Biologically compatible neural networks with reconfigurable hardware

Juan Carlos Moctezuma[1,2], Joseph P. McGeehan[1] and Jose Luis Nunez-Yanez[2]
[1]*CCR Group, Electronic Engineering, University of Bristol, Bristol, U.K*
[2]*Micro Group, Electronic Engineering & Computer Science, University of Bristol, Bristol, U.K.*
eejcme@bristol.ac.uk, j.l.nunez-yanez@bristol.ac.uk, j.p.McGeehan@bristol.ac.uk

Keywords: FPGA neuro-simulator, synaptic integration, biophysically accurate model, hardware neuro modelling, Traub model, Pinsky-Rinzel model, biological compatible neurons.

Abstract: This paper presents a reconfigurable hardware neuro-simulator specifically designed to emulate biophysically accurate and biologically compatible neural networks. The platform is based on FPGA technology which is used to create real-time custom neuroprocessors with floating point accuracy and a novel hybrid time-event driven architecture for synaptic integration. Through a series of experiments the dynamics of the neuroprocessors are evaluated and compared with real neuron responses. The problem of interconnecting neurons with individual synapses is tackled with a novel synaptic architecture where all incoming synapses are merged efficiently in one single accumulative process without losing biological information. The case studies demonstrate the suitability of conductance-based models and FPGA platforms to simulate living organisms' behaviour in a biological compatible context.

1 INTRODUCTION

There are two major research areas for modelling neural systems: based on simplified neuron models (behaviour-based model) in a large neural network or using realistic neuron models (also called conductance-based models) in a size-constraint neural network. In simplified models, some biological information is sacrificed in order to save computational resources and build large neural networks e.g. several thousands or millions of neurons. On the other hand, realistic models treat the neuron aspects from the point of view of ionic conductances and they can emulate a great variety of complex dynamics at the ionic level that are present in real neurons. The main goal of this work is the design of a suitable hardware platform to support the simulation of biologically accurate neural systems. The paper also shows the importance of bio-realistic modelling in size-constraint neural systems using bio-compatible neuron models. In nature, we have many examples of such small neural networks; for instance the nervous system behaviour of the Caenorhabditis elegans worm has been investigated and it has around 300 neurons [1], song recognition and sound localization have been analysed in the auditory nervous system of the grasshopper [2]; even the question of "how powerful a single neuron is?" has been investigated by several researchers [3]. Therefore the study of such size-constrain systems is crucial to understand the nervous system in many biological organisms presented in nature and the realization of simulation platforms to support such systems is needed.

Conductance-based modelling is suitable to emulate realistic neuron models since it is consistent with the dynamics of real neurons and it can incorporate as much cellular detail as it is needed. One of the most important features of these models is that they are biophysically compatible and hence neuroscientists, biologists, psychologists can, at certain level, study their properties and co-relate directly parameters with their biological counterparts. With these models we can explore the dynamics of neurons at cellular level [4]. However the main drawback is that they are highly computational-intensive limiting simulations to small nervous systems or single neurons.

In this work, we take the two-compartment Pinsky-Rinzel (P-R) model as the keystone of the bio-realistic neuron and develop a neuro-simulator able to generate actions potentials for several case studies with real applications in Biology. The suitability of the neuro-simulator is validated successfully through a series of experiments. In this context, the main contributions of this work are:

1. The neuron dynamic's study and characterization of the single and two-compartment P-R neuron model for the hardware implementation.
2. A neuroprocessor architecture with better performance and latency/area reduction using hardware design methodologies including: efficient exponential operation and novel event-time-driven for the synaptic integration.
3. A number of case studies that show the advantages of conductance-based models to simulate accurate biological neural networks and the suitability of FPGA technology for this task.
4. The full platform including the VHDL source code and case studies has been made available open-source.

The present paper is organized as follows: in section 2 a review of the main related work done in neuro-simulation is presented; in section 3 a brief introduction to P-R representation and numerical methods applied to the hardware system are introduced; then the hardware design methods including a novel synaptic integration architecture are described in section 4; in section 5 the details of the complete FPGA-based neuro simulator are presented; to validate the simulator section 6 develops a number of case studies which are analysed in a biological meaningful context; finally in section 7 the conclusions and future work are discussed.

2 Review of previous work

Several neural simulators have been developed to emulate the activities of living brain cells and tissues on different scientific computing platforms. In practice, there are different kind of neuro-simulators platforms; among the most important are software tools, parallel computing platforms, dedicated analogue and VLSI chips, GPU-based platforms and FPGA-based platforms.

Software-based neuro-simulators offer good flexibility, they allow to incorporate custom models and to use a great variety of built-in models, from reduced integrate-and-fire to complex multi-compartment conductance-based models. Also there are good options for conductance-based modelling which permits biophysically realistic neuron simulations such as NEURON, CNS and GENESIS [5]. In addition, the parallel computing solutions (accelerated co-simulation through specific hardware devices connected to a PC) overcome the limitations that CPU have. However the diversity in programming languages that each simulator has and the additional effort to incorporate new tools for parallel computing make these solutions complex to manage and expensive to support. Additionally the operating system dependence with the use of a PC and size of such simulators makes difficult their portability to applications like brain implantable solutions on humans and animals.

Large scale super computers have emerged as an attractive option to accelerate simulation tools. These solutions offer very good performance at any level of simulation, including biophysical realistic models and they provide good extension-processing alternatives to popular simulators. Normally, this kind of solutions focus on large-scale neural networks and have huge network architectures. The size of these machines is not practical for applications that require physical portability like neuro-implants. The objective tends to be the emulation of either the whole or big parts of the human brain. The associated huge complexity means that the models are not biophysically realistic and instead they are based on biological plausible models such Izhikevich [6].

Another interesting hardware simulation approach are GPU-based architectures, they are primary intended for large neural networks and in consequence they use HH single compartment or Izhikevich approach as their biological realistic model and forward-euler as their numerical method, which offers acceptable stable solutions for models that not require many physiology details at ion-channel level. The research in neural networks on GPUs focuses its attention mainly on speed up against CPUs. However this technology has some drawbacks: GPUs underperforms when either a significant overhead in data dependency calculations is incurred or the algorithms is not sufficiently parallel [10]. The memory-intensive neural networks calculations requires to store data constantly; the GPUs have an acceptable memory bandwidth to off-chip memory, but is small compared with the several megabytes per second transfer available on-chip in the FPGA devices counterparts. This is particularly important for accessing neuron parameters and synaptic weights during neuron output processes [7]. In addition, a dependence of PC embedded PCI-rack to plug the GPU card in order to exchange data, which is not suitable for portability and small size chip designs. Also the lack of flexibility in terms of hardware architecture manipulation; due to their fixed architecture, it is necessary to fit the algorithm to the GPU architecture.

