell assemblies, taking into account the intrinsic spontaneous activity of neuronal networks and understanding how to drive them into a desired state or to produce a specific behavior. The long-term goal of replacing damaged brain areas with artificial devices also requires the development of neuronal network models. They will fit with the recorded electrophysiological patterns and will produce in their turn the correct stimulation patterns for the brain so as to recover the desired function. The hardware set-up that will be used to interface the biological component is a Spiking Neural Network (SNN) system. This SNN implements biologically realistic neural network models, spanning from the electrophysiological properties of one single neuron up to network plasticity rules.

Previous works have already implemented large scale SNN on FPGA [9-12-13]. However, those designs have been realized for computation purposes without taking into account biological real time. Our study describes the development of a hardware neuromorphic device which contains a neural network of 117 biorealistic neurons computed in biological real time. This part constitutes one main point of the Brainbow European project [4]. The main novelty of this paper is the implementation of that neural network composed of 117 cells with one single computation core, which calculates each cell in turns. That implementation uses few resources in the FPGA. In this paper we will first describe the neuron model and how it has been implemented. Then we will focus on the topology of the inhibitory and excitatory synapses that will allow the design of a biorealistic neural network. Finally we will present some results to validate this implementation.

Financial support from the European Community under the grant Brainbow FP7-284772.

II. Neuron Implementation

Our main goal is to propose an architecture using less resource as possible for designing a large scale neural network dedicated to future neuroprostheses. For that, we will focus in this section on the neural cell.

A. *Neuron model*

In order to provide a SNN, the first step is the choice of the neuromimetic mathematical model. Indeed, the mathematical model, based on differential equation, can reproduce a behavior more or less close to biological cell. In order to choose our model, we are looking two criteria: the variety of firing rate dynamics that can be reproduced and the number of equation used for computation.

Regarding those criteria, we can compare all models like the Leaky Integrate and Fire (LIF) [5], the Hodgkin-Huxley model (HH) [6], or the Izhikevich model (IZH) [7-8]. The HH model can reproduce all kind of neurons with a good accuracy in terms of shape of spike or complex firing activities. Their main drawbacks are the great number of parameters and the number of equations. In the Brainbow project, we are mainly interested by excitatory neurons like the Regular Spiking neuron (RS) and inhibitory neurons as the Fast Spiking neuron (FS). With HH model, we have to use 26 and 32 parameters for FS and RS neurons respectively. Furthermore, to simulate a FS neuron, we need four equations: dynamic of i) membrane voltage, ii) potassium channel, iii) sodium channel, and iv) leak current. We add a fifth equation describing the slow potassium channel for the RS neuron. Unlike HH, LIF has only two equations but it cannot simulate more complex firing activities than RS neuron. So the good compromise between LIF and HH is the Izhikevich model. This model is based on two equations and can reproduce many different families of neuron by changing four parameters.

B. *Digital Topology*

The IZH model depends on four parameters, which allow reproducing the spiking and bursting behavior of specific

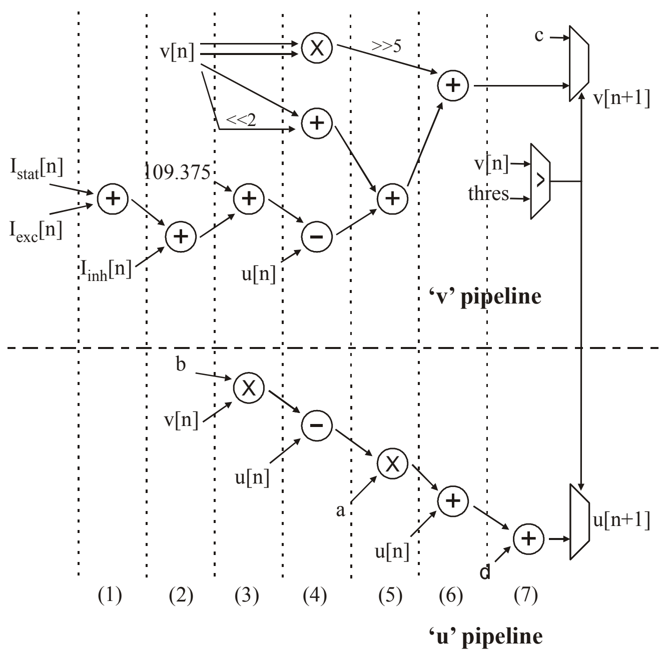

Fig. 1. Architecture of 'u' and 'v' pipelines. The computation cycles are separated by dotted lines.

