Reconfigurable Acceleration of Neural Models with Gap Junctions

Mark Wildie [1], Wayne Luk [1], Simon R. Schultz [2], Philip H. W. Leong [3], Andreas K. Fidjeland [1]

[1] *Department of Computing*
[2] *Department of Bioengineering*
Imperial College London, England
{mark.wild05,w.luk,s.schultz,andreas.fidjeland}@imperial.ac.uk

[3] *School of Electrical and Information Engineering*
University of Sydney, Australia
phwl@ee.usyd.edu.au

***Abstract*—We describe the design and implementation of an FPGA-based architecture for real-time simulation of spiking neural networks that include gap junctions, a type of synapse not often used in neural models due to their high computational cost. Recent research suggests that electrical synapses or gap junctions play a role in synchronizing the activity of larger groups of neurons in the brain, and are potentially important in high level functions such as cognition and memory. We suggest the simulation cost of gap junctions can be reduced by clustering them within the model, which is consistent with evidence of the structure of gap junction networks and allows each cluster to be updated in parallel. Our implementation on a Xilinx Virtex-5 FPGA demonstrates a 24.3 times speedup over a software implementation running on a cluster of four 3.6GHz Intel Xeon processors. This is part of a larger effort to construct tools capable of real-time simulation and exploration of realistic brain networks of comparable size to biological networks.**

I. Introduction

Fast simulation of large, biologically accurate neural networks is a long held goal of researchers studying the function of the human brain. Neuroscience has provided a wealth of information on the operation of the basic components of the brain, neurons and synapses [11]. We now understand the fundamental processes governing the state of neurons and interaction between them, and understand enough about the connectivity of the brain to build networks that display similar dynamics [9]. Despite this, real-time simulation of even modestly sized networks remains out of reach.

A single processor is insufficient to simulate such a massively connected and dependent network in real-time, and real-time simulation does not scale to clusters. Custom architectures built to support as much parallelism per processing unit and as much bandwidth between units as possible have significant potential. FPGAs and GPUs are a promising route to realizing real-time simulation using off-the-shelf components.

In any biologically plausible neural model, the number of synapses will far outweigh the number of neurons. Modeling this connectivity and the communication between neurons is one of the key challenges in neural simulation. Neurons communicate relatively infrequently over chemical synapses, preventing propagation of spikes from overwhelming simulation times. Gap junctions are often not included in large neural networks, as they involve constant communication between connected neurons. There is evidence to suggest that they play an important role in memory and learning, and in synchronizing the activity of large groups of neurons [2]. If we are interested in reproducing and exploring the dynamics of biological networks, it is important to at least have the option of including gap junctions in our models.

There are two areas of research we think will benefit from this work. The first is computational neuroscience. Reproducing experimental results through modeling gives some insight into the function of real networks, and increases our understanding of how populations of neurons act together. Real-time simulation of large networks will let us take this a step further and allow real input, feedback and interaction with sensory and motor systems. The second is research into reproducing the function of the brain, such as biologically inspired control systems for robotics.

Our main contributions are:

- A strategy for minimizing the impact of gap junctions on real-time simulation that is consistent with the biological structure of gap junction networks (Section III).
- An FPGA architecture for flexible, real-time simulation of realistic networks of spiking neurons including gap junctions (Section V).
- Extensions to existing CPU and GPU based modeling systems to include gap junctions, and comparison to our FPGA implementation (Section VI).

II. Background and Related Work

Neurons communicate by means of all-or-none fluctuations in their membrane potentials, called action potentials (or ”spikes”). Patterns of spikes emanating from individual neurons and populations of neurons, and the rate and temporal characteristics of those patterns, are the basic unit of information transfer in the brain.

When a neuron emits a spike it is conveyed to other neurons via chemical synapses, where it contributes to the electrical potential of those neurons and affects the likelihood of them emitting a spike of their own. Neurons can have an inhibitory or excitory effect depending on whether spikes from that neuron increase or decrease the potential of connected neurons.