FPGA technology offers interesting features that makes it an attractive solution for neural simulations. Flexibility, is reflected on the reconfiguration capacity and its ability to implement a large variety of custom architectures according to problem specifications. In FPGA-based neural networks simulators we can see this flexibility reflected in several works [8]. Parallelism is another important feature on FPGA neuro simulation, the capacity to implement several processing modules working at the same time and the possibility of selecting between area, cost and processing speed is an ideal feature for most applications, including neural networks development. The availability of DSP dedicated blocks in modern FPGAs makes possible floating-point operations and efficient exponential algorithm implementations which are crucial for several numerical methods used in neuron modelling [9]. However FPGA technology pays significant area and performance penalty for being reconfigurable and to offer these advantages, resulting in limited size neural networks for biophysically realistic conductance-based models. However, distributed arithmetic, look-up tables techniques, dynamic adaptive memories and multiplexed architectures are some of the propose solutions to overcome this issue [10]. In addition, the possibility of direct-wired connection to external signals reduces significantly the data transfer for interfacing with living tissue experiments compared with GPUs alternatives.

3 The conductance-based hardware neuron

The simplified two-compartment version P-R [11] of the original 19-compartment Traub model [12] is a suitable solution to represent the physiological response of a neuron and represent a good trade-off between area-cost and computational complexity, this model can reproduce more complex burst patterns than the H-H model cannot [8]. The P-R model takes into account information about calcium ion channel Ca^{2+}. Calcium dynamic is another important element in the chemical and electrical behaviour in the neuron. The model includes two parts: a soma-like, which has the Na^{+} and K^{+} activated currents; and a distal dendrite-like, where Ca^{2+} activated and potassium Ca^{2+}-dependent currents are considered.

3.1. Neuron components

In order to represent the two-compartment P-R model, three interconnected hardware modules were developed that represent the soma, dendrite and synapse parts of the neuron. These three neuroprocessors are shown in Figure 1 with a general schematic architecture that represents the hardware-based P-R neuron . Every module was written in VHDL using floating-point operation modules (FPALUs), state machines (FSMCs), logic control and internal RAMs in order to get all outputs and internal results needed. The modules have associated a dual-port RAM (DPRAM) in order to configure and control relevant parameters such as maximum conductance, input current, ions equilibrium potentials, geometric parameters, time step, configuration of state variables, etc.

The soma and dendrite modules exchange their compartment voltages with each other, meanwhile the synapse module receives spiking information from incoming neurons and generates the necessary synaptic conductances (see equation 8) that affect the action potential propagation from the neuron.

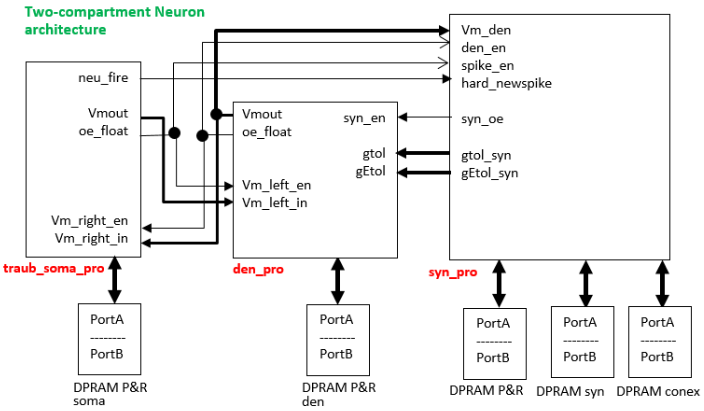


Fig 1. Hardware neuron architecture divided in three modules: soma, dendrite and synapse.

The soma-dendrite modules follows the original conductance-based model from Pinsky and Rinzel work [11]; however in order to give it more accurate geometric information, cable equation and multi-compartment theories are used and hence six different terms are considered for every compartment: the capacitive membrane; the total ionic-channel currents I_m ; the injected current I_e ; and three terms that relate the voltages of current compartment V_j and adjacent left and right segments, V_{j+1} and V_{j-1} respectively.

$$C_m V'_j = I_e - I_m + \frac{a}{2\mathrm{r}_a(\ x)^2}\left(V_{j-1} - 2V_j + V_{j+1}\right) \qquad (1)$$

Where r_a and C_m are the specific axial resistivity in unit of KΩ-cm and the specific membrane capacitance in unit of μF/cm^2 respectively. I_e and I_m are the applied and ionic channel currents per unit area expressed in μA/cm^2. The cylindrical segment (compartment) having radius *a* and length *dx*=*Δx* (in units of cm) provides the morphological information. *V* is the membrane voltage in mV.

In order to solve equation (1), a convenient numerical method is required. The exponential Euler method offers a good trade-off between stability and computational complexity, several well-known neuro-simulators like NEURON use this method to integrate the membrane voltage equation. Rewriting the cell equation (1) in a more convenient way, we have:

$$C_m \cdot V_j' = A - (B \cdot V_j)$$

Where,

(2)

$$A = \psi\ + I_e + \frac{a}{2r_a\ x^2}\left(V_{j+1} + V_{j-1}\right)$$

$$B = \frac{a}{r_a \Delta x^2} + \psi_{Gtot}$$

Where $\psi_{¿}$, ψ_{Gtot} are the weighted averages of all ionic channels conductances. These two terms change according to the type of compartment (soma or dendrite) as follows:

$$Soma\psi_{Gtot} = Gna\ m_{\infty}^{2}\ h + G_{k_{DR}}\ p + Gleak \tag{3}$$

$$Soma\psi\ = Gna\ m_{\infty}^{2}\ h * E_{na} + G_{k_{DR}}\ p\ E_k + Gleak\ E_{leak} \tag{4}$$

$$Den\psi_{Gtol} = g_{ca}\ s^2 + g_{K_{AHP}}\ q + g_{k_c}\ c\ X(Ca) + Syn\psi_{Gtot} \tag{5}$$

$$Den\psi\ = g_{ca}\ s^2\ E_{ca} + g_{K_{AHP}}\ q\ E_k + g_{k_c}\ c\ X(Ca)\ E_k + Syn\psi \tag{6}$$

$$Syn\psi_{Gtot} = g_{ampa}\ r + g_{gaba}\ u + g_{nmda}\ z\ \beta(V) \tag{7}$$

$$Syn\psi\ = g_{ampa}\ r\ E_{ampa} + g_{gaba}\ u\ E_{gaba} + g_{nmda}\ z\ B(V)\ E_{nmda} \tag{8}$$

The transition between the ion gates close and open states is controlled by the state variables (i.e. *m,h,p,s,q,c,r,u* and *z*) which are defined by the *first kinetic formula* [13]. Since it is assumed that the synaptic connections are located in the dendrite compartment, the synaptic conductances $Syn\,\psi_{Gtot}$ and $Syn\,\psi_{¿}$ are summed to the dendrite terms. Later, the details of synaptic integration method are given.

