ESL Design of Customizable Real-Time Neuron Networks

Conference Paper · October 2014

DOI: 10.1109/BioCAS.2014.6981815

CITATIONS
6

READS
56

6 authors, including:

Carlo Galuzzi
Maastricht University
45 PUBLICATIONS **414** CITATIONS
SEE PROFILE

Amir Zjajo
Delft University of Technology
102 PUBLICATIONS **313** CITATIONS
SEE PROFILE

Georgios Smaragdos
Erasmus University Rotterdam
20 PUBLICATIONS **104** CITATIONS
SEE PROFILE

Rene Van leuken
Delft University of Technology
95 PUBLICATIONS **367** CITATIONS
SEE PROFILE

ESL Design of Customizable Real-Time Neuron Networks

Martijn van Eijk[♮], Carlo Galuzzi[♮], Amir Zjajo[♮], Georgios Smaragdos[♯], Christos Strydis[♯], and Rene van Leuken[♮]
[♮] Circuits and Systems Group, Delft University of Technology, The Netherlands
[♯] Erasmus Medical Center, The Netherlands
{c.galuzzi, a.zjajo, t.g.r.m.vanleuken}@tudelft.nl – {g.smaragdos, c.strydis}@erasmusmc.nl

Abstract—**In this paper, we present the design and implementation of an *Inferior-Olivary Nucleus (ION)* network on an *FPGA* device. Compared with existing neuron networks, the proposed design allows to easily customize the network topology and implement existing as well as ad-hoc topologies, in order to explore different levels of connectivities between the cells. Starting from the model of an *ION* cell, the model has been optimized and an *ION* network has been designed and implemented in multiple steps. By using the Xilinx Vivado Suite, the design has been synthesized and mapped on a Virtex 7 XC7VX550T FPGA device. Experimental results show that a network of 48 *ION* cells can be simulated in brain real-time using double floating-point arithmetic, which allows to precisely simulate the network's behavior.**

I. Introduction

The human brain is a very complex system, which possesses billions of nerve cells, also known as neurons. These cells, which are heavily interconnected, constitute altogether the so-called *Neural Network (NN)*. Over the years, tremendous advances in neuroscience gradually led to the creation of realistic mathematical models of these cells and their complex interconnected networks, which do not simply mimic biological behavior in an abstract way, but simulate it with a great level of detail, as in the case of *Spiking Neural Networks (SNNs)*. *SNNs* are a particular kind of *NN* with a high level of realism in the simulation of the neurons. In this kind of networks, information is not just encoded by the firing rate of each neuron in the network, as it happens in classical *Artificial Neural Networks (ANNs)*, but by the transfer of spikes as well [1].

Due to the *SNNs*' ability to model additional neuron characteristics and adapt them according to spike-train amplitude, frequency, and precise arrival times, *SNNs* can have greater computational and predictive power compared to *ANNs* [2]. However, this comes at a cost: the high level of realism of *SNNs* and their complexity require a considerable amount of computational resources which, in turn, limit the size of the achieved simulated networks. The main challenge in building complex and biologically accurate *SNNs* lies largely in the high computational and communication loads of the network simulations. Furthermore, biological *NNs* execute these computations with massive parallelism, something that conventional CPU-based execution cannot cope very well with.

Field-Programmable Gate Arrays (FPGAs), although slower than *Application-Specific Integrated Circuits (ASICs)*, thanks to the high parallelism provided by the hardware, are capable of providing enough performance for real-time and even hyper-real-time neuron simulations. Additionally, the reconfiguration property of the *FPGAs* provides the flexibility to modify brain models on demand. This flexibility is substantially enhanced by the use of high-level synthesis tools, which speed up the development process.