types of cortical neurons. From a mathematical point of view, the model is described by a two-dimensional system of ordinary differential equations [7]:

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

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

with the after-spike resetting conditions:

$$if \quad v \geq 30\,mV \Rightarrow \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3}$$

In (3), v is the membrane potential of the neuron, u is a membrane recovery variable which takes into account the activation of potassium and inactivation of sodium channels, and I_{Izh} describes the input current from other neurons.

Cassidy et al. [9] suggested an easy implementation of the two upper equations thanks to multiplying equation (1) by 0.78125 in order to replace the coefficients i)0.04 by 1/32 that is a right shift by 5 in register and ii)5 by 4 that is a left shift by 2. This change was motivated by the low number of available 18x18 bits multipliers also known as a limiting resource on FPGA. We also choose to represent our numeric value with 18 bits (10 for the entire part and 8 for the decimal one) for the same reason. Moreover we wished to verify if that bit number is sufficient to reproduce the dynamic of a cell. We compare in Table I the dynamics of 17 bits and 21 bits implementations in comparison with the 18 bits one. The mean relative errors of temporal activity for both cases show that 18 bits is sufficient for obtaining the maximal accuracy.

In order to make the Izhikevich neural network more biorealistic, we split the current I_{Izh} in three currents: I_{stat}, I_{exc}, and I_{inh}. I_{stat} is the biasing current, I_{exc} is the positive contribution due to excitatory synapses and I_{inh} is the negative contribution of inhibitory synapses. Those currents will be detailed in Sec. III. This modification gives (4) where the u coefficient is still 1 thanks to I_{stat} current.

TABLE I. MEAN RELATIVE ERROR IN COMPARISON WITH 18 BITS IMPLEMENTATION

Number of bits	**Mean relative error**
17 bits (9 entire part + 8 decimal part)	178%
21 bits (13 entire part + 8 decimal part)	0%

TABLE II. RESSOURCES USED BY ONE COMPUTATION CORE IN XILINX VIRTEX 4 SX55

Ressources	**Total available**	**Percent utilization**
Slice FF's	49,152	473 (1%)
4-LUT's	49,152	749 (1,5%)
18x18 Multiplier	512	1 (<1%)

$$\frac{dv}{dt} = \frac{1}{32}v^2 + 4v + 109.375 - u + I_{stat} + I_{exc} + I_{inh} \tag{4}$$

Moreover, $\frac{dv}{dt} = \frac{v[n+1] - v[n]}{\Delta t}$ and then we obtain:

$$v[n+1] = \frac{1}{32}v[n]^2 + 5v[n] + 109.375 - u[n] + I_{stat}[n] + I_{exc}[n] + I_{inh}[n] \tag{5}$$

C. Architecture and implementation

The Izhikevich model has a time step of one millisecond, and then our architecture has to compute the value of all parameters within that millisecond.

We implement one neuron on the FPGA board. This neuron will be our neural computation core that will update u and v values of all neurons in the network. So far in our explanation, the kind of a neuron is defined by the four Izhikevich parameters: a, b, c, and d from (2) and (3). Moreover, the state of a neuron is defined by u and v values, and the three currents values. By saving those 9 values in a RAM, we can use them the next millisecond which is the step computation. By extension, the same process can be used for every neuron of the network.

Each u and v computation step is done in parallel, using two pipelines based on the architecture presented in [9]. Our topology is presented in Fig. 1. All parameters from (2), (3), and (4), and u and v values used in the computation are synchronized in one cycle before going through the pipelines (not shown in Fig .1).