Then applying the exponential Euler solution to equation (2), we can obtain the explicit solution for every time step given by the equation:

$$V_{n+1} = \frac{A}{B} + \left(V_n - \frac{A}{B}\right)\ e^{-B*dt/Cm} \tag{9}$$

3.2. Efficient Exponential Optimization

As we can see in equation (9), every neuroprocessor module needs to calculate four arithmetic operations: add/sub, multiplication, division and exponential at every time step. The most timing and area consuming arithmetic process is the exponential operation. Indeed, previous work [8] has shown that the latencies for the CORDIC-exponential implementation is around 180 clock cycles compared with the 16 cycles of the second most time consuming operation. To tackle this issue, it is necessary to propose efficient numerical techniques to reduce latency and area cost without affecting the performance and dynamics of neuron behaviour.

Two possible exponential implementation alternatives have been analysed. The first approach is based on a piece-wise exponential Look-Up-Table (PW-LUT). This technique has been successfully used on several neural simulators [14]. Previous research has found that for the specific purpose of neuroprocessors, a piece-wise methodology offers the best performance rather than use a single domain LUT (see table 1). The main idea of PW-LUT is to split the argument range in several regions, given a desired resolution (LUT depth) to "zones of interest", such as the range [-1 0], where it is crucial to perform the numerical method exponential. Such ranges are based on statistical observations of different exponential responses for the neuroprocessors. The second approach is the FLOating-POint-COres (FLOPOCO) project [15], which is an open-source C++ framework that can generate arithmetic cores. It generates a synthesizable VHDL code from a command-line interface where operator parameters can be configured. The main purpose is to explore how FPGA flexibility can be exploited for floating-point arithmetic. Table 1 summarizes the results for the four exponential implementations, where we can observe the scopes and limitations for each approach. Two important things to notice: since exponential piece-wise LUT (expPW-LUT) is divided in three regions, every measure result is calculated for each range; and for FP formats there are two different types of results: one corresponds to the actual maximum rate supported and the other corresponds to the specific range [-16 16] in order to make a fair comparison with the LUT format exponentials. FLOPOCO exponential (FloPoCo) presents the best trade-off with respect to latency and accuracy hence it has been selected to be used for the neuroprocessors. On the other hand, if the objective is area reduction the exponential LUT-based approaches [16] use pre-calculated tables and fewer FPGA resources than the FLOPOCO and CORDIC methods with a reduction of 80% and 170% respectively. Overall, the three considered versions improve over 95% the latency of full CORDIC version and these versions present a better accuracy performance in the neuron output.

Table 1. Measurement of the data format representation for the three types of exponential implementation. FP: floating-point.

	FP_ CORDIC	FP_ FLOPOCO	exp SINGLE-LUT	exp PW-LUT
Latency	180	3	2	3
Precision	FP 32-bit	FP 32-bit	4K-LUT depth	4K-LUT depth
Input Resolution	4e-3	1e-6	8e-3	8e-3 ♠ 8e-4 ♣ 16e-3 ♦
Output Resolution	1.27e-14 1.12e-7	3.67e-40 1.12e-7	1.12e-7	1.12e-7
Input Range	[−32 32]* [−16 16]	[-88 88]* [-16 16]	[-16 16]‡	[-16 16] ‡
Output Range	[1e-14 7.9e13] [1e-7 8.8e6]	[3e-40 3.3e38] [1e-7 8.8e6]	[1e-7 8.82e6]	[1e-7 8.82e6]
Accuracy norm Inf	- - 0.356	- - 1.2e-5	0.775	0.774 ♠ 0.048 ♣ 0.784 ♦

* Maximum range supported

‡ Selected for implementation purpose

♠ Range [0 16]; ♣ Range [-1 0]; ♦ Range [-16 -1]

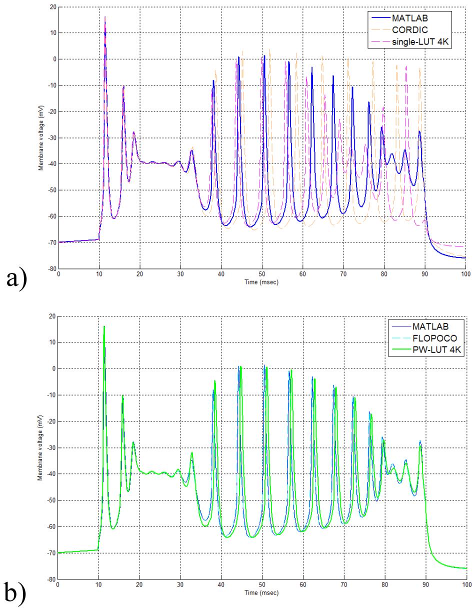


Fig 2. Hardware neuron architecture divided in three modules: soma, dendrite and synapse.

The impact of exponential accuracy in the membrane cell output can be significant or negligible depending of what exponential implementation is used. In order to measure qualitative results, a classic Traub-burst is generated applying a current of 2.3 nA during a 80 msec window. As we can see in Figure 2.a CORDIC and single-LUT can maintain accuracy initially; however there is an error accumulated over the time that results in an eventual loss of accuracy and incorrect membrane dynamics. On the other hand, in Figure 2.b the FLOPOCO and PW-LUT approaches can maintain accuracy and correct neuron dynamics.

4 Synapse integration and System architecture

In this section a novel synapse integration method is proposed to build neural networks more effectively and the complete FPGA-based neuro-system architecture is presented.

4.1. A novel synapse integration method

In neural network modelling the updating and information exchange among synapses often requires higher complexity and more calculations than the computation of the effects of each neuron in isolation [17]. The synaptic integration (or summation) is the process where multiple pre-synaptic potentials from source neurons are combined within one postsynaptic potential in the target neuron. There are two types of summation: spatial and temporal. Normally this process is calculated in two different ways: the time-driven technique and the event-driven technique. In this work, a novel hybrid timing-event-driven method is proposed; i.e. there is a fixed time step where the neuron is continuously running a steady-state process and sensing the incoming spikes, then different processes will be computed according to the type of events arriving. The main idea is to take advantage of both. Hence, rather than treating each synapse individually, the synapses of a given type

(AMPA, NMDA or GABAa) can be lumped together into a single overall synapse $g\sum$. This final synapse is updated only when individual synapses fires: $g\sum = g\sum + g_i$; i.e. when an event in the i-th synapse occurs. In this sense the amount of calculations is directly related with the amount of spiking activity in the network.