In this paper, we present the *Electronic System Level (ESL)* design of a network of extended *Hodgkin-Huxley (HH)* models of a particular type of nerve cell (neuron), the *Inferior-Olivary Nucleus (ION)*, on an *FPGA*. Starting from the model of a single *ION* cell, we designed and implemented an *ION* network using *SystemC*, which allows to describe and simulate the behavior of the entire system using functional programming. *High-Level Synthesis (HLS)* tools, which enable rapid modification and re-synthesis of the model and automates the process of optimal hardware architecture selection, have been used to extract the parallelism of the model, synthesize and map the proposed design on an *FPGA* device. The network can be fine-tuned, as parameter changes can be rapidly carried out by a reconfiguration of the *FPGA*.

A highly optimized hardware implementation has already been proposed in [5] through *Vivado HLS* (*C* flow). While it offers better synthesis results in a head-on comparison with the current work, it does not allow exploration of different *ION* network dimensions and different interconnect architectures, short of re-synthesizing of each different design point. Additionally, our current *SystemC* implementation, through the use of the *TLM* paradigm, allows for rapid simulation of different network instances, as well as rapid profiling of different interconnects (e.g. bus, NoC), with minimal design overhead. More precisely, the main contributions of the work presented in this paper are the following:

- the implementation of a *brain real-time network* of 48 *ION* cells using double floating-point arithmetic, which allows to precisely simulate the network's behavior;
- the possibility to customize the network topology and implement existing, as well as ad-hoc topologies, in order to explore different levels of connectivities between the *ION* cells;
- the description and optimization of the *ION* network, modeled with *SystemC*, and mapped on an *FPGA* to exploit the inner parallelism of the reconfigurable hardware and boost the performance of simulation.

In the following sections, the *ION* model and the implementation of an *ION* network are presented in more details.

II. Model Description

The first version of the *ION* model was built using *Matlab* [3]. To overcome the performance limitations of *Matlab*, the *ION* model was ported to C. The *ION* C model is our starting point. In the following, we first present, in more detail, the model and its optimizations, in order to exploit the parallelism

The work presented in this paper is supported by the European Union and the Dutch government as part of the CATRENE program under the Heterogeneous INCEPTION project.

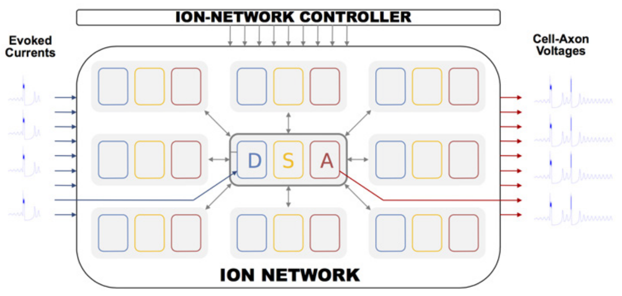

Figure 1. Configuration of the proposed *ION* network [courtesy of S. Isaza, Erasmus Medical Center (NL)]; *D*, *S*, and *A* represent the three compartments of the nerve cell (neuron), namely the *Dentrite*, the *Soma*, and the *Axon*.

provided by hardware. After that, we detail the design and implementation of the *ION* network on the *FPGA*.

A. The ION Model

The *HH* model describes how action potentials are initiated and propagated in nerve cells (neurons) and it can be adapted to mimic the behavior of a preferred type of cell [7]. The model consists of a set of nonlinear differential equations, which is used to approximate the electrical characteristics of the cell. It is a continuous time model describing the behavior of the cells over time, which, at each time step, calculates a set of parameters for each cell. For the calculation of a single parameter, one exponential function and several multiplications and divisions need to be carried out. Moreover, to reach maximum precision and realistic signal representation, the use of floating-point arithmetic is essential. These requirements make the computational complexity of the model very high. Nonetheless, as all neurons in a *SNN* function individually, the behavior of multiple neurons can be evaluated concurrently to decrease the overall computation time. From [3], we know that the minimum simulation output interval to achieve a realistic representation of the *ION* behavior is determined at $50\mu s$, known as the *brain real-time*. This means that a single simulation step takes $50\mu s$ to produce a brain-real time output. When the output is generated, the calculated chemical values are fed back as initial input values, as they define the new state of the model.