In the beginning of the 'v' pipeline, we will add I_{exc}, I_{inh} and I_{stat} in two cycles meanwhile 'u' pipeline is still inactive (that is the step 1 and 2). The currents sum is added to the constant *109.375* and in the same time we do our first multiplication (step 3). By multiplexing operands, we can use the same multiplier for next multiplications that are in

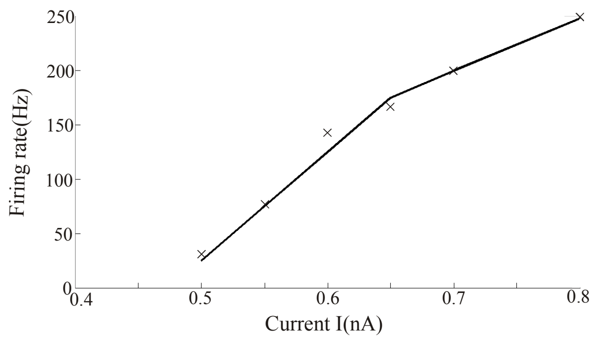

Fig. 2. Comparison of the firing rate vs. stimulation current of a biological (black curve) and a digital (crosses) Fast Spiking neurons.

different computation cycles. In step 4, there is another multiplication in order to obtain v^2. By a simple shift by two bits, we will obtain $4v$ and add it to v. In the same time, two subtractions are done involving u. The step 5 consists on the shift by 5 bits for obtaining $(1/32)v^2$, an addition and the last multiplication. In step 6, we complete both computations of u and v. In the next step, we test v value with the threshold in order to know if the neuron has emitted a spike or not. This test will give the next u and v values of this neuron. Considering resources, a neural computation core uses few FPGA resources and only one multiplier (see Table II).

D. *Results*

We validate our implementation comparing the activity of different kinds of neurons. We present here the FS neuron behavior.

To validate the dynamic of the neurons, we also compare the firing rate of RS neuron to biological data extracted from [10]. Stimulating our digital neuron with different currents, we compare its firing rate with the biological neuron one (see Fig. 2). To make this comparison, we normalize the current I_{Izh} by:

$$I_{Bio} = 0.9091(I_{Izh} - 7) + 0.5.10^{-9} \tag{6}$$

We observe the same frequency vs. stimulating current curves in both cases. We also consider that our implementation is realistic enough for our application.

III. SYNAPSE

A. *Synapse model*

A synapse connects a presynaptic neuron to a postsynaptic one. In case of excitatory (inhibitory) synapse, a positive contribution I_{exc} (negative contribution I_{inh}) is added to the postsynaptic neuron when the presynaptic neuron spikes. Those both kinds of synapse are well known in biology: AMPA and GABA$_a$ for excitatory and inhibitory synapses respectively [10]. The synaptic current rising lasts 1 ms that is the computation time step. Then it will exponentially decrease until zero. The constant time of excitatory decay (τ_{exc}) and of inhibitory decay (τ_{inh}) are different, i.e. are equal to 3 ms and 10 ms respectively. Due to those different constant times, we have to split synaptic currents into two terms for the computation. That explains why we wrote $I_{izh} = I_{stat} + I_{exc} + I_{inh}$ in (4).

B. *Digital topology*

To compute the exponential function we can use different method like Taylor series, but it needs many divisions. A division means using a multiplier and as we explained previously, we want to minimize the use of multiplier. Then we choose another approach in this paper that is the linear regressive method. In this method, we compute the exponential value of the next step using the derivative of the exponential that is also an exponential. So, we have:

$$I_{exp}(t) = e^{\frac{-t}{\tau}} \Rightarrow I_{exp}'(t) = \frac{-1}{\tau} I_{exp}(t) \tag{7}$$

Then

$$I_{exp}(t) = -\tau . I_{exp}'(t) = -\tau . \frac{I_{exp}(t+T) - I_{exp}(t)}{T} \tag{8}$$

$$I_{exp}(t+T) = \left(1 - \frac{T}{\tau}\right) . I_{exp}(t) \tag{9}$$

And with the computation step equal to one millisecond, we obtain:

$$I_{exp}(t+1) = \left(1 - \frac{1}{\tau}\right) . I_{exp}(t) \tag{10}$$

I_{exp}(t) and I_{exp}(t+1) represent the current and next values of I_{exp}. In order to complement the understanding of our network, each next value is saved in the RAM to be used in the next millisecond.