The work of [18] propose a full update rule splitting the synapse process in two parts depending on the presence (r_{on}) or absence (r_{off}) of neurotransmitters released. In a similar way, if we separate the total number of synapses N into those that activate and those that do not, we have N_{on} and N_{off} respectively. Then using the capital letters convention to refer to merged state variables, we can make next substitutions: $R_{on}=\sum_{i=1}^{Non} r_{i\,on}$, $R_{off}=\sum_{i=1}^{Noff} r_{i\,off}$ and $R_{\infty}=\sum_{i=1}^{Non} R_{\infty}= N_{on}\cdot R_{\infty}$. Using the previous substitutions we obtain the equivalent representation of a synaptic update rule with an arbitrary number of incoming synapses N:

$$R_{on}= N_{on}\cdot R_{\infty}\cdot\left[1- e^{-dt/\tau}\right]+R_{on}\cdot e^{-dt/\tau} \qquad (9)$$

$$R_{off} = R_{off}\ \ e^{-\beta\, dt} \qquad (10)$$

The next step, is to update both integration equations when a particular single synapse-i changes. This change in the new r_i needs to be reflected in the complementary state variables R_{on} and R_{off} ; i.e. when synapse-i changes to activation (off→on) then the corresponding r_i must be added to R_{on} and subtracted to R_{off} , conversely when synapse-i changes to inactivation (on→off) then R_{on} must be decremented and R_{off} augmented by the corresponding r_i . Keeping track of the individual r_i and since synaptic state variables are voltage-independent, then the new r_i is easily updated following next rule:

when on→off:

$$r_i = R_{\infty}\ \left[1-e^{-Cdur/\tau}\right]+r_i\ \ e^{-Cdur/\tau} \qquad (12)$$

when off→on:

$$r_i = r_i\ \ e^{-\beta\ ISI} \qquad (13)$$

Keeping the assumption that neurotransmitter released during the synapse process is a pulse of duration Cdur. Then when a transition on->off occurs in the synapse-i, the time step *dt* becomes the duration of the synapse pulse (Cdur). On the other hand, when transition off->on occurs, then $dt= t- t_{0\text{falling}}= ISI$; i.e. the difference between current time *t* and the last time event or the so called inter-spike interval (ISI). Finally, in neural networks the individual synaptic connections have different “weights” $g_i{}'s$ (maximal conductances and number of neurotransmitters). This can be take into consideration in equations (6.3–6.4) by creating a new variable $r_i{}' = r_i \cdot g_i$ and then redefine the merged variables: $R_{on} = \sum_{i=1}^{Non} r\,'_{i\,on}$, $R_{off} = \sum_{i=1}^{Noff} r\,'_{i\,off}$ and $N_{on} = \sum_{i=1}^{Non} g_i$.

A custom hardware neuroprocessor has been developed in order to implement the hybrid timing-event-driven synapse integration where four different events are distinguished and then the corresponding processes are executed: RE (rising-edge), FE (falling-edge), BOTH (RE and FE happen at the same time) and NC (no change). Fig. 3 depicts an example where four incoming synapses receive spikes at different times. Assuming a time step dt=0.1 msec and Cdur=1 msec for all synapses, then in the 8 msec window the processes RE, FE, BOTH and NC are executed 5, 5, 1 and 69 times respectively. In this common spiking scenario, it is clear that process NC is the most likely and the least compute intensive, showing the advantage of the proposed hybrid method.

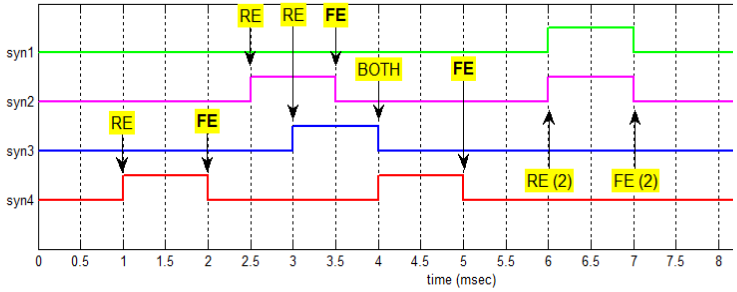


Fig 3. Example of multiple execution process for four incoming synapse pulses. The relevant events are marked in yellow, for all remaining time steps, process NC is executed.

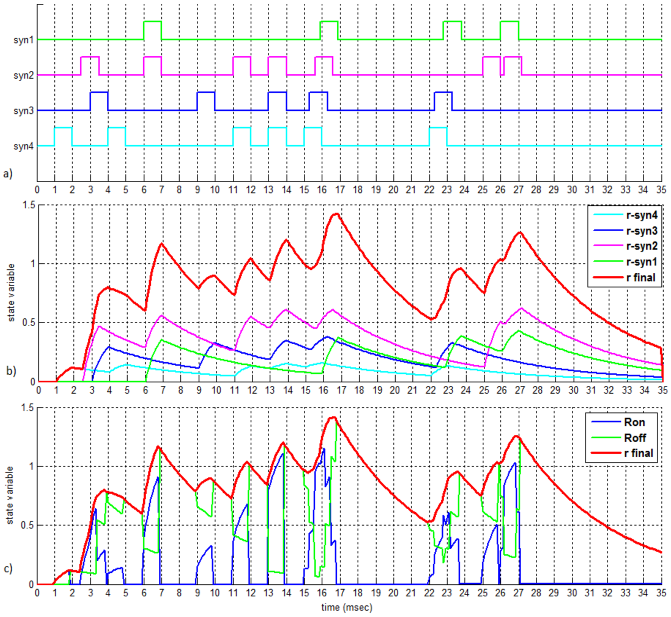


Fig 4. Qualitative comparison between full parallel time-driven method (b) and the hybrid time-event-driven method proposed (c). The final state variable *r* is depicted in red. A series of four incoming synapses with different spikes supply the post-synaptic neuron (a).

In order to validate the accuracy of the hybrid synaptic integration method, Figure 4.a shows an experiment in which an arbitrary state variable (*r,u* or *z*) and four input synapses fire at different times. A comparison against the results obtained by the full parallel time-driven approach is shown in Figure 4.b where the four outputs are sum up together to get the final summation (r-final in red). In a qualitative comparison, the hybrid time-event-driven approach gives an almost identical final summation as seen in Figure 4.c with the update of only two variables R_{on} and R_{off} at each time step instead of updating every variable for each individual synapse.

This results in a significant reduction in calculations for the proposed hybrid integration method especially the exponential operation. The reason behind this reduction is the number of times that each process in the hybrid integration method is executed. For this example process RE, FE, BOTH and NC are executed 21, 21, 1 and 315 times respectively; since process NC and FE do not need exponential calculations, just 22 exponentials operations are needed over the whole simulation against the 1400 exp operations for the full-parallel time-driven method. In total, a reduction of 80%, 76% and 98% was achieved for add/sub, multiplication and exponential operations respectively.

