ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

An FPGA-based Model suitable for Evolution and Development of Spiking Neural Networks

Hooman Shayani[1], Peter J. Bentley[1], and Andrew M. Tyrrell[2]

[1] UCL, Dept of Computer Science, WC1E 6BT - UK

[2] University of York, Dept of Electronics, York - UK

Abstract. We propose a digital neuron model suitable for evolving and growing heterogeneous spiking neural networks on FPGAs using a piecewise linear approximation of the Quadratic Integrate and Fire (QIF) model. A network of 161 neurons and 1610 synapses with 4210 times real-time neuron simulation speed was simulated and synthesized for a Virtex-5 chip.

1 Introduction

The evolution of directly mapped recurrent spiking neural networks on FPGAs has been tackled by a few researchers (*e.g.* [1, 2]) using very simplified versions of the Leaky Integrate and Fire model (LIF) [3]. Recently, Schrauwen *et. al* [4] proposed a high-speed spiking neuron model for FPGA implementation based on the LIF model with serial arithmetic and parallel processing of the synapses utilising pipelining in a binary tree for dendrites. Researchers have sought to create POE (Phylogeny, Ontogeny, Epigenesis) neural networks capable of evolving, developing (growth) and learning *in situ* to adapt themselves to the given problem and environment (*e.g.* POEtic chip [5]). However, as yet none of these digital neuron models are quite suitable for an online developmental model [1] capable of regeneration and growth (Morphogenesis) on FPGA. They are typically either limited in terms of number of inputs per neuron or impose constraints on the patterns of connectivity and/or placement on the actual chip mostly due to implementation issues. They also do not allow a heterogeneous networks of flexible parametric neurons and learning rules as important bio-plausible features.

This paper proposes a digital spiking neuron model suitable for developmental evolution of heterogeneous recurrent neural networks on FPGAs aiming at: flexibility, development-friendliness, high simulation and learning speeds, parallelism, and bio-plausibility, while having hardware implementation in mind.

2 Digital Neuron Model

To improve the performance of evolution, the developmental digital neuron model should be as flexible as possible, for any constraint may impair evolvability. Evolution must be able to modify everything from network topology and dendrite structures to learning rules, membrane decay constants and other cell parameters and processes. Evolution should be also free to create a suitable neuro-coding technique for each application. Therefore, the network activity is not guaranteed to be restricted as assumed in event-based simulation of spiking

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

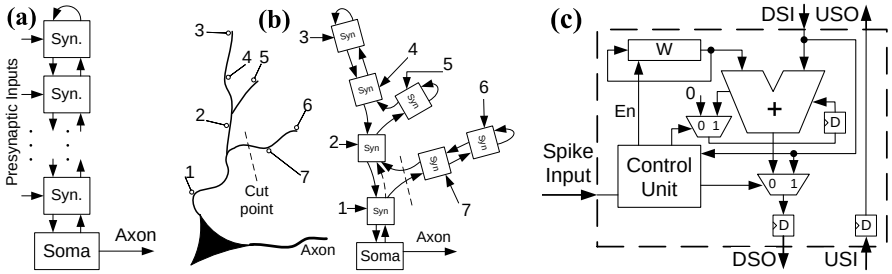

Fig. 1: a) General architecture of the digital neuron (Syn=synapse) b) Example of the dendrite structure and its adaptability (c) Synapse unit architecture.

neural networks [4]. Thus, a time-step simulation technique is used here. This model also needs to be relatively fast as running a POE system [5] involves iterative nested cycles of evolution, development and learning. Such a fast parallel spiking neural network on FPGA can also be used for real-time applications.