By choosing $\tau = 2^k$ in (10) where k is an integer, we will use a right shift by k bits instead of using a multiplier for the division. However, the time constant is 2^k ms while AMPA and GABA$_a$ have a time constant of 3 ms and 10 ms respectively. To match accurately the biological time constant (τ_{exc} and τ_{inh}) with the digital one ($\tau_{dig} = 2^k$), we compute all values of the exponential decay with τ_{dig} following a virtual time step. Then we will shrink τ_{dig} to τ_{exc} to find the I_{exc} value following τ_{exc}.

Using the ratio τ_{exc}/τ_{dig} as a guide, we can figure out the proper number of repetitions of (10) we have to do. For instance, if τ_{exc} = 3 ms and τ_{dig} = 16 ms, we have to do sixteen times the computation of (10) distributed on three milliseconds. This approximation is more efficient when we equally allocate the sixteen computations in every three other time steps of 1 ms. In our case, we do five computations during the first millisecond, six during the second one, and five in the last one. By extension, for τ_{inh} = 10 ms and τ_{dig} = 16 ms, we have to distribute sixteen computations of (10) in the ten time steps of 1 ms. This method is accurate for $\tau_{exc}/\tau_{dig} < 1$.

C. *Implementation and results*

The synaptic current computation is split in two parts. We separate the update of the currents and the exponential decay computation.

The current update is presented in Fig. 3. We will compute an excitatory or inhibitory contribution depending on the sign of the synaptic current w (positive or negative respectively). In case of excitation, *sign(w)* selects I_{exc}[n] on the upper way. If the presynaptic neuron spikes, the next

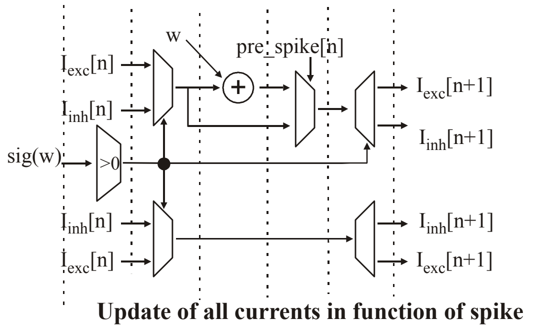

Fig. 3. Updating of excitatory and inhibitory currents.

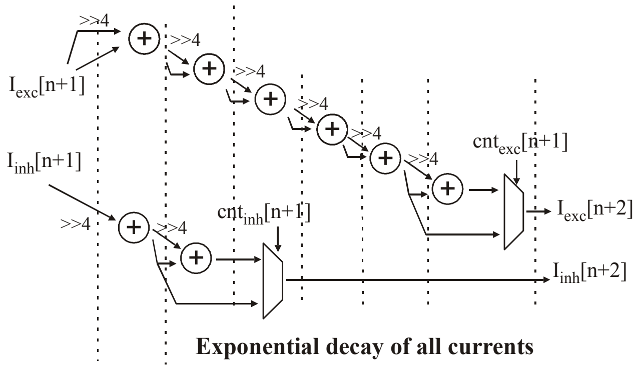

Fig. 4. Exponential decay computation.

value I_{exc}[n+1] will be I_{exc}[n]+w else I_{exc}[n]. Due to excitation case, I_{ihn}[n+1] will stay equal to I_{ihn}[n].

