# Implementation of a Biologically Realistic Spiking Neuron Model on FPGA Hardware

**B Glackin, LP Maguire, TM McGinnity, A Belatreche, Q Wu**

Intelligent Systems Engineering Laboratory
Magee Campus, University of Ulster,
Derry, Northern Ireland, BT48 7JL, UK.
Phone: +44-28-7137-5591, Fax: +44-28-7137-5570
Email: b.glackin@ulster.ac.uk

## Abstract

This paper presents a strategy for the implementation of biologically plausible Spiking Neural Networks (SNNs) on FPGA hardware. This re-configurable platform provides a suitable substrate for such networks, enabling increased performance by parallel processing. Combined with the flexibility of a software system, this facilitates rapid prototyping and modification of the model parameters. The Integrate and Fire model presented has been implemented on FPGA hardware following the high level design flow presented in this paper. Analysis of the implementation shows favourable performance compared with the Hodgkin Huxley neuron model.

**Keywords**: SNN, Spiking, I&F, FPGA, Hardware.

## 1. Introduction

Recent trends in computational intelligence have indicated a strong tendency towards forming a better understanding of biological systems and the details of neuronal signal processing [1]. Such research is motivated by the desire to form a more comprehensive understanding of information processing in biological networks and to investigate how this understanding could be used to improve traditional information processing techniques [2]. Spiking neurons differ from conventional artificial neural network models as information is transmitted by the mean of spikes rather than by firing rates [3-6]. It is believed that this allows spiking neurons to have richer dynamics as they can exploit the temporal domain to encode or decode data in the form of spike trains [7].

Software simulation of network topologies and connection strategies provides a platform for the investigation of how arrays of these spiking neurons can be used to solve computational tasks. Such simulations face the problem of scalability in that biological systems are inherently parallel in their architecture whereas commercial PCs are based on the sequential Von Neumann serial processing architecture. Thus, it is difficult to assess the efficiency of these models to solve complex problems [8].

When implemented on hardware, neural networks can take full advantage of their inherent parallelism and run orders of magnitude faster than software simulations and can become adequate for real-time applications. Developing custom ASIC devices for neural networks however is both time consuming and expensive. These devices are also inflexible in that a modification of the basic neuron model requires a new development cycle to be undertaken. Field Programmable Gate Arrays (FPGAs) are devices that permit the implementation of digital systems, providing an array of logic components that can be configured in a desired way by a configuration bitstream [3]. The device is reconfigurable such that a change to the system is easily achieved and an updated configuration bitstream can be downloaded to the device. Previous work has indicated that these devices provide a suitable platform for the implementation of conventional artificial neural networks [9,10]. The authors' current research seeks to devise a strategy for the implementation of large scale SNNs on FPGA hardware. A primary objective of the research is to investigate how biological systems process information and as a result the system employs biologically plausible neuron models.

Section 2 of this paper describes the neuron model that has been chosen for implementation in hardware. In section 3 an overview of the high level design flow is given, detailing the translation from high level topologies to networks configured and implemented on FPGA hardware. Results from an experiment undertaken to compare the performance of the implemented model with that of the Hodgkin Huxley

(HH) model are presented in section 4 while conclusions and future work can be found in section 5.

## 2. Neuron Model

Considerable research has been undertaken in developing an understanding of the behaviour of a biological neuron. However there are less research results available on how large arrays of these interconnected neurons combine to form powerful processing arrays. The HH spiking neuron model [11] is representative of the characteristics of a real biological neuron. The model consists of four coupled nonlinear differential equations which are associated with time consuming software simulations and would incur high "real-estate" costs when a hardware implementation is targeted [12]. Thus hardware implementations demand the application of a simpler neuron model. One such model is the Integrate-and-Fire (I&F) model which has been widely investigated by other researchers [12-14]. The model consists of a first order differential equation where the rate of change of the neuron voltage, v, is related to the membrane currents. The I&F model used is,

$$c_m \frac{dv(t)}{dt} = g_l(E_l - v(t)) + \sum_j \frac{w^j g_s^j(t)}{A}(E_s^j - v(t)) \quad (1)$$