Overall, the hybrid time-event-driven method shows a qualitative good equivalence comparing with conventional time-driven method and it saves a considerable amount of arithmetic calculations and area cost resources.

4.2. FPGA system architecture

The simulator platform for neural networks presented in this paper is implemented in FPGA technology. Two types of components form the system: FPGA vendor IP-cores and custom IP. The IP-cores are hardware modules available to build the basic system infrastructure, in this list we have the microprocessor Microblaze, AXI-buses and memory blocks. Secondly, the custom-IP are user-designed hardware modules that permit to create custom architectures to perform specific functions; the modules for the soma, dendrite and synapse neuroprocessors proposed in this paper and general control logic are examples of these cores. A system on chip (SoC) platform has been been developed where neuroprocessors form single or two-compartment P-R neurons and control the connection between them. The general SoC FPGA architecture is shown in Fig. 5; the custom-IP neurons are connected through an AXI-bus interface; the Microblaze (MB) processor is connected to an external DDR3 RAM to store relevant data and it is running a firmware which continuously interacts with the MATLAB user-interface running in an external PC.

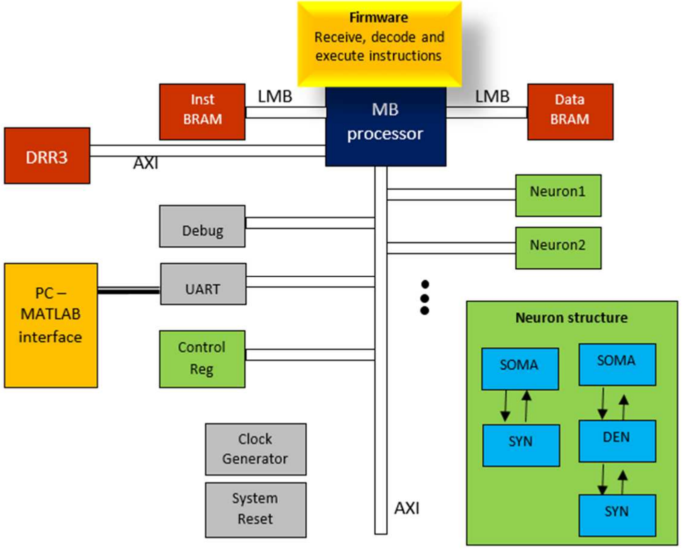


Fig 5. SoC system reconfigurable architecture

Table 2. Resources utilization for different number of neuron structures attached to the Microblaze system.

Neuron structures	**BRAMs 416 x 36kb**	**Slice Registers 301,440**	**Slice LUTs 150,720**	**TOTAL Slices 37,680**
1	6 %	7 %	18 %	26 %
2	8 %	11 %	37 %	45 %
3	8 %	18 %	49 %	61 %
4	12 %	22 %	58 %	75 %
5	14 %	30 %	72 %	88 %

The entire FPGA simulation platform is implemented in a commercial Virtex-6 ML605 board. This board has a Virtex-6 LX240T FPGA with an external 512MB DDR3 memory for general purpose storage. The Microblaze-system processor and buses work at a 50 MHz clock frequency, meanwhile the neuroprocessors do it at 100 MHz. At this operation frequency, the neuroprocessor can achieved the real-time neuron constraint of 0.1 msec. Table 2 summarizes the FPGA complexity for the whole embedded system depending of the number of neuron structures implemented; a maximum number of five two-compartment neuron structures can be used. Assuming that every neuroprocessor is working at multiplexing mode, the system is able to support neural networks of 105 neurons fully-connected (~10,000 synapses).

Our work is the first system that targets the two-compartment Pinsky-Rinzel model (P-R) and for this reason a direct comparison between this work and previous research is not possible, since there is no hardware P-R implementation reported in literature. In any case it is useful to put into context this hardware with recent FPGA-based neuron simulators. Table 3 summarizes the details in terms of aim, neuron type, performance, network structure and technology of recent research efforts in this area. As seen in Table 3 these systems are based around the H-H model which offers less biological accuracy than P-R model or the fast Izhikevich models which are not biological meaningful.

Table 3. Comparison of the presented system with recent research efforts

Research	Aim	Neuron model	Network size	Performance	Precision/NM	NN Architecture	Device
[9]	highly detailed ION cell network model	3-comp ION model (HH-based)	96 neurons	real time (323 clock cycles)	32 bit FLP	multiplexed neuron modules and kernel control	Virtex 7 – 100 MHz
[6]	synchronization on conductance-based neural networks	simplified HH	400 neurons	real time	32 bit FLP / NS	NS 20x20 topology connected	Virtex 4 – 100 MHz
[23]	extreme-scale real-time neural network	Izhikevich	64 K neurons 64 M synapses	real time (1 msec time step)	16 bit FXP / NS	centric-based communication	Stratix IV/ Cyclone IV – 200 MHz
[7]	large-scale spiking neural network	Izhikevich	64 K neurons	2.5x real time (1 msec time step)	16 bit FXP / NS	custom architecture	Virtex 6 – 100 MHz
[22]	large-scale biological networks	1-comp HH Izhikevich 1-comp Wilson	0.5 M neurons 2 M neurons 0.5 M neurons	37x over CPU 9x over CPU 6x over CPU	32 bit FLP / NS	custom with 2 layer network	Stratix II – 150 MHz Intel Xeon 3.0 GHz dual-core processor
[24]	hippocampus-inspired spiking neural network	Izhikevich	54 neurons	0.5 msec time step	32 bit FXP / NS	CA3-CA1 hippocampal topology	Virtex II – 100 MHz
[21]	leech heartbeat neural network	Izhikevich	8 neurons	real time	18 bit FXP / NS	specific 8 neuron leech architecture	Virtex 4 / 100 MHz
This work	Biophysically accurate neurons	2 compartment P-R	100 neurons	real time	32-bit FLP	Multiplexed neurons arranged in Neuroprocessors	Virtex-6 / 100 MHz

The system was also implemented in software using a Core-i7 2.3 GHz with 4 G RAM. The speedup of the hardware platform against this software solution is approximately 20 times and this is in line with the results obtained in [8] that presented a single neuron model without arithmetic optimizations. Scalability in the presented configuration is limited by device resources, maximum connectivity in the AXI interface and memory accesses. The current platform combines a controller node implemented in a single Microblaze processor with a number of Neuro processors structures that can be masters but also need to be accessed by the Microblaze as slaves. The current set of tools limit the number of slaves in a single AXI interconnect to 16 which will not fit in the Virtex-6 LX240 used in this research. Larger devices from the Virtex-6 or Virtex-7 devices could use this full cluster and combine several of this clusters to build more complex system but ultimately scalability will be limited by the memory accesses. A single cluster with 16 neuron structures will saturate the DDR3 interface available in the ML605 board so a single board configuration with a single DDR3 interface will limit scalability to approximately 16 hardware neurons.

