Low Latency FPGA Implementation of Izhikevich-Neuron Model

Vitor Bandeira, Vivianne L. Costa, Guilherme Bontorin, Ricardo Reis

▶ **To cite this version:**

Vitor Bandeira, Vivianne L. Costa, Guilherme Bontorin, Ricardo Reis. Low Latency FPGA Implementation of Izhikevich-Neuron Model. 5th International Embedded Systems Symposium (IESS), Nov 2015, Foz do Iguaçu, Brazil. pp.210-217, 10.1007/978-3-319-90023-0_17 . hal-01854162

HAL Id: hal-01854162
https://inria.hal.science/hal-01854162

Submitted on 6 Aug 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Low Latency FPGA Implementation of Izhikevich-Neuron Model

Vitor V. Bandeira[1], Vivianne L. Costa[1,2],
Guilherme Bontorin[1], and Ricardo A. L. Reis[1]

[1] Universidade Federal do Rio Grande do Sul
PGMicro/PPGC – Instituto de Informática
[2] Universidade Federal do Paraná
PPGMNE
{vvbandeira,gbontorin,reis}@inf.ufrgs.br; vlcosta@ufpr.br;

Abstract. The Izhikevich's simple model (ISM) for neural activity presents a good compromise between waveform quality and computational cost. FPGAs (Field Programmable Gate Array) are powerful, flexible, and inexpensive digital hardware that can implement such model. In this paper, we present a highly combinational, low latency implementation of ISM for FPGA. In the absence of official benchmark to compare different implementations, we propose two different metrics to compare the technical literature with our implementation. In this benchmark, we can implement a system that, when compared to the literature, has almost 1.5 times the number of digital neurons (DN), and latency more than 56 times smaller. This shows that our implementation is best suited for hybrid network systems and presents a fair performance for only-artificial networks.

1 Introduction

The human brain has about 10^{11} neurons, and each one can have more than 10^4 synaptic connections with others neurons [8]. As the most inspiring and powerful computing machine we know at present, it is normal to try breaking the code and understand how it works. We believe its computer capacity comes from a three level complexity: (a) the number of adaptable cells, the neurons; (b) the capability of configurable connections, the synapses; and (c) the waveform that is at the same time robust against noise and capable of encoding information, the spike or action potential.

In terms of the waveform, the literature presents various spike models, each one with a respective biological plausibility and computational complexity. The Izhikevich's Simple Model (ISM) [9] presents one of the best compromises between waveform quality and computational cost at the moment. It is composed of a system of two ordinary differential equations of the first order that can be easily digitalized. In terms of capability of configuration connections and the number of cells, it is important to find a hardware that can at the same time be

powerful, flexible, and inexpensive. FPGAs (Field Programmable Gate Array) seem to fill all these requirements as reprogrammable digital circuits.

Some papers describe different implementations of ISM in FPGA [1, 4–6, 10, 11]. They differ from how serial or parallel the computations are implemented and the number of pipeline stages used. Our implementation is highly combinational and present low latency.

No other paper before has proposed any benchmark. To compare our work with others, we propose two different metrics. The first one is neural lattice network. It estimates the maximum number of cells we can simulate in a single hardware.

The second metric we propose is the latency of one neuron. It is the time a variation on the input takes to propagate to the output. This metric has a direct correlation to how parallel an implementation is. Depending on the application, this performance can or cannot be important. Hybrid neural networks, like [2], are an example of systems where such a performance is fundamental. These are systems where the whole network is composed of the real-time communication between an artificial and a living part. Low and reliable latency is fundamental to ensure the real-time communication integrity between networks. As biological neurons have latency slower than digital ones, we expect to reuse the hardware, virtualizing a greater number of neurons.

In Section 2, we review the simple model proposed by Izhikevich and adapt its equation for digital computing. Section 3 presents the hardware implementation of the neuron and shows a lattice network for comparison reasons. Section 4 shows the hardware results of the implementation to compare it to current literature. Section 5 concludes on the potential of this work and comments about future projects.