A second class of synapses called electrical synapses or gap junctions have been demonstrated to occur between inhibitory neurons. They differ from chemical synapses in that the interiors of the opposing neurons are directly connected. There is strong evidence that gap junctions between inhibitory neurons in the neocortex form functionally distinct networks [6] connecting populations of inhibitory interneurons with the same properties.

There is a significant amount of existing research into the application of FPGAs [10], and more recently GPUs [4] and other custom architectures [5], to simulating traditional and spiking neural networks.

III. Neural Model with Gap Junctions

A number of models ranging in complexity and biological plausibility have been proposed to represent neurons in spiking neural networks. We use the recent Izhikevich model [7], as it is efficient and supports a large number of different neuron behaviours. Each neuron is described by the following three equations:

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

$$u' = a(bv - u) \tag{2}$$

$$\text{if } v \geq 30 \text{ then } \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3}$$

where v is the neuron membrane potential, u is the membrane recovery variable, and I is the input current. The neuron spikes when v reaches 30mV, and is reset as described by equation (3).

We replace I in the Izhikevich update equation with a term to represent synaptic input to the neuron

$$v' = 0.04v^2 + 5v + 140 - u + \sum_{j \epsilon \phi} S_{i,j} F \tag{4}$$

To update neuron i at time t, ϕ is the set of neurons j connected to i that fired at time $t - \delta$ where δ is the conductance delay between i and j, $S_{i,j}$ is the synaptic weight between i and j, and F is a constant scaling factor. We include electrical synapses in the neuron update equation with an additional term that takes into account the difference in potential between the two gap junction connected neurons

$$\begin{aligned} v' = 0.04v^2 + 5v + 140 - u + \sum_{j \epsilon \phi} S_{i,j} F \\ + \sum_{k \epsilon \psi} (v_i - v_k) R_{i,k} G \end{aligned} \tag{5}$$

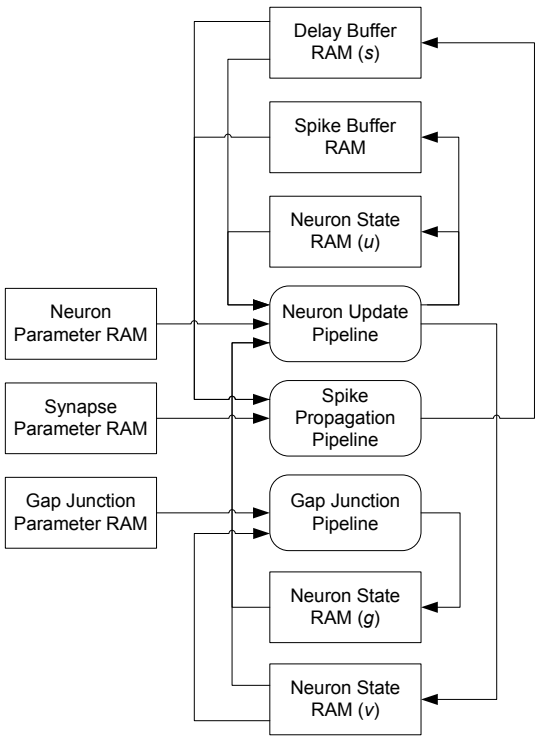

Fig. 1. A core for processing a single group of neurons

where ψ is the set of neurons connected to i by an electrical synapse, $R_{i,k}$ is the electrical conductance between i and k, G is a scaling factor, and $(v_i - v_k)$ is the difference in potential between the two neurons.

IV. Parallel Simulation of Spiking Neural Networks

An expression for the cost of serial simulation of spiking neurons is given by Brette et al. [1]. For a model containing N neurons with average firing rate per neuron of $F_c(N)$, the cost of a single second of simulation is given by

$$c_u \times \frac{N}{dt} + c_c \times F_c(N) \times N \times p_c \tag{6}$$

where dt is the duration of the time bin, c_u is the cost of updating a single neuron, c_c is the cost of propagating a single spike, and p_c is the average number of targeted neurons per spike.