The I_{exc}[n+1] and I_{ihn}[n+1] values are then used for the exponential decay computation in another frame. This computation is done following Sec. III-C, i.e. with shift register by k bits (See Fig. 4). I_{inh} and I_{exc} have different time constants, and then both have different successions of computations in order to obtain their own time constant. Those successions of computations have to be done in one millisecond. As explained above, we allocate the exponential decay computation of I_{exc} on three time step. Then we will use a 2-bit counter for following that. In case of I_{inh}, we will use a 3-bit counter (10 ms/16 ms = 5/8) for allocating the exponential decay computation.

To validate our implementation, we plot on Fig. 5, the measurements of both kinds of synaptic currents. The synaptic current values are sent to the internal DAC of the FPGA and we monitor the output voltage. The upper curve (lower) represents the current of excitatory (inhibitory) synapse. Using the method of the tangent at the origin, we observe that our approximation of the exponential decay computation is sufficient.

IV. NETWORK

A. Towards the network

A biological neural network is composed of Nn neurons and Ns synapses. To define which neuron is connected to which and with which kind of synapse (excitatory or inhibitory), we describe the network by two matrixes: connectivity matrix and synaptic weight matrix respectively. To save RAM, both matrixes are implemented as sparse matrixes. They have Nn lines. The i^{th} line of the connectivity matrix corresponds to the connectivity of the presynaptic neuron N_i to the other neurons. The synapses are notified by the postsynaptic neuron addresses. For example, the connection to the neuron Nj is notified by the number j on the i^{th} line (See Fig. 6). In the worst case, each neuron is connected to itself and all the others, and then we have Nn columns. We end each matrix line by a virtual neuron (address Nn+1). This implementation is not optimal for the worst case, but the gain becomes significant for biologically plausible networks in which the total synapse number is at least four times smaller.

The synaptic weight matrix has the same size as the connectivity matrix, i.e. same number of lines, number of columns and the same position for the virtual neurons. Then there is a direct link between both matrixes. The connectivity between two neurons described by the coordinates (k,l) in the connectivity matrix has the weight written in the case (k,l) of the synaptic weight matrix.

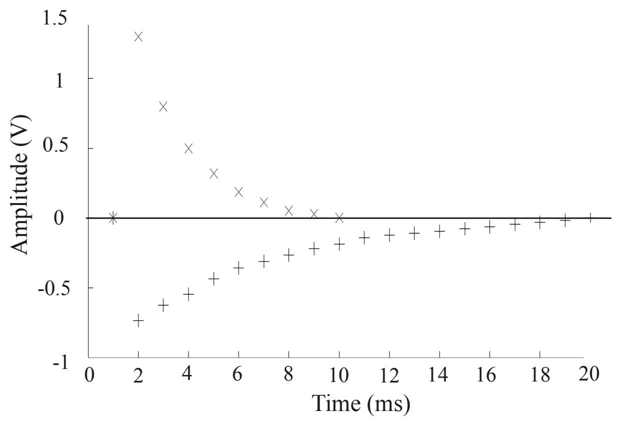

Fig.5. Measurements of one excitatory (x plot) and one inhibitory (+ plot) currents thanks to internal DAC of the FPGA.

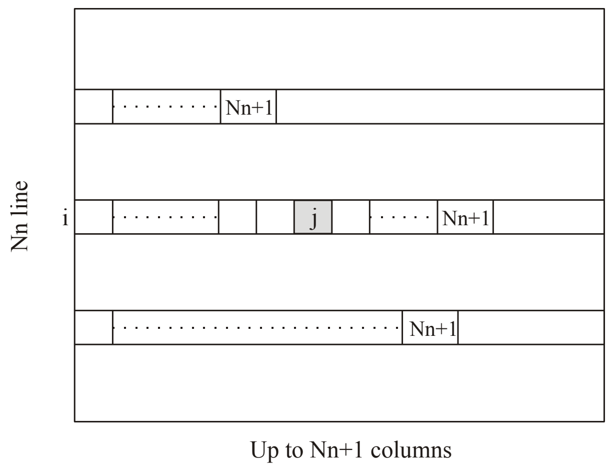

Fig.6. Example of coordinates (i,j) in the matrixes.