2 Izhikevich's Simple Model

Table 1. PARAMETERS FOR EACH NEUROCOMPUTATIONAL FEATURE AND INJECTED CURRENT USED IN THE IMPLEMENTATION

		Neural Behaviour					
		Tonic spiking	Phasic spiking	Tonic bursting	Phasic bursting	Mixed mode	Spike-frequency adaptation
Parameters	a*	0.015625	0.015625	0.015625	0.015625	0.015625	0.0078125
	b*	0.15625	0.1953125	0.15625	0.1953125	0.1953125	0.1953125
	c	-65	-65	-50	-55	-55	-65
	d*	4.6875	4.6875	1.56125	0.0390625	3.125	6.25
Input	I*	10.9375	5	11.71875	4.6875	9.375	23.4375

2.1 Equations Model

Ensuring some biological plausibility, the ISM reduces the Hodgkin-Huxley model in two-dimensional system of ordinary differential equations [9]:

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

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

with the auxiliary after-spike resetting:

$$v \geq 30mV \implies \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3}$$

where v is the membrane potential of the neuron and u is the membrane recovery variable, both in millivolts (mV); t is the time in milliseconds (ms); I is the total injected currents in nanoamperes (nA); a, b, c, and d are parameters to set the desired waveform or the neuronal activity.

The parameter a describes the time scale of the recovery variable u. The parameter b represents a sensitivity of the recovery variable u to the subthreshold fluctuations of the membrane potential v. The parameter c describes the after-spike reset value of the membrane potential v. The parameter d represents after-spike reset of the recovery variable u. Different choices of the parameters a, b, c, and d result in different intrinsic firing patterns.

2.2 Change of Variables

In order to facilitate the model implementation in a digital circuit, it is possible to rewrite Equations (1) to (3) as:

$$h\frac{dv}{dt} = \frac{1}{32}v^2 + 3.90625v + 109.375 - u^* + I^* \tag{4}$$

$$h\frac{du^*}{dt} = a^*(b^*v - u^*) \tag{5}$$

$$v \geq 30mV \implies \begin{cases} v \leftarrow c \\ u^* \leftarrow u^* + d \end{cases} \tag{6}$$

where $h = 0.78125$, $u^* = hu$, and $I^* = hI$; the parameters a, b, and d are replaced by $a*$, $b*$, and $d*$, respectively, each one also multiplied by h. This transformation is suggested in [4] and [1], but both neglect the factor h.

The new system of differential Equations (4) to (6) ensures the same behavior of Equations (1) to (3). We can solve by Euler's Method [3], a numerical method of the first order, which produces accurate results. This approach results on:

$$v_{n+1} = v_n + \Delta t \left[\frac{1}{32}v_n^2 + 3.90625v_n + 109.375 - u_n^* + I_n^*\right] = v_n + \Delta t.kv \tag{7}$$

$$u^*_{n+1} = u^*_n + \Delta t\,[a^*(b^* v_n - u^*_n)] = u^*_n + \Delta t.ku \tag{8}$$

where Δt is the time increment of the Euler's Method. Moreover, in Equation (7) we also approximate $3.90625v \approx 4v$ [4]. kv and ku are used further in the implementation and they represent the variation for each iteration.

The parameters in Table 1 are adapted from the original publication [9] considering the factor h, and it depends on the type of the simulated neuron. The choice of parameters is beyond the scope of this paper, as it is a current research topic. The input current is set to an appropriated input to reveal a realistic behavior.

In the next section, we show the methodology for ISM implementation on FPGA.

3 Neuron Implementation

3.1 One Neuron

For our implementation, we use combinational logic for the most of the circuit. The circuitry is as parallel as possible, optimized for latency rather than the area with no reuse of any adder or multiplier. Figure 1 presents a single neuron, and Figure 2 the computation for the new value of *v, u** as well as the activity log. For the calculation of the next *v and u**, we opted for two parallel operations, one for the case with a spike and other without a spike and select between them afterward.