We extend equation (6) to account for parallel simulation over many independent nodes. The additional variable P represents the number of nodes in the system. If neurons are distributed randomly between nodes, we can write the cost of distributed simulation as

$$\begin{aligned} \frac{1}{P} \times \left(c_u \times \frac{N}{dt} \right) + \frac{1}{P^2} \times (c_c \times F_c(N) \times N \times p_c) \\ + \left(1 - \frac{1}{P} \right) \times (c_{tc} \times F_c(N) \times N \times p_c) \end{aligned} \tag{7}$$

where c_{tc} is the cost of transferring a single spike over the network. We assume propagation of spikes between nodes is a serial operation, and nodes update the state of their internal network in parallel.

To include gap junctions we add the variable c_g to represent the cost of propagating a single update over a gap junction connection, and p_g to represent the average number of targeted nodes per gap junction. The variable N_g represents the number of gap junction connected neurons in the simulation. The equation for distributed simulation including gap junctions becomes

$$\begin{aligned}\frac{1}{P}\times\left(c_u\times\frac{N}{dt}\right)+\frac{1}{P^2}\times\left(c_c\times F_c\left(N\right)\times N\times p_c\right)\\+\left(1-\frac{1}{P}\right)\times\left(c_{tc}\times F_c\left(N\right)\times N\times p_c\right)\\+\frac{1}{P^2}\times\left(\frac{N_g}{dt}\times c_g\times p_g\right)\\+\left(1-\frac{1}{P}\right)\times\left(\frac{N_g}{dt}\times c_{tg}\times p_g\right)\end{aligned}\tag{8}$$

We propose that neurons should be distributed such that distinct networks of gap junction connected neurons are allocated to separate nodes of the system, minimizing the term containing c_{tg} in equation (8). This is consistent with biological evidence [6]. It reduces the network overhead of constant communication between gap junction connected neurons, and allows each gap junction network to be updated in parallel with networks on all other nodes.

V. FPGA Design and Implementation

The design is divided into a number of identical cores, each responsible for maintaining the state of a group of neurons in the simulation. The layout for a single core is given in Figure 1, the overall simulator architecture in Figure 2.

At the level of individual groups of neurons, there are three independent operations that can be carried out at each step of the simulation in parallel: updating of neuron state, propagation of spikes within the group, and propagation of gap junction current within the group. We implement each operation as a separate pipeline for a single core. The total time to update the entire model in a given step is the maximum number of cycles required by any single pipeline across all cores. Propagation of spikes and gap junction current between cores occurs in a single sequential operation at the end of parallel update of the model.

The neuron update pipeline implements equations (2), (3) and (5). The synapse and gap junction pipelines generate terms $\sum_{j\epsilon\phi} S_{i,j}F$ and $\sum_{k\epsilon\psi}(v_i - v_k)R_{i,k}G$ in equation (5), representing incoming current to the neuron from chemical synapses and gap junctions. All parameters and state variables for neurons, synapses, and buffered spikes are stored in embedded RAM.

Previous approaches have used a fixed network of connections between neurons [12] or handled spike propagation off-chip to avoid the overhead of handling arbitrary and