5 In silico case studies

In this section we present two case studies to link the hardware FPGA simulator with real neuron-related experiments presented in Biology.

5.1. Analysis of synaptic mechanisms responsible for Epilepsy

We study the P-R neuron model to analyse some of the possible mechanisms that are responsible of epilepsy behaviour at a neuron response level. One of the most important features that characterize epilepsy are recurrent spontaneous seizures caused by after-discharged electrical signals presented in the neuron [19]. After-discharging (AD) occurs when neurons have the ability to discharge periodic impulses/bursts that can last several seconds, however such impulses appears after the stimulus finalisation. The after-discharge signalling is characterized by an initial large burst (F) followed by shorter sequence of burst or spikes (S). Some typical responses of experimental after-discharge CA1/CA3 pyramidal neurons in-vitro of a *guinea pig hippocampal* slice have been recorded.

If it is possible to intentionally induce epileptic signals, then the mechanism that cause such phenomena can be determined and analysed. There are effective pharmacology methods that can achieve this purpose such as manipulating the ionic composition of the bathing medium (e.g. reducing [Ca2+],[Mg2+], [Cl-]; or increasing [K+]); application of certain drugs that block specific synaptic receptors (e.g. GABAa) or by injecting a biological toxin prior to in-vivo experiments.

In this experiment, we will evoke an AD by manipulating NMDA receptors and [Mg2+] concentration. The experiments considers a cell 2 that receives a presynaptic burst from cell 1. In this scenario where only AMPA receptors are present (see Fig 6.b-left), the postsynaptic cell 2 can generate a burst as well; when AMPA and NMDA are acting together the postsynaptic burst is stronger having a few more spikes (see Fig 6.b-middle); in addition when the magnesium concentration [Mg2+] decreases, the conductance on NMDA channel increases originating a train of spikes (see Fig 6.b-right); this is one of the reasons that originates after-discharging. As we can see in Figure 6.a, the results are consistent with experiments done in [12]. The study reveals that low magnesium concentration decreases the seizure thresholds and generates spontaneous discharges in animal models of epilepsy.

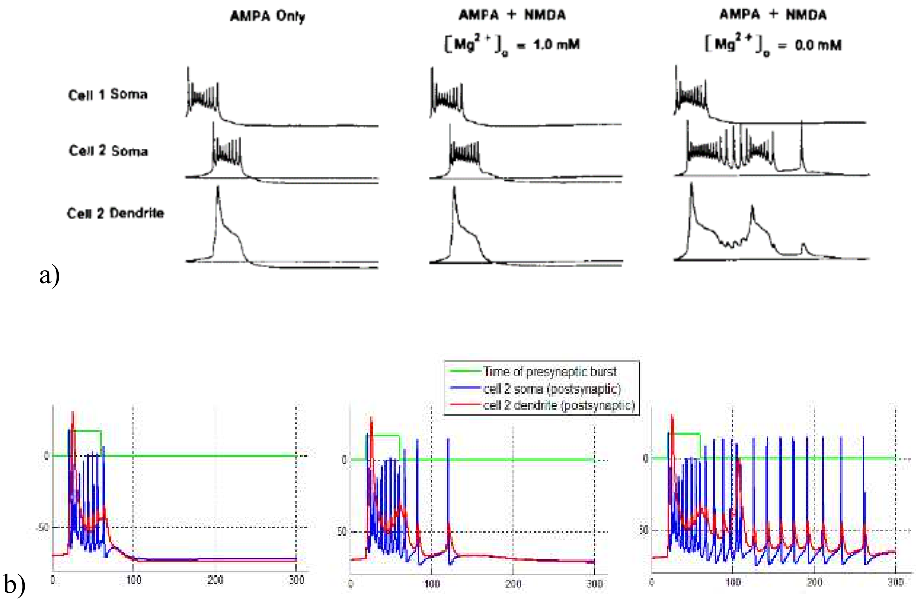
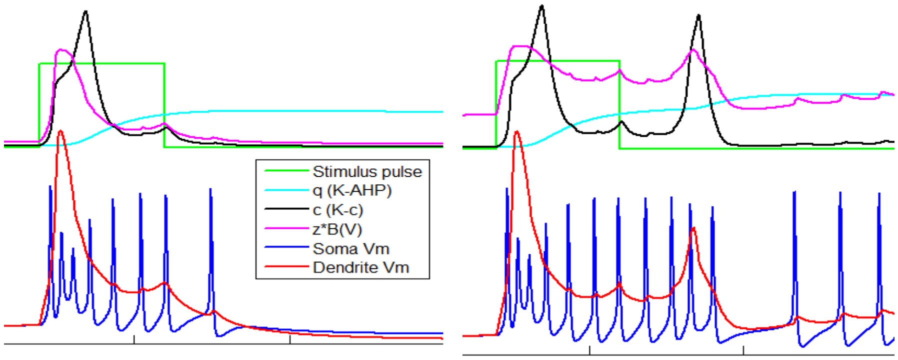


Fig 6. SoC system reconfigurable architecture

One of the main advantages of conductance-based models is that we can go into the neuron dynamics behaviour in order to analyse the results and obtain a biological meaning. For instance we analyse some relevant state variables when magnesium Mg2+ concentration changes and hence produce AD effect. In Fig 7.b the Mg2+ concentration decrease from 1 mM to 0.1 mM, this is directly reflected in the voltage-dependant NMDA channel, since the magnesium reduction increases the spike voltage dependency in the NMDA channel maintaining the z*B(V) at high levels (pink trace) and producing subsequent big dendrite spikes. When this happens, then the long slow variable *q* (cyan trace) is activated again producing a gap of hyper-polarization and stopping for a while the spiking. Due to the relatively high level of z*B(V) subsequent action potentials are generated with less frequency. In contrast Fig 7.a the bigger level of magnesium concentration makes z*B(V) decay faster and the generation of subsequent spikes does not occur.



a) b)

Fig 7. Magnesium concentration influence in the state variables changes. a) Mg2+ = 1.0 mM. b) Mg2+=0.1 mM.

There are other works that explore the non-synaptic mechanisms involve in the seizure after–discharges. For instance, the work of [19] explores the ionic channels Na+ and K+. In this work they conclude that the threshold and duration of ADs depend of variations in Na+ and K+ currents. In addition the accumulation of excess K+ can produce seizure after-discharges. Such experiments are also suitable for the the FPGA-based simulation platform proposed in this research.

5.2. Emulation of the leech heartbeat neural network

In this section we focus on the specific mechanisms that produce the leech heartbeat. Previous research has studied the Central Pattern Generator (CPG) that governs this behaviour (i.e. this is the main mechanism in living organisms to produce oscillatory patterns in neural activity), the neural network and specific neurons involved are well-defined and have been correctly identified [20]. There are two main features in a CPG: the intrinsic bursting of neurons and mutual inhibitory connections between coupled neurons. The P-R Traub model has a ping-pong effect between its soma and dendrite compartments, this makes possible to have the intrinsic bursting property and hence a suitable candidate for the experiment.