In a *NN*, each *ION* is highly interconnected (via *gap junctions*) with a number of neighbor cells, which affect its (electro-chemical) status. As a result, it is very important to correctly describe the cell distributions in the network. Due to the "regular" distribution of the cells, it is possible to represent the *IONs* in a grid, which, in turn, allows the use Cartesian coordinates to uniformly describe the neighbor cells. Figure 1 shows a 3×3 cell network with interconnections between the cells. For illustrative purposes only the central cell is drawn with all connections to the neighbors although, in the model, all cells are connected to their neighbors. For the same reason, only a single input and output wires are drawn, which represent the input and output for every single cell.

B. Code Optimization

The *ION* model, originally written for a *General Purpose Processor (GPP)* architecture with parallel execution in mind, was not optimized for highly parallel architectures, like the *FPGA*. For this reason, before implementing the *ION* network in hardware, the model needed to be optimized for the targeted architecture, so to make optimal use of the parallelism provided by the hardware. A number of optimizations have been carried out to improve the model. Among others, the main optimizations included a reduction of the memory accesses and mathematical simplifications.

When *C* code is synthesized for an *FPGA*, usually, arrays are implemented in a *Block RAM* and variables are implemented as registers. As memory usage should be avoided when it comes to computation performance, memory accesses and, thus, array accesses, should be minimized by removing the superfluous ones. As a result, the code has been optimized in such a way that the intermediate values are stored in registers and the updated parameters are stored in memory. In this way, it is possible to read the new initial values directly from the same memory locations in which they are stored in. Intermediate values are stored in temporary variables and they are written back to memory only at the end of the calculation loop. Both previous and new arrays are stored in registers to completely avoid memory look-ups. As registers require more *FPGA* area compared to *RAM* but less cycles to read/write, this optimization can provide a trade off between speed up and area.

Once the model has been optimized in terms of memory accesses, a number of mathematical simplifications has been applied in order to reduce the number of operations carried out to simulate the behavior of the *IONs*. By affecting only the computational complexity and performance of the model without compromising its precision, via mathematical manipulations, we were able to effectively reduce the number of multiplications, the most time consuming operations in the model. After these and other optimizations (e.g., substitution of floating-point division-by-constant by multiplication-by-constant), have been applied to the model, it is possible to proceed with the design and implementation of the *ION* network in hardware, described in the following.

C. Design and Implementation of the ION Network

The structure of the *ION* network (see Figure 1) led to the design of several individual (neural) computation units interconnected with each other. The computation units, which can be seen as actual cells mimicking the functionality of the *IONs*, are implemented as separate units on the *FPGA*. To incorporate the gap-junction influence in the *ION* network, every cell needs to be able to transfer its data to the neighbor cells that are affected by it. Due to the finite amount of resources available on the *FPGA*, only a limited number of cells can be implemented in hardware. Similarly, as mentioned before, the precision has a large impact on the hardware resources used for the implementation of the model. Although double floating-point operations take more resources compared to single floating-point operations, in this work, we used double precision operations so to implement a realistic and accurate model of the *ION* network.

THE PHYSICAL CELLS: The (neural) computation units, from now on referred to as *Physical Cells (PhCs)*[1], are task specific machines implemented in hardware with a dedicated

[1]The computation units are called *Physical Cells* to recall that they are physically implemented in hardware and that the output of their computations mimic the actual cell behaviors.