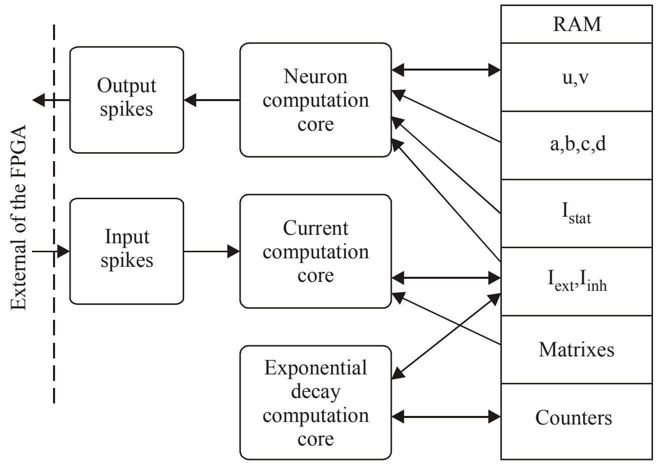

Fig. 7. Global architecture of the spiking neural network.

B. Digital topology

The synaptic current is computed in three steps successively:

- state *EXT*: The first consist of using the block from Fig. 3 for updating the synaptic current. In this case, the pre_spike[n] are external events that are the stimulation from biological neurons in the case of neuroprostheses.
- state *NEUR*: During this step we compute in parallel the neuron membrane ('u' and 'v' from Fig. 1) and all exponential decay values (see Fig. 4).
- state *SYN*: The last steps consist of the update of synaptic current where pre_spike[n] have been computed in state *NEUR*. Those updated current values are used in the state *EXT* during the next cycle.

States *EXT*, *NEUR*, and *SYN* have to be done within the time step that is one millisecond. If the computation of those three states is done before 1 ms, we will wait for the end of the cycle thanks to an IDLE state. Moreover, the block from Fig. 3 will be multiplexed in time for reducing the implementation area.

Our architecture has two main limits: the number of available cycles (*Nc*) in one millisecond and the size of the RAM for saving all parameters. From those two limits, we can write two equations that determine the maximal size of the implementable neural network, in terms of number of neuron (*Nn*) and number of synapses (*Ns*).

During the state *EXT*, all synaptic currents are updated in 6 cycles for each neuron, i.e. 6·Nn cycles. We need 11 cycles for each neuron for the computation of state *NEUR*, i.e. 11·Nn cycles. The synaptic current update during state *SYN* takes 6 cycles by synapse, i.e. 6·Ns. Then, we have:

$$6 \cdot Nn + 11 \cdot Nn + 6 \cdot Ns \leq Nc \tag{11}$$

Our implementation operates at 84.809 MHz, so all computations must be done within 84,809 cycles. In a network where each neuron is connected to all others (all-to-all configuration), Ns is equal to Nn^2 that is the worst case. Then based on (11), we have:

$$17 \cdot Nn + 6 \cdot Nn^2 \leq 84809 \tag{12}$$

With a Xilinx Virtex 4 SX55, our architecture can support an all-to-all network composed of 117 neurons and computed by only one core. For comparison, [9] implements 8 neurons by core.

On the other hand, there is a constraint on the RAM:

- Each neuron has 9 parameters (*a*, *b*, *c*, *d*, *u*, *v*, I_{exc}, I_{inh}, and I_{stat}) on 18 bits then we need 162 bits by neurons (162·Nn).
- Each synapse is determined by a synaptic weight on 18 bits and an address on Na bits for the Nn neurons. Na is equal to log(Nn)/log(2) rounded up. Then for Ns synapses, we have Ns·(18+(log(Nn)/log2)). In the connectivity matrix, our Nn lines end with a virtual element, then we have in addition Nn·(18+(ln(Nn)/ln(2)) bits. Moreover, each neuron has two counters (2-bit one and 3-bit one described in Sec. III-C) for I_{exc} and I_{inh}.