For a given synapse j, an action potential (AP) event at the presynaptic neuron at time $t_{ap}$ triggers a synaptic release event at time $t_{ap} + t_{delay}^j$, a discontinuous increase in the synaptic conductance

$$g_s^j(t_{ap} + t_{delay}^j + dt) = g_s^j(t_{ap} + t_{delay}^j) + q_s^j \quad (2)$$

otherwise $g_s^j(t)$ is governed by

$$\frac{dg_s^j(t)}{dt} = \frac{-1}{\tau_s^j} g_s^j(t) \quad (3)$$

As the I&F model equation is a differential equation in time the forward Euler integration scheme was used for numerical integration of the equation with a time step of dt=0.125ms. Using this method differential equation (1) becomes

$$v(t+dt) = v(t) + \frac{1}{c}((g_l(E_l - v(t)) + \sum_j \frac{w^j g_s^j(t)}{A}(E_s^j - v(t)))dt \quad (4)$$

and differential equation (3) becomes,

$$g_s^j(t+dt) = g_s^j(t) + (\frac{-1}{\tau_i} g_s^j(t))dt \quad (5)$$

The parameter set for neurons is as follows

$\{c_m = 8\text{nF/mm}^2,\ E_l = -70\text{mV},\ v_{th} = -69\text{mV},\ v_{reset} = -70\text{mV},$

$\tau_{ref} = 6.5\text{ms},\ t_{delay}^j = 0.5\text{ms}\}$

The synapses parameters are either set to excitatory or inhibitory values depending on the type of synapse required. The following list shows both the excitatory and inhibitory values where the notation used is parameter=exc_value/inh_value,

$\{g_l = 1\text{uS/1uS},\ A = 0.03125\text{mm}^2/0.015625\text{mm}^2,$

$E_s^j = 0\text{mV/-75mV},\ \tau_s^j = 1\text{ms/4ms}\}$

The hardware implementation employed a fixed point coding scheme. 18 bits were used to represent the membrane voltage (4), 10 bits of which correspond to the fractional component. The synaptic conductance (5), was implemented using 12 bit precision for the fractional component. Multiplicand and divisor parameters were chosen as powers of 2 so that they could be implemented using binary shift thus utilising a smaller amount of logic than what would be required by a full multiplier or divider.

## 3. High-Level System Design

To enable an efficient translation from abstract network topologies, a high level design environment is required. Such an environment should provide both graphical and high level language facilities for large scale network exploration and simulation but should also enable a seamless integrated route between the high level representation and the silicon implementations incorporating simulation, synthesis, and place and route tools. An overview of this design flow is shown in Fig. 1.

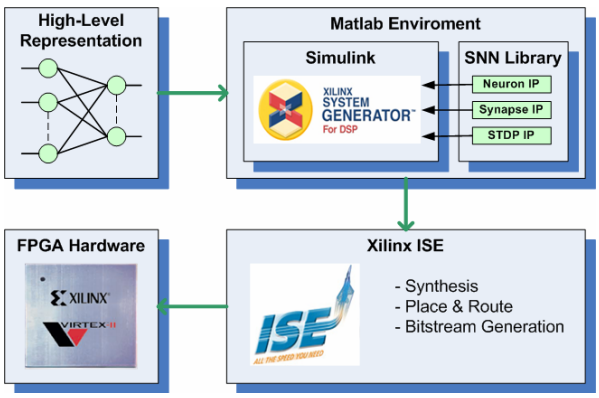


Fig. 1: High level design flow

Matlab is a well known standard tool for high level language and graphical exploration of various signal processing and network topologies. It was decided to

explore the possibility of reaching silicon from this top level environment. Matlab includes the possibility of utilising Xilinx System Generator to automatically generate synthesisable VHDL code for digital signal processing applications. The Matlab-Simulink-Xilinx System Generator concept was then extended by adding a custom designed Spiking Neural Network (SNN) blockset.

Using the Matlab-Simulink-Xilinx System Generator route by means of the SenseMaker SNN blockset, users can apply the Simulink graphic design tool to design SNN models. The Xilinx System Generator can subsequently automatically generate VHDL code for input to the Xilinx ISE tool. It is then possible to use ISE to generate a bit-stream and download it to configure the Xilinx FPGA device.