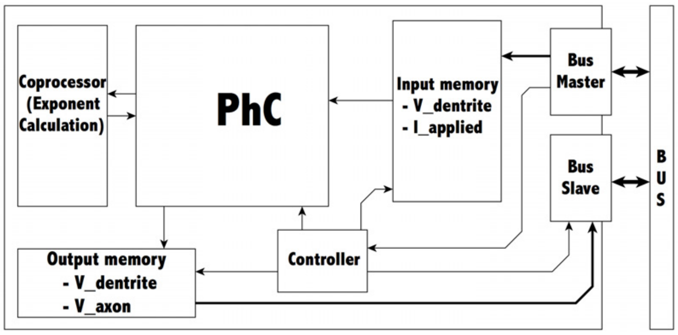

Figure 2. A *Physical Cell (PhC)* computation module with separate exponent computation unit.

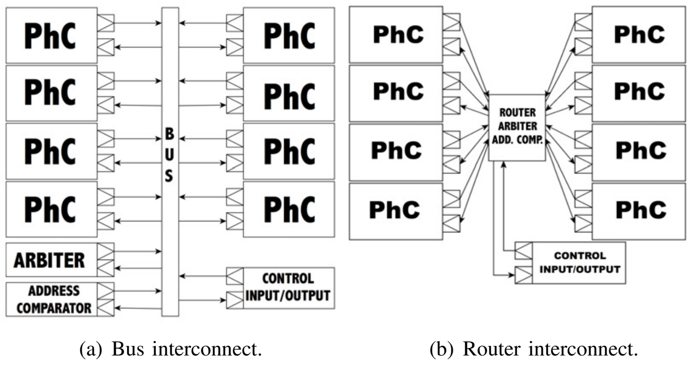

Figure 3. Implementation of the all-to-all *ION* network.

memory (see Figure 2). In this way, each unit can function independently from the other ones, which is essential to guarantee the simultaneous simulation of multiple cells. As expected, when implemented in hardware, the cells occupy a certain (fixed) amount of resources, which depends on the implementation and design choices and, therefore, only a limited number of cells can be simulated on any available *FPGA*. To maximize the use of the hardware in terms of parallelism and performance, the *PhCs* are time-multiplexed to allow the implementation of a much larger *ION* network. This is possible as a *PhC* takes only a fraction of the brain real-time to simulate an *ION*'s behavior. As a result, each *PhC* can be reused multiple times within the brain real-time boundary. The number of times corresponds to the *Multiplexing Factor (MF)*. This means that, we are capable of simulating a much bigger network.

INTERCONNECT: In our network model, we assume that each cell is interconnected with any other cell. However, in reality, the effects that neighbor cells have on each other are a function of the distance between the cells. This means that it is possible to explore different kinds of network connectivities and to assess their effects on the entire network behavior. As a result, the network model should be configurable. In our implementation, each *PhC* has a private connectivity table to look up, during the computation, for the effects of its neighbor cells. This means that the output of each cell should be distributed among the other cells according to a certain network topology. Before defining this topology, we need to analyze the requirements. First of all, the data produced by each cell should be available, at the same time, to any other cell in the network. Additionally, the data communications should have equal priority. As a result, the data is sent in an all-to-all fashion and, to avoid collisions on the network due to simultaneous data dispatching, an arbiter for traffic control has been implemented.

As depicted in Figure 3(a), as a proof of concept, a shared bus has been used as actual interface between the *PhCs*. No extra logic is needed to break down the 64 bit floating-point variables in single bits for serial dispatching. The shared bus is practically implemented as a router connected to all *PhCs* (see Figure 3(b)), which arbitrates the traffic as it is passing. In our design, 32 bit addresses are chosen, which are divided in 2×16 bit to address up to 2^{16} cells. In this way, it is possible to simulate large array networks.

To implement the appropriate arbitration scheme, a certain number of details needs to be taken into account. First of all, all traffic should be concentrated between the computation phases to minimize delays in dispatching the data. All the network nodes (*ION cells*) should act as both master and slave, as each node (*ION cell*) sends data to its neighbors (master) and receives data from them (slave). Each node individually transfers its data and the amount of data transferred between the nodes is considered equal for all nodes. Finally, assuming that all computations should start simultaneously, they should also end simultaneously, as each node in the network runs the same algorithm. Based on these considerations, despite the effects of *IONs* on each others diminish with distance, which means that different amount of data are produced and dispatched by each node, we assume that the same amount of data is produced by each node and dispatched to its neighbors, with equal priority on the network communication channels. As a result, a simple round-robin scheme can be implemented to arbitrate the network traffic.