So, we can determine the quantity Q of RAM (in bits) by the following equation:

$$Q(RAM) = 162 \cdot Nn + Ns \cdot \left(18 + \frac{\log(Nn)}{\log(2)}\right) + Nn \cdot \left(18 + \frac{\log(Nn)}{\log(2)}\right) + 5 \cdot Nn \tag{13}$$

And we obtain:

$$Q(RAM) = Nn \cdot \left(185 + \frac{\log(Nn)}{\log(2)}\right) + Ns \cdot \left(18 + \frac{\log(Nn)}{\log(2)}\right) \tag{14}$$

In the worst case, i.e. all-to-all network, we have:

$$Q(RAM) = Nn \cdot \left(185 + \frac{\log(Nn)}{\log(2)}\right) + Nn^2 \cdot \left(18 + \frac{\log(Nn)}{\log(2)}\right) \tag{15}$$

For a 117-neuron SNN with 13689 synapses (all-to-all configuration), we need 354.40 Kb meanwhile the FPGA Virtex4 board has 5120 Kb.

C. Global Scheme

Fig. 7 shows the global architecture of the spiking neural network. That is composed of:

- buffers: input and output spikes;
- computation cores: neuron core, current core, and exponential decay core;
- RAM.

V. Results

Our architecture can be freely configurable from an independent-neuron configuration to all-to-all configuration or a mixed with small networks and independent neurons.

To validate our work in this paper, we implemented a mix of independent neurons and small networks such as:

- N0: independent Chattering neuron (CH) with the parameters: a=0.0156, b=0.01563, c=-50, and d=1.5625;
- N6 to N11: a ring-network of six CH neurons with a=0.02, b=0.2, c=-50, and d=2. All neurons are connected to their neighbors by inhibitory synapses, e.g. N6 to N7 and N11, N7 to N6 and N8, etc.;

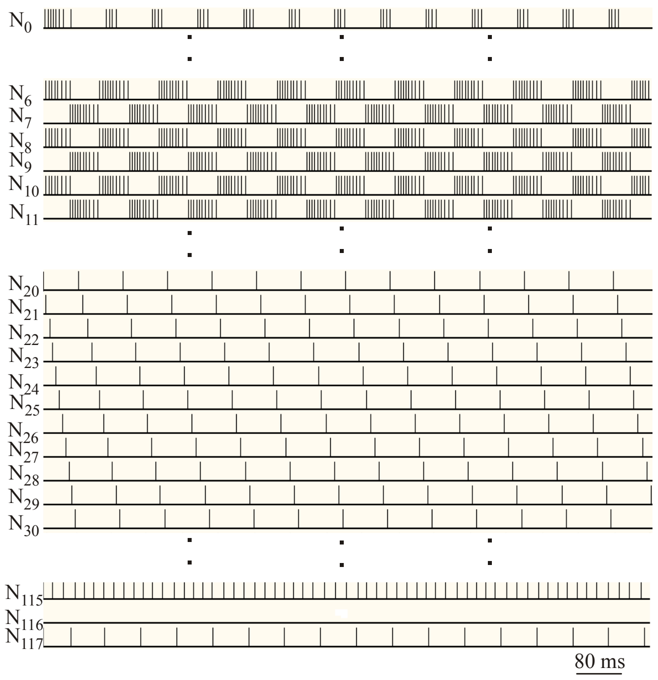

Fig. 8. Measurement of the neural network activity.

- N20 to N30: a loop-network of Regular Spiking neurons (RS) with a=0.02, b=0.2, c=-65, and d=8. The neuron N_i is connected to N_{i+1} by excitatory synapses. The last neuron (30) is connected to the first one (N20) to close the loop. All neurons are biased close to the spontaneous firing activity. We initialized u and v of N20 to spike when the experiment starts;
- N115: an independent Fast Spiking neuron (FS) with a=0.02, b=0.1, c=-65, and d=2;
- N116 and N117: two independent RS neurons (same parameters than N20 to N30) with different stimulation current.
- All the other neurons among the possible 117 ones are not used in this experiment.