The SNN blockset includes excitatory and inhibitory synapse blocks, I&F neuron models, and Spike Time Dependant Plasticity (STDP) components. The blockset has been applied to develop electronic/computational representations of the models described in the previous section and thus can be used to implement SNN topologies on FPGA devices.

Although the model described is less complex than the HH model the amount of logic required to implement the network component blocks on the FPGA means that the number that can be implemented in parallel is limited. The logic requirements to incorporate the SNN components on a Xilinx Virtex II series device (XC2V8000) are shown in Table 1.

| SNN Component | Slices | Embedded Multipliers |
|---|---|---|
| Synapse | 33 | 0 |
| Neuron | 63 | 1 |

**Table 1:** SNN Logic Requirements (XC2V8000 device)

Using a fully parallel implementation the number of neurons that can be implemented on an FPGA is generally limited to the number of embedded multipliers provided. Further multipliers can be generated from the logic, however this is area intensive and rapidly consumes the remainder of the resources, leaving little logic to implement the synapses. The largest device in the Xilinx Virtex II series of FPGAs is the XC2V8000 device which consists of 46,592 slices and 168 embedded multipliers. Therefore using a synapse to neuron connection ratio of 1:1 a total of 168 neurons and 168 synapses could be implemented on a XC2V8000 device. Increasing the connection of synapses to neurons to a more realistic ratio of 100:1 enables a network consisting of 13 neurons and 1300 synapses to be implemented.

Utilising this fully parallel implementation with a clock speed of 100 MHz and a Euler integration time step of 0.125ms per clock period, a 1 second real time period of operation could be completed in 0.08ms. This provides a speed up factor of 12,500 compared to real time processing. This compares very favourably with a software implementation where the data is processed serially and can take hours to run a several second real time simulation. The speed up factor indicates that there are possibilities for examining speed/area trade offs and determining alternative solutions that will lead to larger network capabilities. Such implementations are currently under development as part of this research work.

# 4. Two-Neuron Experiment

To benchmark the performance of the I&F model implemented on the FPGA with the HH model, the two neuron experiment used in the SenseMaker project was implemented [15]. Simulations of the experiment for the HH model using the Neuron software tool were provided by Alain Destexhe [16]. The experiment involves a single pyramidal (excitatory) neuron (PY) which is presynaptic to an excitatory synapse of peak conductance $w_{EI}$ located on an inhibitory neuron (IN) which is in turn presynaptic to an inhibitory synapse of peak conductance $w_{IE}$ located on the PY neuron. Additionally the PY neuron is injected with a constant current I. A schematic illustrating the experiment can be found in Fig. 2.

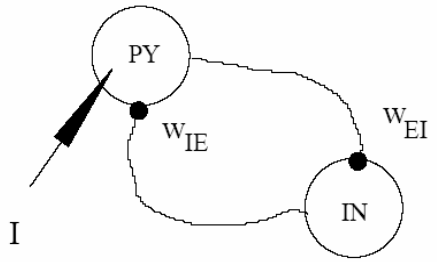


Fig. 2: Two neuron experiment schematic

This experiment was implemented on a Xilinx Virtex II XC2V8000. The graph in Fig. 3 shows a comparison between a software simulation of the HH model (Neuron), a software simulation of the I&F model (Matlab) and the FPGA hardware implementation of the I&F model. The graph plots the firing rate of the PY neuron for varying values of synaptic strength. Overall the performance of the two models is quite similar however there are some discrepancies between the software and hardware implementations of the I&F model at the lower values of synaptic strength. In general however, variations are

expected when using a fixed point coding scheme due to quantisation errors.