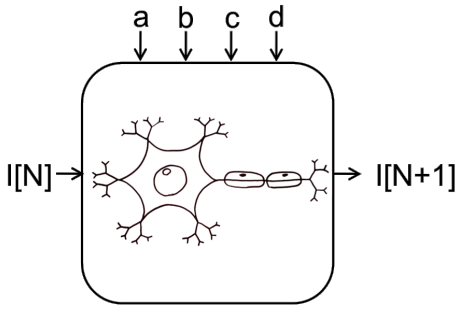

Fig. 1. Implementation of One Neuron

Each neuron receives the parameters (*a*, b*, c, d**) from a top module and stores locally the initial values for *v and u**. We use an 18-bit fixed point two's complement representation: 1 sign bit, 9 bits for the integer part and 8 bits for the fractionary part. This representation is better suited for digital implementations than floating point, and 18 bits uses more efficiently the available hardware without compromising the accuracy as presented on [1].

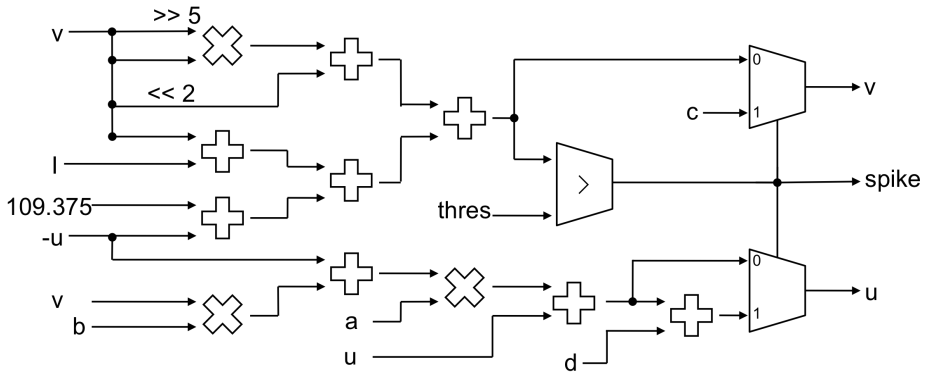

Fig. 2. Schematics of one neuron, the operations to compute Equations (4) to (6). All variables and parameters in this figure already account for the variable change presented in Section 2.

The initial values of v and u^* are, respectively -70 mV and -15.63 mV. The time incremental is $\Delta t = h = 0.78125$ ms (milliseconds). The parameters and injected currents are exhibited in Table 1.

3.2 Network for Tests and Metrics

We have chosen a lattice network: one neuron is directly connected to the next, by $I_{[N]}$ and $I_{[N+1]}$, Figure 3. Even though this has low biological meaning, it can be used estimate the maximal number of neurons that can be implemented on a single FPGA chip.

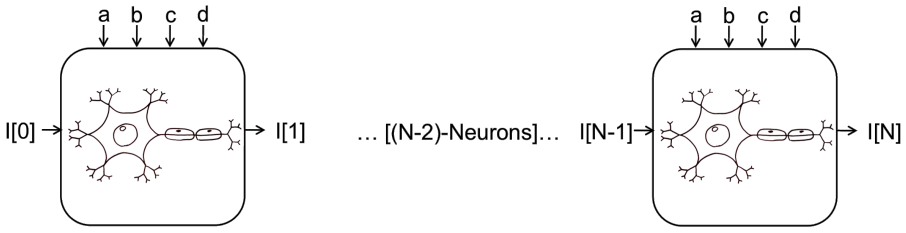

Fig. 3. Schematics of the lattice network used.