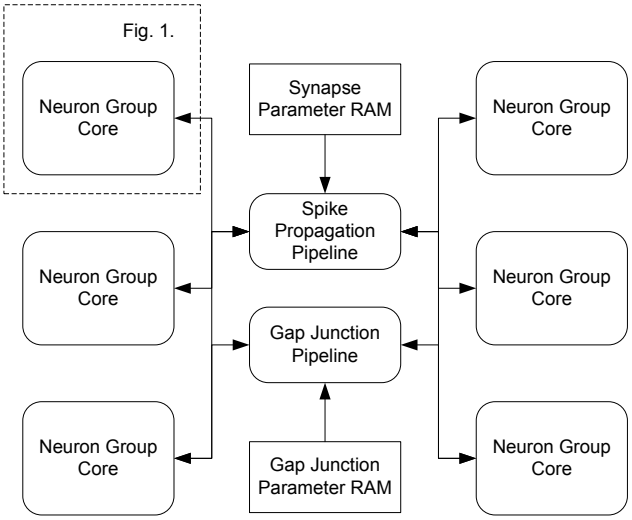

Fig. 2. Overall simulator architecture

dynamically changeable networks of synapses. Fixed connections require reprogramming the FPGA to simulate different networks. In a research environment, where the parameters of a model are often under constant change, it is important that the same hardware can be used to switch quickly between models with different parameters and connectivity.

A limitation of our current implementation is the use of 24-bit fixed point rather than floating point arithmetic. Our focus has been on optimizing the design to utilize the structure of the network and synaptic connections. Further analysis is required to demonstrate that functionally equivalent networks of Izhikevich neurons can be constructed using either fixed point or floating point arithmetic, and determine the fixed point precision required to build such networks. We leave this as an area of future research.

VI. Results

Implementation was carried out in Handel-C using a Xilinx Virtex-5 xc5vlx330t FPGA. This device contains 324 block RAMs of 36Kb each, and just over 50,000 slices each containing four 6-LUTs and four Flip-Flops, totalling over 200,000 logic elements. With eight cores our design occupies 18% of slices and 43% of block RAMs on the chip. We use a clock frequency of 20 MHz.

The size, basic structure, and initial choice of neuron parameters in our test network are based on the model used in [8], and described in the appendix of the same paper. The network consists of $1,000$ Izhikevich neurons. Of the total neurons 80% are excitory and 20% inhibitory, a well known proportion in the brain. The number of synaptic connections and weights are chosen to allow for self-sustaining activity in the network with minimal external input. We compare networks without gap junctions, with gap junctions randomly distributed between inhibitory neurons, and with clustered gap junctions where no gap junctions connect neurons on different cores.

Simulation times for a single second of data, running the simulation over an increasing number of cores, are given in

Table I. Modeling distinct clusters of gap junctions gives a significant performance improvement over random connections, with between 1.3 times speedup for two cores to 5.7 times for eight cores.

For comparison we test the performance of our design against a similar GPU implementation, the NeMo tool under development at the Imperial College Cognitive Robotics group [4] modified to support gap junctions. Results for simulating an equivalent 1,000 neuron network with clustered gap junctions, on an increasing number of cores of a Telsa C1060 GPU, are shown in Table II. Each data point is the average of ten runs. For a network of this size the FPGA implementation shows comparable performance to NeMo over eight cores, although a larger network is required by the NeMo simulator to take full advantage of all cores on the GPU. It is likely that the relative performance of the two platforms will depend on the size and connectivity of the network.

It is important to consider the relative power consumption of the two devices, with the GPU consuming over 180W and the FPGA implementation under 10W. We view biologically inspired control systems for robotics and embodied spiking neural networks as important areas of future research. For any mobile robotics application power consumption is a key issue, and the choice of platform will vary between applications.

CPU results for an equivalent 1,000 neuron network with clustered gap junctions are shown in Table III. We modified existing simulation tool NEST [3] to support both the Izhikevich neuron model and gap junctions. Simulations were performed on the Imperial College High Performance Computing (HPC) system, a cluster of 150 dual core, dual CPU, 3.6GHz Intel Xeon Dell blade servers, connected by InfiniBand. Each dual CPU node is configured with 1.5 GB of memory. Table IV shows the results of clustering gap junction connections for a larger 100,000 neuron network, structured in the same manner as the 1,000 neuron network described above.