Fig. 8 shows the measurements done on the FPGA board thanks to a logic analyzer. Those measurements are similar to results obtained with Matlab code from [7] (not shown here):

- The CH neurons present their typical activity that consists of a burst phase and then a silent phase. About the ring-CH neurons, we also observe two groups that oscillate alternatively due to inhibitory synapses;

- Thanks to initial condition of N20, the latter spikes and then stimulates N21 that launches the next one, etc. We then obtain the typical loop activity;

- The FS neuron presents its typical activity where the firing rate is constant following a constant current stimulation;

- The RS neurons present a slower firing rate than FS neuron with a constant current stimulation.

VI. Conclusion

In this paper, we have presented an architecture that can reproduce with only one computation core based on one single multiplier, a neural network of 117 neurons. Those neurons can reproduce all complex firing activities defined in [7]. Moreover, this architecture allows configuring freely the network, regardless the type of neurons and/or their connectivity.

TABLE III. Ressources used in Xilinx Virtex 4 SX55

Ressources	**Total available**	**Used**
Slice FF's	49,152	970 (2%)
4-LUT's	49,152	1,598 (3.3%)
18x18 Multiplier	512	1 (<1%)
RAM	5120 Kb	144 Kb (2.9%)

Furthermore, our implementation used a few part of resources (see Table III). Then this architecture can be duplicated many times on the same FPGA in order to scale up the size of the neural network. However, the size of the available RAM will be the boundary. For our goal that is the implementation of neuroprostheses, we will scale up our network thanks to adding an external RAM.

References

[1] Nicolelis, M. A.L., and Lebedev, M. A., "Principles of neural ensemble physiology underlying the operation of brain-machine interfaces", *Nature Reviews Neuroscience*, 10, 530-540, 2009.

[2] Hochberg, L.R., Bacher, D., Jarosiewicz, B., Masse, N.Y., Simeral, J.D., Vogel, J., Haddadin, S., Liu, J., Cash, S.S., Van Der Smagt, P., and Donoghue, J.P, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm", *Nature Methods* 485, 372-375, 2012.

[3] Hochberg, L.R., Serruya, M.D., Friehs, G.M., Mukand, J.A., Saleh, M., Caplan, A.H., Branner, A., Chen, D., Penn, R.D., and Donoghue, J.P., "Neuronal ensemble control of prosthetic devices by a human with tetraplegia", *Nature* 442, 164-171, 2006.

[4] www.brainbowproject.eu

[5] Indiveri, G, "Synaptic plasticity and spike-based computation in VLSI networks of integrate-and-fire neurons. Neural Information Processing" , *Letters and Reviews* 11:135-146, 2007.

[6] Hodgkin AL, Huxley AF, "A quantitative description of membrane current and its applications to conduction and excitation in nerve", *The Journal of Physiology* 117:500-544, 1952.

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

[8] Izhikevich, EM, "Which model to use for cortical spiking neurons", IEEE Transactions on Neural Networks 15:1063-1070, 2004.

[9] Cassidy, A., Andreou, A.G , "Dynamical digital silicon neurons", IEEE Biomedical Circuits and Systems Conference, BioCAS 2008, pp.289-292, 20-22 Nov. 2008.

[10] Pospichil, M. , Toledo-Rodriguez, M. , Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., Destexhe A., "Minimal Hodgkin-Huxley type models for different classes of cortical and thalamic neurons", *Biological Cybernetics*, 99:427-441, 2008.

[11] Destexhe,A., Mainen, Z.F. and Sejnowski, T. J, "An efficient method for computing synaptic conductances based on a kinetic model of receptor binding", *Neural Computation*, 1993.

[12] Cassidy, A. S., Andreou, A. G., Georgiou, J., "Design of a one million neuron single FPGA neuromorphic system for real-time multimodal scene analysis", 45th Annual Conference on Information Sciences and Systems (CISS), pp. 1-6, 2011.

[13] Cassidy, A. S., Denham, S. L. , Kanold, P. O., Andreou, A. G., "FPGA based silicon spiking neural array", IEEE Biomedical Circuits and Systems Conference (BioCAS), pp. 75–78, 2007.