We use an Altera's DE4 Board (EP4SGX230KF40C2) to estimate the number of cells that we could implement, and measure latency. Figures 4 and 5 represent about 200 ms in biological time and about 1.8 microseconds in FPGA with a 250-MHz clock. The maximum and minimum tensions are, respectively, +32 mV and -70 mV. Table 1 contains the parameters used for the implementa-

tion as well as the input current. These results were obtained with the SignalTap II, provided in the Quartus II software, and will be presented in the next section.

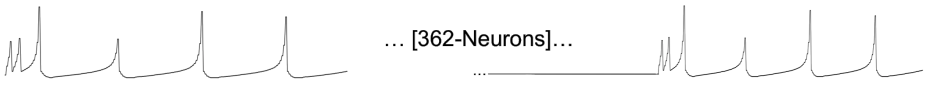

Fig. 4. Simulation results of the lattice network used.

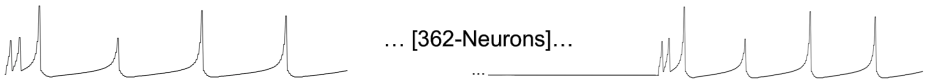

Fig. 5. Measurement of the lattice network used.

4 Results

Table 2. COMPARISON OF OUR IMPLEMENTATION WITH LITERATURE

Ref	Digital Neurons	HW Use		Time Performance			Representation (bits)		FPGA	
		FF	LUT	Clock (MHz)	Pipeline Stages	Latency (ns)	Total	Integer Decimal	Vendor	Family
[5]	32	28%	44%	50	0	320	-		Xilinx	Spartan
[4]		64%	78%	40	5	150	18	10.8		
[1]	1	1%	1,5%	84.81	7	94.33				Virtex-4
[10]	25	79%		198	23	121.21	44	32.12		
[6]	32	32%	36%	110.47	6	63.37	18	9.8		Virtex-5
[11]	256	3.39%		307	96	315.96	32	-		
		7.04%		214	147	453.27	64			
This	364	93%		250	0	8	18	10.8	Altera	Stratix IV

Figure 4 shows the simulation of our Verilog description of lattice network using ModelSim^{TM}. Figure 5 shows the measurement of the lattice network on the FPGA, with only the first and last neuron being presented. The data for Figure 5 was obtained using the SignalTap II tool from Quartus II Software. This tool implements a circuit on the FPGA that acquires data directly on the logic circuits and then send it through a JTAG connection to the computer. The data can be displayed and handled on SignalTap II or exported to other

software. Only one neuron activity is shown, but we have tested the activity of the six neurons presented on Table 1.

With a lattice network, we can fit 364 digital neurons (DN) on an FPGA. This is 1.5 times more than previous from the literature, [11], which presented 256 DN (Table 2). Our estimative is from a lattice network, but this number alone is expressive. And it is important to estimate how many realistic DN we can implement in a realistic physical network.

We consider that there is no coherence of network implementations and available data in the literature, since each paper implements a different network. Therefore, we do not compare values for virtual neurons at the network level. Indeed, we have not found any paper comparing it either.

There is much evidence of biological data from experiments indicating that the information in brain structures can be coded, among other ways, in the time interval between spikes [12]. Because of this we have considered paramount to our implementation to have a high precision on the spike timestamp, which is the instant that the spike occurred. That is achieved with low latency and reliable system.

The latency is the time that the cell takes to provide a valid output from a variation on the input. We have shown that our latency (8 ns) is more than 56 times smaller than the literature, the best comparison being with [11] (453.27 ns), Table 2.

The pipelines presented in the literature do not show a parallel load, causing to have a bigger latency. And also it implies an approximation of the spike timestamp as high as the pipeline extension. We suppose that such an approximation do not interfere with their applications [1, 4, 6, 10, 11], but the same cannot be said to all applications.

5 Conclusion

In this paper, we presented a highly-combinational low-latency implementation for Izhikevich's Neuron Model in FPGA. This approach is better suited for hybrid network applications as it has the best latency in the literature, Table 2. Some other implementations can be better suited for emulation of networks purely artificial with less precision in spike timestamp, as they use fewer resources than ours.

