Multi-Chip Dataflow Architecture for Massive Scale Biophysically Accurate Neuron Simulation

Hofman, Jaco; Zjajo, Amir; Galuzzi, Carlo; van Leuken, Rene

DOI
10.1109/embc.2016.7592053

Publication date
2016

Document Version
Final published version

Published in
Proceedings - 38th Annual IEEE International Conference of the Engineering in Medicine and Biology Society (EMBS)

Multi-Chip Dataflow Architecture for Massive Scale Biophysically Accurate Neuron Simulation

Jaco Hofmann, Amir Zjajo, Carlo Galuzzi, and Rene van Leuken

Abstract— **State-of-the-art neuron simulators are capable of simulating at most few tens/hundreds of neurons in real-time due to the exponential growth in the communication costs with the number of simulated neurons. In this paper, we present a novel, reconfigurable, multi-chip system architecture based on localized communication, which effectively reduces the communication cost to a linear growth. The system is very flexible and it allows to tune, at run-time, various parameters, e.g. the intracellular concentration of chemical compounds, the interconnection scheme between the neurons. Experimental results indicate that the proposed system architecture allows the simulation of up to few thousands biophysically accurate neurons over multiple chips.**

I. Introduction

The biologically accurate simulation of neuron networks has two main implications, i.e. it allows us to understand how the brain processes information without having to perform in vivo experiments, and it can lead to the design of brain implants capable of restoring damaged, destroyed, or even missing parts of the brain. Several realistic mathematical models have been proposed for various nerve cells and their complex interconnected networks [1]-[2]. Consequently, this allowed an increase in the biological accuracy of the simulated neural networks, e.g. the spiking neural networks, where information is encoded by both the firing rate and the transfer of spikes [1], [3]. One of the main characteristics of the neurons is that, although they are heavily interconnected, they function individually. Accordingly, to decrease the overall computation (simulation) time, the behavior of multiple neurons can be evaluated concurrently. Very large scale integration (VLSI) designs provide the necessary parallelism but do not allow for altering the neuron model after manufacturing and, thus, are too expensive for the experimental stage. A highly parallel architecture, such as a field programmable gate array (FPGA), provides sufficient hardware parallelism and performance for real-time and even hyperreal-time neuron simulations. Additionally, via (partial) reconfigurations of the hardware, various neuron models, e.g. simple models (integrate and fire [4], Izhikevich [5]), or Hodgkin-Huxley [6], simplified Hodgkin-Hoxley [7], extended Hodgkin-Hoxley [8], as well as different network topologies and cell interconnect schemes can be simulated.

This research was supported in part by the European Union and the Dutch government, as part of the CATRENE program under Heterogeneous INCEPTION project.

J. Hofmann is with the Embedded Systems Group, Darmstadt University of Technology, Hochschulstrasse 10, D-64289, Germany.

A. Zjajo and R. van Leuken are with the Circuits and Systems Group, Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands.

C. Galuzzi is with the BMI Research Group, Maastricht University, Bouillonstraat 8-10, 6211 LN, Maastricht, The Netherlands.

The simulation of a network spread over multiple FPGA devices is a demanding task due to the identification of appropriate partitioning schemes, the limited speed of the FPGA interconnects, as well as their limited number. In this paper, we propose an efficient multi-chip dataflow architecture, which exploits data locality and minimize network communications over one or multiple FPGA devices. The system methodology uses double floating-point arithmetic for the most biologically accurate cell behavior simulation, and offers easy implementation of various neuron network topologies, cell communication schemes, as well as models and kinds of cells. All parts of the system are generated automatically based on the neuron interconnection scheme in use. The system is cycle accurate, flexible, and it allows to tune, at run-time, various parameters, e.g. the interconnection scheme between the neurons, the intracellular concentration of different chemical compounds (ions), which affect how action potentials are initiated and propagate.

II. Multi-Chip Dataflow Architecture Design

Communications between neuron cells are a function of their distances: cells placed close to each other in the network (neighbor cells) communicate more than cells placed far away from each other [1]. Accordingly, there are two aspects, which need to be considered: the network topology and the cell communication scheme. Different network topologies can be implemented; starting from an all-to-all network, we can progressively reduce the number of connections between the cells and see the effects on the entire network. Similarly, different cell communication schemes can be considered, even with the same network topology. Subsequently, various trade-offs between number of cells and number of connections/communications can be examined.

In the proposed network (Fig. 1), the neuron cells are connected with decreasing probability the further they are apart. For neuron cells that are placed close to each other in the network, we introduce a structure called (neighbor) cluster, as illustrated in Fig. 2. Each cluster consists of either 2 or 4 individual computation units, called physical cells (*PhC*s)[1], arranged around a shared memory for storing all the communication data needed by the *PhC*s. The physical cells considered are based on an extended Hodgkin-Huxley model [9] for an accurate representation of the inferior-olivary nucleus cell state [8], [10]. However, the neuron network structure is not limited to any neuron model, and diverse neuron models can be easily used as well. Each *PhC* receives its input data from the cells it is connected to.

[1] The computation units are called physical cells (*PhC*) to recall that they are physically implemented in hardware, and that the outputs of their computations mimic the actual inferior-olivary nucleus (ION)-cell behavior. See [10] for additional information.