In the proposed model, each digital neuron consists of a set of synapse units and a soma unit connected in a daisy chain architecture shown in figure 1(a). The spike input of each synapse is connected to the axon of the pre-synaptic neuron. This architecture creates a 2-way communication channel and allows the development of different dendrite structures as demonstrated in the example of figure 1(b). The signal pairs that connect the units forms a loop that conveys data packets (a start bit and 16 data bits). The soma unit sends an upstream packet containing the current membrane potential on its upstream output (USO). Synapse units pass upstream packets unchanged but process downstream packets. If a synapse unit receives a pre-synaptic action potential it adds (subtracts) its synaptic weight to the first arriving downstream packet. Therefore, the soma unit receives the sum of membrane potential and post-synaptic currents in its downstream input (DSI). After processing this packet, the soma unit sends another packet with the updated membrane potential. Serial arithmetic is used in all the units to create pipelined parallel processing inside each neuron, meaning that neighbouring units process different bits of the same packet at the same time. Using this architecture has a number of collective benefits. First, a 2-way communication channel makes it possible to have a local synaptic plasticity mechanism in each synapse leading to a higher level of parallelism. Most of the bio-plausible unsupervised learning mechanisms like STDP and its variants involve a local learning process in each synapse. Secondly, it minimizes the number of local and global connections, which leads to a significant relaxation of constraints imposed upon the network architecture as limited routing resources is the major constraint in optimal utilization of FPGA functional resources. Each unit needs only a global clock signal to work. Another global signal can be added for a global supervised learning mechanism. Although other architectures may bring about less pipeline latency, they need more local and

global connections. For instance, a binary tree structure similar to [4] needs about double the number of local connections including the upstream links (excluding the global control signals). Third, it allows to develop any dendrite structure similar to biological dendrites. The user is free to trim (add) dendrite sub-trees at any point simply by cutting (connecting) a (pair of) connection(s) and bypassing (inserting) the root unit of the sub-tree as shown by the dashed lines in figure 1(b). This can be implemented in FPGA using multiplexers or other routing resources (The detail is beyond this paper). This flexibility is vital for a developmental model that needs on-line growth and modification. Fourth, it maintains the regularity of the model by limiting the diversity of the module types (synapse and soma units) and connection types (dendrites, axons) to a biologically plausible bare minimum. This simplifies the place and route or dynamic reconfiguration process if a regular infrastructure of cells and connections (similar to [6]) is used. Finally, it is possible to add other variables to the data packet (*e.g.* the membrane recovery variable in the Izhikevich model [3]).

2.1 The Synapse Unit

The synapse unit, shown in figure 1(c), comprises a 1-bit adder, a shift register containing the synaptic weight, two pipeline flip-flops, and a control unit. The upstream input (USI) is simply directed to the upstream output (USO) through a pipeline flip-flop. The control unit disables the adder and weight register when no spike has arrived by redirecting the downstream input (DSI) to the downstream output (DSO) through another pipeline flip-flop. When the control unit detects a spike, it waits for the next packet and resets the carry flip-flop of the adder when it receives the start bit. Then it enables the shift register and the adder until the whole packet is processed. A learning block can be simply inserted into the feedback loop of the weight register in order to realize an unsupervised local learning mechanism like STDP. This learning block can access the current membrane potential and the input. It is also possible to modify the synapse to create a digital DC current input unit by loading the DC current into the weight register.

2.2 The Soma Unit (PLAQIF model)

Most of the hardware models are based on the LIF [3] or simplified LIF neuron models [1, 2]. However, a Quadratic Integrate and Fire neuron model (QIF) is biologically more plausible compared to the popular LIF model [3]. Here, a Piecewise-Linear Approximation of the Quadratic Integrate and Fire (PLAQIF) is proposed as a new soma model. Using a PLAQIF model has a number of benefits in our context. While it is relatively inexpensive (in terms of hardware resources) to convert a serial arithmetic implementation of a LIF neuron model into a PLAQIF model (as shown later), PLAQIF model can generate a bio-plausible action potential. This is particularly important as we use the membrane voltage in the learning process. Moreover, the Behaviour of the model can be specified with a number of parameters (*i.e.* time constants and reset poten-

tial). These parameters can be placed in registers and look-up-tables (LUT) to be modified at run-time (*e.g.* by partial dynamic reconfiguration) or can be hard-wired for hardware minimization. Finally, it is easy to extend this model to a piecewise-linear approximation of Izhikevich model (with a wide range of bio-plausible behaviours *e.g.* bursting, chattering, and resonating [3]) by adding another variable, if hardware budget permits. The dynamics of the QIF model can be described by a differential equation and reset condition of the form [3]:

$$\dot{u} = a(u - u_r)(u - u_t) + I \ , \quad \text{if } u \geq u_{peak} \text{ then } u \leftarrow u_{reset} \tag{1}$$

where u is membrane voltage, a is specifying the time-constant, I is the post-synaptic input current, and the u_r and u_t are nominal resting and threshold voltages respectively, when $I = 0$. Note that in contrast with LIF models, the actual resting and threshold voltages are dynamic and they change with the input current I [3]. Applying first-order Euler method results in an equation of general form:

$$u_{k+1} = u_k + a(u_k - u_r)(u_k - u_t) + I_k \ , \quad \text{if } u_{k+1} \geq u_{peak}, \text{ then } u_{k+1} \leftarrow u_{reset} \tag{2}$$

where k is the step number. The PLAQIF model is based on the serial arithmetic implementation of a LIF model with equation $u_{k+1} = u_k + I - au_k$ (for $a < 1$) with a little modification. The last term can be approximated using two taps:

$$u_{k+1} = u_k + I_k + \underbrace{\left\lfloor \frac{u_k}{P_1} \right\rfloor}_{\text{Tap 1}} + \underbrace{\left\lfloor \frac{u_k}{P_2} \right\rfloor}_{\text{Tap 2}} \tag{3}$$

where $P_i = (-1)^{s_i} \cdot 2^{p_i}$ with p_i and s_i being the parameters of ith tap. Each tap is computed by adding (or subtracting depending on s_i) the shifted version (arithmetic shift right by p_i bits) of the binary representation of u_k. By replacing the sign bit (`S`) and the most significant bit (`MSB`) of u_k with the complement of `MSB` we can produce the piecewise linear function $V(u_k) = |u_k| - 2^{14}$ (assuming a 16-bit representation). This function is shown in figure 2(a) as the V-shape function. By tapping (modulating) $V(u_k)$ with different parameters ($p_{i,0} \ldots p_{i,3}$ and $s_{i,0} \ldots s_{i,3}$) for different combinations of `S` and `MSB` (positive or negative, small or large values of u_k) we get:

$$u_{k+1} = u_k + I_k + \left\lfloor \frac{V(u_k)}{P_1(u_k)} \right\rfloor + \left\lfloor \frac{V(u_k)}{P_2(u_k)} \right\rfloor \tag{4}$$

$$\text{where } P_i(x) = (-1)^{s_{i,j}} \cdot 2^{p_{i,j}} \ , \quad j = \left\lceil \frac{x}{2^{14}} \right\rceil + 1 \tag{5}$$

It is possible to approximate eq. 2 with eq. 4 by tuning the parameters $p_{i,j}$ and $s_{i,j}$ as shown in figure 2(a). The soma unit, showed in figure 2(b), comprises a 1-bit adder, a 32-bit buffer shift register (holding the partial sums from the last cycle), a 16-bit shift register (holding reset voltage u_{reset}), a lookup-table (LUT, a 8x5 bits RAM, which holds the parameters $p_{i,j}$ and $s_{i,j}$), a control unit (CU,

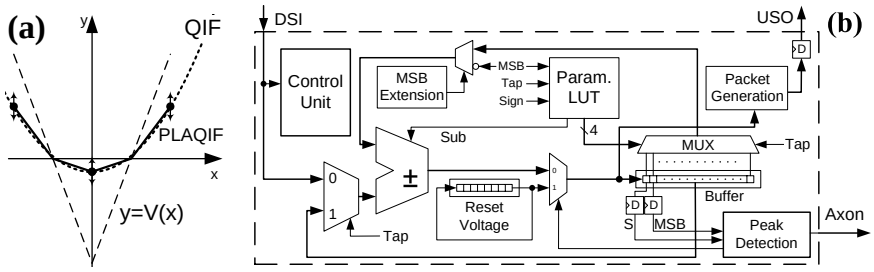

Fig. 2: (a) The PLAQIF model approximates the QIF model (the dotted curve) with a piecewise linear function by modulating the V-shape function $V(x)$. The control points (arrows) can be moved by tuning the parameters. (b) Soma unit