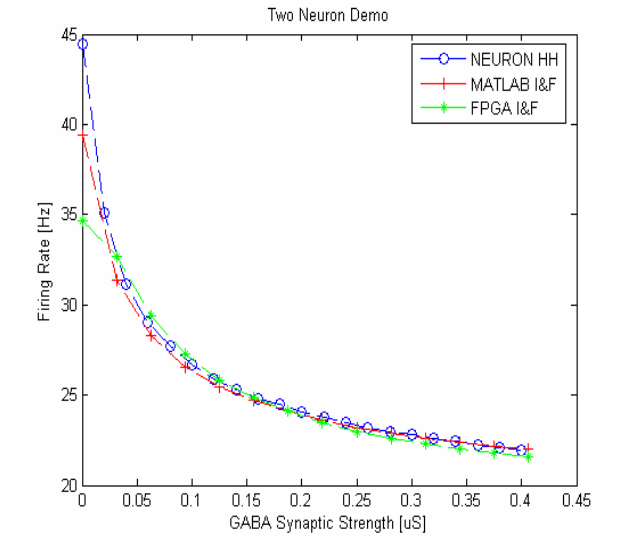


Fig. 3: Two neuron experiment (f-v-$W_{IE}$)

## 5. Conclusions

This paper has presented a strategy for the implementation of biologically plausible neural networks on FPGA hardware. The I&F model was found to behave favourably compared to the HH neuron model when performance was compared using the two-neuron experiment. FPGAs provide a viable platform for the implementation of SNNs and the high level approach presented provides a design flow from a high level environment to silicon.

Analysis of the approach however indicates that the available FPGA resource limits the size of network that can be implemented. Speed performance of the implementation would suggest that trade offs in terms of speed/area could be made to provide a system that can simulate large arrays in real time. This work is currently under investigation with the aim of realising large-scale implementations on FPGAs. In addition, further research aims to improve the speed performance by employing an event based strategy as utilised in some biologically plausible neural simulator tools.

## 6. Acknowledgements

The authors acknowledge the financial and technical contribution of the SenseMaker project (IST-2001-34712) which is funded by the EC under the FET life like perception initiative.

## 7. References

[1] U. Roth et al, "Hardware Requirements for Spike-Processing Neural Networks", From Natural to Artificial Neural Computation (IWANN), Springer Verlag, 720 -727, 1995.

[2] Wolfgang Maass, Christopher M. Bishop, Pulsed Neural Networks, MIT Press.

[3] A. Upegui et al, A methodology for evolving spiking neural-network topologies on line using partial dynamic reconfiguration, ICCI, Medellin, Colombia. November 2003.

[4] Haykin. Neural Networks, A Comprehensive Foundation, Prentice-Hall, New Jersey, 1999.

[5] A. Perez-Uribe. Structure-adaptable digital neural networks. PhD thesis. 1999. EPFL.

[6] E. Ros, R. Agis, R. R. Carrillo E. M. Ortigosa. Post-synaptic Time-Dependent Conductances in Spiking Neurons: FPGA Implementation of a Flexible Cell Model. Proceedings of IWANN'03: LNCS 2687, pp 145-152, Springer, Berlin, 2003.

[7] D. Roggen et al, "Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot," NASA Conference on Evolvable Hardware July 2003.

[8] A. Delorme et al, SpikeNET: A simulator for modeling large networks of integrate and fire neurons, Neurocomputing, 26- 27, 989-996 1999.

[9] J.J. Blake, T.M. McGinnity, L.P. Maguire, The Implementation Of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks Using FPGAs, Information Sciences, Vol. 112, No. 1-4, pp. 151-68, 1998.

[10] B. Glackin , L. P. Maguire, T. M. McGinnity, Intrinsic and extrinsic implementation of a bio-inspired hardware system, Information Sciences, Volume 161, Issues 1-2 , April 2004, Pages 1-19

[11] Hodgkin, A. L. and Huxley, A. F. (1952) "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve" Journal of Physiology 117: 500-544

[12] Gerstner W, Kistler W, Spiking Neuron Models: Single Neurons, Populations, Plasticity, Cambridge University Press, 2002

[13] Dayan P, Abbott LF, Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, MIT Press, Cambridge, 2001

[14] Koch C, Biophysics of Computation: Information Processing in Single Neurons, Oxford University Press, 1999

[15] http://sensemaker.infm.ulst.ac.uk/

[16] http://www.cnl.salk.edu/~alain/