TABLE I
FPGA SIMULATION TIMES(SEC) FOR A 1,000 NEURON NETWORK, 1 SECOND OF SIMULATED DATA

	Number of Cores						
	2	3	4	5	6	7	8
Without	0.027	0.024	0.023	0.022	0.022	0.021	0.023
With	3.204	2.036	1.679	1.488	1.429	1.230	0.983
Clustered	2.501	1.123	0.635	0.411	0.297	0.220	0.173

TABLE II
GPU SIMULATION TIMES(SEC) FOR A 1,000 NEURON NETWORK WITH CLUSTERED GAP JUNCTIONS, 1 SECOND OF SIMULATED DATA

	Number of Cores				
	2	4	5	8	10
Clustered	0.404	0.251	0.229	0.174	0.160

TABLE III
CPU SIMULATION TIMES(SEC) FOR A 1,000 NEURON NETWORK WITH CLUSTERED GAP JUNCTIONS, 1 SECOND OF SIMULATED DATA

	Number of Processors				
	4	8	12	16	20
Clustered	4.205	4.257	4.626	5.063	5.925

TABLE IV
CPU SIMULATION TIMES(SEC) FOR A 100,000 NEURON NETWORK, 10 SECONDS OF SIMULATED DATA

	Number of Processors				
	4	8	12	16	20
Without	1352.913	713.260	494.850	410.096	347.341
With	1937.752	1821.136	1553.673	1440.809	1430.271
Clustered	1714.888	900.549	635.799	484.435	411.788

VII. SUMMARY

We derive expressions for the cost of distributed simulation of spiking neural networks with and without gap junctions. Combined with biological evidence for the structure of gap junction networks, this suggests partitioning a neural model into clusters based on gap junction connections as a means of improving simulation performance. Implementation on an FPGA and in an existing CPU based simulation tool demonstrate a speedup of up to 5.7 and 3.5 times respectively for clustered versus non-clustered networks. Modifications to an existing GPU based simulation tool to support gap junctions show comparable performance to the FPGA for simulation of clustered networks. We suggest FPGA-based simulation of spiking neural networks to include realistic effects of gap junctions, especially for research in power limited applications such as robotic control systems.

Acknowledgement. We thank EPSRC and Xilinx for their support.

REFERENCES

[1] R. Brette et al. Simulation of networks of spiking neurons: A review of tools and strategies. *J Comput Neurosci*, 23(3):349–398, Oct 2007.

[2] M. R. Deans, J. R. Gibson, C. Sellitto, B. W. Connors, and D. L. Paul. Synchronous activity of inhibitory networks in neocortex requires electrical synapses containing connexin36. *Neuron*, 31(3):477–85, Aug 2001.

[3] M. Diesmann and M. Gewaltig. Nest: An environment for neural systems simulations. In *Forschung und wisschenschaftliches Rechmen*. Beiträge zum Heinz-Billing-Preis, 2001.

[4] A. K. Fidjeland, E. B. Roesch, M. P. Shanahan, and W. Luk. NeMo: A platform for neural modelling of spiking neurons using GPUs. *Proc. 20th IEEE Conf. Application-specific Systems, Architectures and Processors*, 2009.

[5] S. Furber, S. Temple, and A. Brown. On-chip and inter-chip networks for modeling large-scale neural systems. In *Proc. IEEE International Symposium on Circuits and Systems, ISCAS-2006*, May 2006.

[6] S. Hestrin and M. Galarreta. Electrical synapses define networks of neocortical gabaergic neurons. *Trends Neurosci*, 28(6):304–9, Jun 2005.

[7] E. M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–70, Sep 2004.

[8] E. M. Izhikevich. Polychronization: computation with spikes. *Neural computation*, 18(2):245–82, Feb 2006.

[9] E. M. Izhikevich and G. M. Edelman. Large-scale model of mammalian thalamocortical systems. *Proc Natl Acad Sci USA*, 105(9):3593–8, Mar 2008.

[10] L. Maguire, T. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin. Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing*, 71(1-3):13–29, Dec. 2007.

[11] G. M. Shepherd. *The Synaptic Organization of the Brain*. Oxford University Press, 2003.

[12] D. B. Thomas and W. Luk. FPGA accelerated simulation of biologically plausible spiking neural networks. *IEEE Symposium on Field-Programmable Custom Computing Machines*, Apr 2009.