The neural network (CPG) for the leech heartbeat is formed by seven bilateral pairs of segmental interneurons that produce intercalated membrane voltage burst-oscillations (between 0.2 – 0.1 Hz) to drive the rhythm in its two hearts. However the first four pairs of interneurons (HN) are in charge of the pattern generation oscillations forming an 8-cell *timing neural network*, shown in Fig 8-left. The oscillation activity is originated by the reciprocal interaction of the third and fourth HN interneurons located in their ganglia counterpart. The alternate oscillations are mainly present at the third and fourth pair of oscillator interneurons; e.g. HN(3,Left) and HN(3,Right).

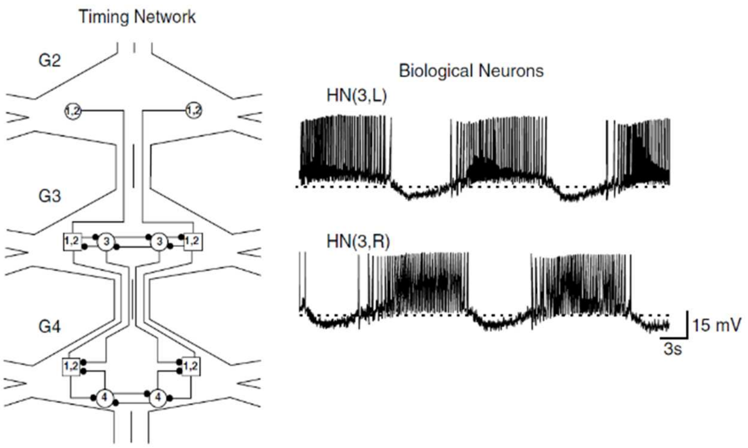


Fig 8. The timing neural network for the leech heartbeat and output signals.

The 8-cells timing neural network was built in the FPGA platform. The network is formed fully by inhibitory GABAa synapse connections. The results of the leech heartbeat FPGA-based neural network are shown in Fig 9. This figure shows that the alternate oscillatory pattern was achieved successfully, this suggest that P-R model has a good level of flexibility to emulate biological small neural networks for several applications with the appropriate conditions and the importance of conductance-based models for such experiments.

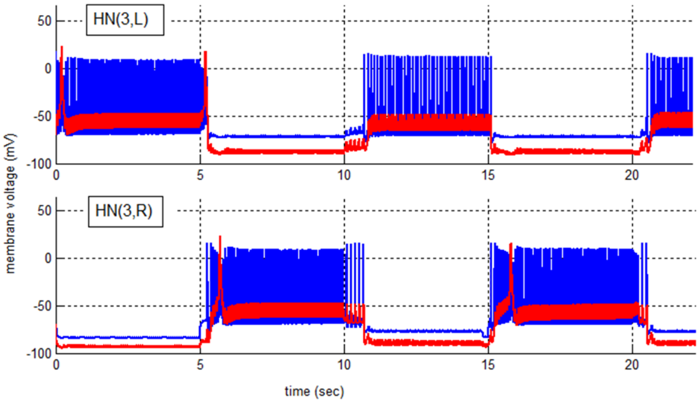


Fig 9. The FPGA timing neural networks of the leech heartbeat results. Blue and red traces correspond to soma and dendrite compartments respectively.

Other works have implement the behaviour of this CPG on FPGAs [21] but they use abstract models that decrease the biological realism in the output of the network, given a flat pulse train pattern. In addition the parameters used in conductance-based models are biophysically compatible and neuro-experimentalist can modify them to replicate organic behaviour. For example the periodic transition between inhibited and burst states is produced by a mechanism called *escape* and can be analysed using this kind of models in a well-controlled simulation platform. This mechanism happens when a neuron being inhibited by the bursting of its counterpart neuron can "escape" from this state and hence start to spike, inhibiting the bursting neuron (see Fig 10).

We can analyse the escape behaviour in a sequential series of events depicted in Fig 10. At the beginning of the graphic the left neuron HN(3,L) is inhibited by the spikes of the right neuron HN3-R. However the total synaptic conductance gtot (green trace), equivalent to just GABAa ionic channel, is decreasing gradually until the point of inhibition on HN(3,L) is no longer strong enough; in that moment the neuron HN(3,L) escapes from the inhibition stage and starts to spike. Next since HN(3,L) starts to fire, such spikes produce an inhibitory effect on the neuron HN(3,R), there is a point where HN(3.R) stops spiking and then HN(3,L) increases its burst frequency making stronger the inhibition in its right neuron counterpart. Conversely the process starts again when the synaptic GABAa conductance in HN(3,R) decreases and then it is able to escape from the inhibition stage. Such escape process is crucial in the oscillation generation process in the leech timing neural network [20].

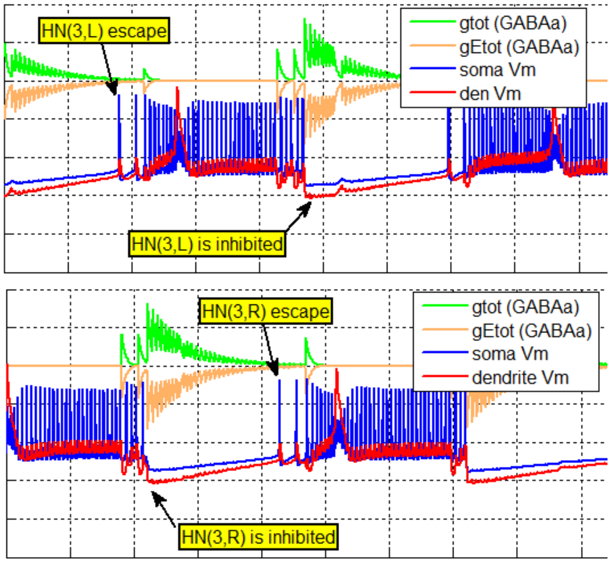


Fig 10. The “escape” mechanism in the half centre oscillator for HN3 neuron.

6 Conclusions and future work

A complete FPGA-based platform for the simulation of biophysically accurate neural networks has been proposed. The main components of this platform are the soma, dentrite and synaptic neuroprocessors in charge of implementing the P-R conductance-based model.

In order to simulate larger neural systems, the exponential which is the most area and time consuming operator was analysed and enhanced. The presented results show that the FLOPOCO exponential offers the best trade-off in terms of latency and accuracy. In addition, a piece-wise exponential LUT-based approach was proposed which uses fewer FPGA resources than the FLOPOCO approach. Both versions improve in over 95% the latency of CORDIC used in previous research. Also both present a better accuracy performance in the neuron output according to accuracy experiments.