We implement *PhCs* in hardware, which are time-multiplexed as the computation time of each *PhC* on the *FPGA* is only a fraction of the brain real-time. Each *PhC* executes the same algorithm. As such, all the *PhCs* have the data ready for dispatching at the same time. As a result, due to the all-to-all communication policy, the network traffic will be, clearly, very heavy. In order to use the bus in the most efficient way, the data is sent during the computation phase. In a time-multiplexed *PhC* situation, the next *ION* computation uses data independent from the previous computation. Therefore, each time a *PhC* ends a computation, it can immediately start the computation of the subsequent *ION* cell, as the data needed for the computation is already available. Only when a new simulation step needs to start, the *PhC* needs to receive data from the neighbor cells. When all data is received, the computation of the new simulation step can be started. If data dispatching is delayed, the new computation should wait all the data before starting.

III. EXPERIMENTAL RESULTS

The hardware implementation of the proposed network is accomplished in multiple steps. First of all, the optimized *ION* C model (see Section II.B) of a single cell is implemented using *SystemC* in order to obtain the hardware resource usage of a single cell. This allows to determine the maximum size

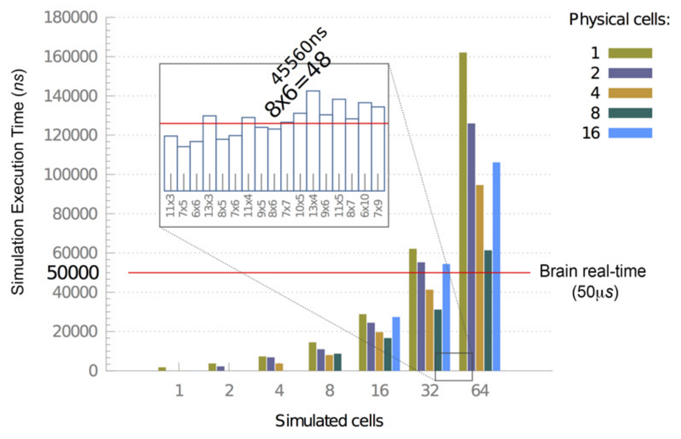

Figure 4. Simulated cells obtained using different combinations of PhCs and MF. The targeted architecture is a Xilinx Virtex XC7VX550 FPGA board with a clock frequency of 100MHz. The horizontal red line represents the brain real-time threshold ($50\mu s$, see [3]).

of the *ION* network implementable in hardware, taking into consideration that *PhCs* are time-multiplexed (see Section II.B).

Based on this information, a *SystemC TLM* model is implemented to simulate a hardware implementation of the *ION* network. The *SystemC TLM* model allows to simulate different kinds of network topologies in a fast way, thanks to the high abstraction level and, at the same time, it allows to test the feasibility of ad-hoc network topologies as well as different level of connectivities between the cells. As mentioned in Section II.C, as a proof of concept, in this work, we use a shared bus as the actual interface between the *PhCs*. However, different network topologies can be easily implemented and the overall effect on the *ION* network evaluated. Finally, in order to implement the *ION* network model in hardware, the *SystemC TLM* model needs to be converted in a *SystemC* model. Based on the *SystemC* model, by using *Vivado HLS*, the network has been synthesized into an *RTL* description mapped on an *FPGA* device. After that, the *FPGA* implementation has been done using *Vivado IDE*. In the following, we evaluate the *ION* network behavior based on the proposed design and implementation choices.