which detects the arriving packet and generates all the control signals *e.g.* `Tap`, `ShiftEn`, etc.), and a few multiplexers. The soma unit initiates a data packet thorough USO and waits for a packet on DSI input. At this point, the buffer holds the value u_k in its left half and `S` and `MSB` flip-flops hold the sign and most significant bit of u_k. The LUT selects the correct shifted version (according to `S` and `MSB`) of u_k through the multiplexer and has its first bit ready on the input of the adder. The first tap starts with receiving a packet. An arriving packet, which contains the value $u_k + I_k$ goes to the other input of the adder. The LUT also selects the add or subtract operation in each tap(s_i). As the operation goes on, the MSB extension block switches the multiplexer to $\overline{\texttt{MSB}}$ at the right time to generate the value $\left\lfloor \frac{V(u_k)}{P_1(u_k)} \right\rfloor$ on the input of the adder. Therefore, the new value of $u_k + I_k + \left\lfloor \frac{V(u_k)}{P_1(u_k)} \right\rfloor$ shifts into the buffer through a multiplexer. The second tap starts immediately and the value in the left half of the buffer goes to the adder input. The other input of the adder is again $\left\lfloor \frac{V(u_k)}{P_2(u_k)} \right\rfloor$ now generated by selecting the correct shifted version of the u_k from the right half of the buffer. The adder generates the updated value of u (u_{k+1} in eq. 4) at its output, which is shifted into the buffer and is also used to generate a new packet in the upstream output of the soma unit. This value is also used to update the `S` and `MSB` flip-flops according to the new value of u_{k+1}. This process continues until the peak detection block detects a transition of `S` without any change in `MSB`, which indicates an overflow, and immediately corrects the sign bit of the departing packet, generates a pulse in the axon, and initiates the absolute refractory period. The absolute refractory period, which lasts for a complete membrane update cycle, is like any other cycle except that in the second tap the output of the adder is ignored and contents of the reset voltage shift register is used instead. The membrane update period (*i.e.* latency of the whole pipeline), thus neuron time constants, depend on the number of synapses n ($T = 2n + 18$ clock cycles). This can be compensated by evolving parameters.

3 Implementation

The behaviour and flexibility of the neuron model was verified by VHDL simulation of a single neuron. Random spikes were fed into 16 synapses with different weights using different bio-plausible parameter settings and its membrane potential was monitored and compared to the expected dynamics of equation (4). With efficient use of the 32-bit shift registers in Virtex-5 FPGA, a random small-world network of 161 16-bit neurons with 20 inputs, 20 outputs and, 10 fixed-weight synapses per neuron, was simulated and synthesized for a XC5VLX50T chip using VHDL and Xilinx ISE resulting 85% utilization and a maximum clock frequency of 160MHz (4210 times real-time neuron simulation speed with a 1 ms resolution). We believe that we can improve some of these figures by low-level design optimization and a cellular floor-planning similar to [6]. Compared to [4], this model is 43% faster, but needs more hardware resources. However, it is difficult to compare these two designs due to differences in design objectives (flexibility and adaptability instead of hardware minimization and speed) and technologies. Clearly, the replication of the synapse control units increased hardware resources but also contributed to the adaptability of the dendrite structure and the real possibility of introducing meaningful development into the process.

4 Conclusions

A digital spiking neuron model with a new architecture and a novel soma model (PLAQIF) was proposed and its suitability for evolution and development of heterogeneous neural networks in FPGAs was shown in terms of parametric flexibility of the soma units, adaptability of the dendrite structures, and model simulation speed. It was also shown how a local learning unit can be added to each synapse to improve parallelism. Although the replication of the synapse control units increases the hardware usage, it brings about the adaptability of the dendrite structures and may also adds to the fault-tolerance of the network.

References

[1] Daniel Roggen, Diego Federici, and Dario Floreano. Evolutionary morphogenesis for multi-cellular systems. *Genetic Programming and Evolvable Machines*, 8(1):61–96, 2007.

[2] Andres Upegui, Carlos Andres Pena-Reyes, and Eduardo Sanchez. An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and Microsystems*, 29(5):211–223, June 2005.

[3] Eugene M. Izhikevich. *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting (Computational Neuroscience)*. The MIT Press, November 2007.

[4] B. Schrauwen and J. van Campenhout. Parallel hardware implementation of a broad class of spiking neurons using serial arithmetic. *Proceedings of ESANN*, 2006.

[5] J.M. Moreno, Y. Thoma, E. Sanchez, J. Eriksson, J. Iglesias, and A. Villa. The POEtic Electronic Tissue and Its Role in the Emulation of Large-Scale Biologically Inspired Spiking Neural Networks Models. *Complexus*, 3(1-3):32–47, 2006.

[6] A. Upegui and E. Sanchez. Evolving Hardware with Self-reconfigurable connectivity in Xilinx FPGAs. In *Adaptive Hardware and Systems, 2006. AHS 2006. First NASA/ESA Conference on*, pages 153–162, 2006.