We could also implement more DN in a single FPGA board than the literature when we consider our lattice network. Although as in many cases with bioinspired circuits, these implementations are very particular, and a fair comparison between two different networks is near impossible.

Future works include: (a) to implement a network with more biological meaning; (b) to reuse logic blocks and to implement a pipeline for some calculations to achieve a better speed without compromising latency; (c) to use precomputed values in auxiliary shared memory to reduce computation time and latency; and (d) to explore the parallelism technique for multiple virtual neurons, increasing the maximum size of a network in a single FPGA chip.

As our application is to study and interface with natural living neural networks, the FPGA implementation is preferable for it has easier configurability, reconfigurability, and test. Other implementations such as artificial networks implemented on a full-custom analog [7], or digital integrated circuits may be interesting to implement the short-term objectives (a) through (d) are achieved. Such long-term implementation can improve power consumption, timing performance, and area occupation at the expense of configurability.

Acknowledgement

This work is funded by the following agencies: Federal Agency for Support and Evaluation of Higher Education of Brazil (CAPES), the National Council for Technological and Scientific Development (CNPq), and the Foundation for Research of the State of Rio Grande do Sul (FAPERGS). The authors thank the Macnica-DHW Ltda for the FPGAs boards and technical support.

References

1. Ambroise, M., Levi, T., Bornat, Y., Saighi, S.: Biorealistic spiking neural network on fpga. In: Information Sciences and Systems (CISS), 2013 47th Annual Conference on. pp. 1–6 (March 2013)
2. Bontorin, G., Renaud, S., Garenne, A., Alvado, L., Le Masson, G., Tomas, J.: A real-time closed-loop setup for hybrid neural networks. In: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE. pp. 3004–3007 (Aug 2007)
3. Burden, R.L., Faires, J.D.: Numerical Analysis. Brooks/Cole Publishing Company, Boston, 9 edn. (2011)
4. Cassidy, A., Andreou, A.: Dynamical digital silicon neurons. In: Biomedical Circuits and Systems Conference, 2008. BioCAS 2008. IEEE. pp. 289–292 (Nov 2008)
5. Cassidy, A., Denham, S., Kanold, P., Andreou, A.: Fpga based silicon spiking neural array. In: Biomedical Circuits and Systems Conference, 2007. BIOCAS 2007. IEEE. pp. 75–78 (Nov 2007)
6. Cheung, K., Schultz, S., Leong, P.: A parallel spiking neural network simulator. In: Field-Programmable Technology, 2009. FPT 2009. International Conference on. pp. 247–254 (Dec 2009)
7. Indiveri, G., Horiuchi, T.K.: Frontiers in neuromorphic engineering. Frontiers in neuroscience 5 (2011)
8. Izhikevich, E.M.: Neural Excitability, Spiking and Bursting. International Journal of Bifurcations and Chaos 10(6), 1171–1266 (2000)
9. Izhikevich, E.M.: Simple model of spiking neurons. IEEE 14, 1569– 1572 (2003)
10. Rice, K.L., Bhuiyan, M., Taha, T., Vutsinas, C.N., Smith, M.: Fpga implementation of izhikevich spiking neural networks for character recognition. In: Reconfigurable Computing and FPGAs, 2009. ReConFig '09. International Conference on. pp. 451–456 (Dec 2009)
11. Thomas, D.B., Luk, W.: Fpga accelerated simulation of biologically plausible spiking neural networks. In: Pocek, K.L., Buell, D.A. (eds.) FCCM. pp. 45–52. IEEE Computer Society (2009)

12. Wennberg, R., Velazquez, J.L.P.: Coordinated Activity in the Brain : Measurements and Relevance to Brain Function and Behavior. Springer-Verlag New York, New York, NY (2009)