The hardware architecture for the synapse integration provides almost identical qualitative results compared with the conventional full parallel time-driven method, which needs to evaluate all individual synapses at every time step. In contrast the hybrid method has only a single accumulative synapse approach that can handle an arbitrary number of incoming synapses. This advantage means that it is possible to have only one synapse neuroprocessor per neuron, instead of one neuroprocessor for every single synapse or connection. The hybrid time-event-driven method offers a feasible solution to tackle the integration problem reducing over 90% the number of arithmetic operations, maintaining real-time operation and resulting in no loss of biological information. The real-time properties of the FPGA system means that is possible to devise experiments in which the neuron models behave as virtual neurons and interact with organic neurons accurately.

Several new directions of research can be explored as future work. For instance, with component reuse it is possible to design a generic architecture that supports other conductance-based models. This type of models are based in the same mathematical first kinetic formula and they have a well-established dynamic mechanism. Another research line is to develop a multi-FPGA platform that supports larger neural networks so that several hundreds of neurons can be achieved. To facilitate this future research and enable reproducible research we have made all the hardware description files and supporting software available open-source at http://seis.bris.ac.uk/~eejlny/ downloads/neuroprocessor.zip

REFERENCES

[1] Dunn, N.A.; Conery, J.S.; Lockery, S.R., "A neural network model for chemotaxis in Caenorhabditis elegans," *Neural Networks, 2003. Proceedings of the International Joint Conference on* , vol.4, no., pp.2574,2578 vol.4, 20-24 July 2003

[2] Stumpner, A. and B. Ronacher (1994). "Neurophysiological Aspects of Song Pattern-Recognition and Sound Localization in Grasshoppers." American Zoologist 34(6): 696-705.

[3] Scott, A. (2000). "How Smart is a Neuron? A Review of Christof Koch's 'Biophysics of Computation'."

[4] Herz, A. V., T. Gollisch, C. K. Machens and D. Jaeger (2006). "Modeling single-neuron dynamics and computations: a balance of detail and abstraction." Science 314(5796): 80-85.

[5] Poggio, J. M. a. U. K. a. T. (2010). "CNS: a GPU-based framework for simulating cortically-organized networks." (MIT-CSAIL-TR-2010-013 / CBCL-286).

[6] Beuler, M., A. Tchaptchet, W. Bonath, S. Postnova and H. Braun (2012). Real-Time Simulations of Synchronization in a Conductance-Based Neuronal Network with a Digital FPGA Hardware-Core. Artificial Neural Networks and Machine Learning – ICANN 2012. A. P. Villa, W. Duch, P. Érdi, F. Masulli and G. Palm, Springer Berlin Heidelberg. 7552: 97-104.

[7] Cheung, K., S. Schultz and W. Luk (2012). A Large-Scale Spiking Neural Network Accelerator for FPGA Systems. Artificial Neural Networks and Machine Learning – ICANN 2012. A. P. Villa, W. Duch, P. Érdi, F. Masulli and G. Palm, Springer Berlin Heidelberg. 7552: 113-120.

[8] Zhang, Y., J. Nunez and J. McGeehan (2010). "Biophysically Accurate Floating Point Neuroprocessors." University of Bristol.

[9] Smaragdos, G., S. Isaza, M. F. v. Eijk, I. Sourdis and C. Strydis89-98 (2014). FPGA-based biophysically-meaningful modeling of olivocerebellar neurons. Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays. Monterey, California, USA, ACM: 89-98.

[10] Kulakov, A. (2012). "Multiprocessing Neural Network Simulator (PhD Thesis)." Faculty of Engineering and Applied Science. Department of Electronics and Computer Science. University of Southhampton, UK.

[11] Pinsky, P. F. and J. Rinzel (1995). "Intrinsic and network rhythmogenesis in a reduced Traub model for Ca3 neurons." Journal of Co[14]mputational Neuroscience 2(3): 275-275.

[12] Traub, R. D., R. K. Wong, R. Miles and H. Michelson (1991). "A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances." J Neurophysiol 66(2): 635-650.

[13] Hille, B. (1992). Ionic Channels of Excitable Membranes, Sinauer Associates Inc.

[14] Hughes, S. W., M. Lőrincz, D. W. Cope and V. Crunelli (2008). "NeuReal: An interactive simulation system for implementing artificial dendrites and large hybrid networks." Journal of Neuroscience Methods 169(2): 290-301

[15] Dinechin, F. d., J. Detrey, O. Cret and R. Tudoran260-260 (2008). When FPGAs are better at floating-point than microprocessors. Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays. Monterey, California, USA, ACM: 260-260.

[16] Moctezuma, J. C., J. P. McGeehan and J. L. Nunez-Yanez (2013). Numerically efficient and biophysically accurate neuroprocessing platform. Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on.

[17] Fox, P. J. and S. W. Moore (2012). "Efficient Handling of synaptic updates in FPGA-based large-scale Neural Network Simulations.". Workshop on Neural Engineering using Reconfigurable Hardware 2012

[18] Destexhe, A., Z. F. Mainen and T. J. Sejnowski (1994). "An Efficient Method for Computing Synaptic Conductances Based on a Kinetic-Model of Receptor-Binding." Neural Computation 6(1): 14-18.

[19] Kager, H., W. Wadman and G. Somjen (2007). "Seizure-like afterdischarges simulated in a model neuron." Journal of computational neuroscience 22(2): 105-128.

[20] Hill, A. A., J. Lu, M. Masino, O. Olsen and R. L. Calabrese (2001). "A model of a segmental oscillator in the leech heartbeat neuronal network." Journal of computational neuroscience 10(3): 281-302.

[21] Ambroise, M., T. Levi and S. Saïghi (2013). Leech Heartbeat Neural Network on FPGA. Biomimetic and Biohybrid Systems, Springer Berlin Heidelberg. 8064: 347-349.

[22] Bhuiyan, M. A., A. Nallamuthu, M. C. Smith and V. K. Pallipuram (2010). "Optimization and Performance Study of Large-scale Biological Networks For Reconfigurable Computing." High-Performance Reconfigurable Computing Technology and Applications (HPRCTA): 1-9.

[23] Moore, S. W., P. J. Fox, S. J. T. Marsh, A. T. Markettos and A. Mujumdar (2012). Bluehive - A field-programable custom computing machine for extreme-scale real-time neural network simulation. Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on.

[24] Mokhtar, M., D. Halliday and A. Tyrrell (2008). Hippocampus-Inspired Spiking Neural Network on FPGA. Evolvable Systems: From Biology to Hardware. G. Hornby, L. Sekanina and P. Haddow, Springer Berlin Heidelberg. 5216: 362-371.