The *ION* network has been synthesized and mapped on a reconfigurable architecture, in order to see the benefits of our design methodology. In our experiments, we targeted a *Virtex 7 XC7VX550 FPGA* board with a clock frequency of 100Mhz [4]. As mentioned before, *Vivado HLS* and *Vivado IDE* (*Vivado Suite 2013.4*) [6] have been used for the synthesis and implementation of the design.

In order to identify the highest number of *ION* cells to map on the given *FPGA*, various combination of *PhCs* and MFs have been taken into consideration. Figure 4 shows different configurations in order to simulate a particular number of cells. The corresponding number of *PhCs* per number of simulated cells is given and the optimal execution time is the lowest bar in each cluster. Starting from 16 simulated cells, the optimal number of *PhCs* is 8. The magnification of the part between 32 and 64 simulated cells shows the lowest-execution-time

Table I
RESOURCE USAGE OF THE *ION* NETWORK MODEL IMPLEMENTATION WITH 8 PHCS AND A TIME-MULTIPLEXING FACTOR OF 6 ON A *Virtex 7 XC7VX550 FPGA* WITH A CLOCK FREQUENCY OF 100MHZ.

FF	LUT	MEM. LUT	I/O	BRAM	DSP 48	BUFG
28%	74%	1%	12%	2%	48%	6%

configurations for the possible numbers of simulated cells. This is done to determine the maximum number of cells which can still be computed within the brain real-time deadline (the horizontal red line in Figure 4). The largest number of simulated cells that meets the brain real-time requirement is 48 simulated cells with $45560ns$ in a configuration of 8 *PhCs*, each one time-multiplexed 6 times. As shown in the figure, the 7×7 configuration just crosses the boundary with $50280ns$. The resource usage of the proposed design, according to *Vivado HLS*, is presented in Table I. The estimated resource usage of the sythesized model is very predictable. When the number of *PhCs* is increased, the resources grow at the same rate. When increasing the *MF*, the number of *FFs* increases, as expected, for extra variable storage in the registers, and the number of *LUTs* increases, in order to implement the needed extra muxing logic.

IV. Conclusions

In this paper, we presented the design of an *ION* network and its implementation on an a reconfigurable architecture. Starting from the *C* model of an *ION* cell, the model has been optimized and an *ION* network has been designed and implemented on an *FPGA* device using *SystemC*, which allowed to describe and simulate the behavior of the entire system using functional programming. Via our design and optimizations, we were able to implement a realistic (brain real-time) *ION* network of *48 ION* cells. By exploiting the parallelism and performance of computation of the *FPGA* device, the cells has been implemented as 8 *PhCs*, time-multiplexed 6 times, for a total of *48 ION* cells implemented on the *FPGA*. Additionally, the proposed design allows the implementation of any network topology and/or connectivity strategy, which is essential in order to test their effects on the behavior of the entire network.

References

[1] G. Wulfram and W. Werner, *Spiking Neuron Models*, Cambridge University Press, 2002

[2] W. Maass, *Noisy Spiking neurons with Temporal Coding have more computational power than sigmoidal neurons*, Neural Information Processing Systems, pp. 211-217, 1996

[3] J. R. de Gruijl, P. Bazzigaluppi, M. T. G. de Jeu, and C. I. de Zeeuw, *Climbing fiber burst size and olivary sub-threshold oscillations in a network setting*, PLoS Computational Biology, vol. 8, no. 12, pp. 1-10, 2012

[4] Xilinx, *http://www.xilinx.com/*

[5] G. Smaragdos, S. Isaza, M. Van Eijk, I. Sourdis, and C. Strydis, *FPGA-based Biophysically-Meaningful Modeling of Olivocerebellar Neurons*, in International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 89–98, 2014

[6] Vivado Design Suite - Xilinx, *http://www.xilinx.com/products/design-tools/vivado/*

[7] A.L. Hodgkin and A.F.Huxley, *A quantitative description of membrane current and its application to conduction and excitation in nerve*, The Journal of physiology, vol. 117, no. 4, pp. 500-544, 1952