978-1-4577-0220-4/16/$31.00 ©2016 IEEE

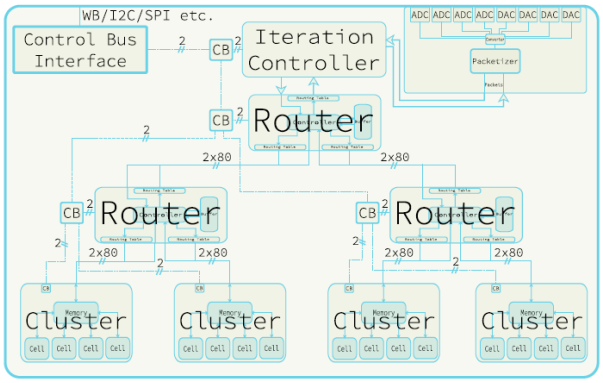

Fig. 1: The system overview. The computing elements (the *PhCs*) are grouped inside a cluster to make communication between neighboring cells fast. These clusters are connected in a tree topology network on chip. The router fan-out in this case is two and can be changed according to the requirements of the implementation. The same holds true for the number of *PhCs* in any cluster.

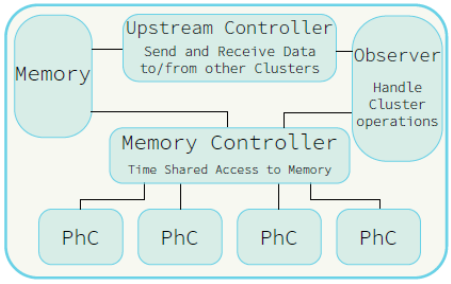

Fig. 2: A cluster as used in the proposed system. The cluster allows for instant communication between any cells that are located inside the cluster. Each of the *PhC* time shares the calculation of multiple cells. The data of the cells that might be needed by other clusters is forwarded into a network-on-chip.

Each time a physical cell in a cluster completes a calculation step, its output is written into the dual-port shared memory: to interface the cluster with the network, and for the time-sharing within the cluster. Every *PhC* gains read and write access to the memory using a round-robin scheme. The cluster is controlled by three controllers. The first controller handles all the book keeping, including monitoring the start signals that are received from the network, and issues a done packet to the round controller after all data for the next iteration is present in the memory. The second controller observes which packets arrive from the network, stores them into the memory, and informs the book keeping controller that all needed packets have arrived. The last controller sends all packets upstream into the network and controls the round-robin memory access. Additionally, it stalls the calculation if the upstream router is not able to receive more packets.

In the proposed communication scheme, the majority of the communications is performed within the clusters, i.e. the data from other clusters is not required by most *PhCs*. If communication with other clusters is required, e.g. with cells located at the cluster boundaries, the output of these cells is forwarded upwards into the interconnection network. To incorporate the cell communication scheme into the network, we implemented a cycle-free, tree-based network topology. In this network each router has 2 to n children and each child can be either a cluster or another router. The leaves of the tree-based network consist of only clusters.

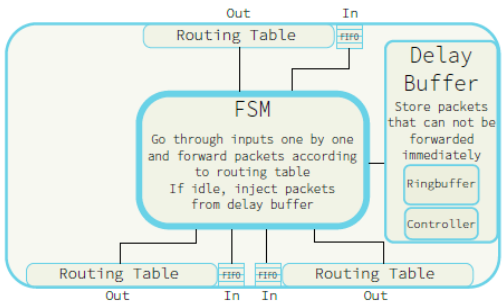

Fig. 3: Diagram of a router as used by the proposed system. The routers are arranged in a tree topology. Each router has n children and except for the root router one upstream router. The router reads input packets from the input first-in, first-out buffer (FIFO). Based on routing tables it determines where the packets have to be forwarded. If a receiving FIFO is full, the packet is placed in the delayed buffer to be forwarded when the receiving router is free again.

The number of children a router can have is homogeneous throughout the system. The clusters are unaware of the connections inside the network and forward the output of every cell calculated within a cluster to the (upstream) router they are connected to. The data produced by each cell in the network is combined with the cell identification number in a packet, which is injected into the network. Each router (Fig. 3) decides, based on a static routing table (which reflects the way the cells communicate), in which direction i.e. to which cells, the packet has to be forwarded to: to all outputs and/or only to a subset, depending on the cell connections inside the system. Since no packet is allowed to be dropped, packets that cannot be forwarded right away, i.e. the receiving buffer is full, are stored for delayed delivery. The width of this delayed buffer is $b_p+[\log_2(n_o)]$ bit, where b_p is the amount of bits for a packet, and n_o is the amount of outputs of the router. The depth of the delayed buffer is determined empirically; a depth of 240 is suitable even for high usage scenarios with all-to-all connections. To avoid cases that the router continuously tries to deliver delayed packets to full routers, new packets always have precedence over the delayed ones. The routing tables are generated during system initialization from the adjacency matrix that describes the cell connections. The root router of the tree is connected to the round controller, which controls the iterations of the system. Since packet forwarding is not aware of the complete network connectivity, the components are efficient and with limited overhead.

The input and output module interface the system with the outside world. Each of the inputs receives a unique identification number starting from n_c+2, where n_c is the number of cells in the system, and at each system iteration these packets are injected into the system. In return, the outputs receive packets from specific cells. Additional interface is the control bus, i.e. I^2C or SPI, for run-time control of the system. Each component can be addressed in a uniform fashion through a control bus, i.e. any router in the system, any cluster, any physical cell, any shared cell level. All routing tables can be modified at run-time, in order to add or remove connections between the cells. In a multiple FPGA scenario, two modes are available to support the control bus: either each FPGA is individually controlled, or one FPGA functions as a main node and redirects the network traffic to the appropriate FPGA.

The speed between the different FPGAs largely depends on the techniques used for transmissions and can vary from faster than a transmission between routers inside the FPGA for very sophisticated (and expensive) interconnects, to many times slower for simple or low power interconnect [11]. As the communication frequency decreases closer to the root of the network tree, multiple FPGAs can be connected at the highest level without significant impact on performance. However, while applying the tree topology on multi-FPGA systems and adding another tree layer promises easy extendibility, the restricted connection possibilities of each FPGA and need for an extra FPGA for routing between the FPGAs containing the cluster, severely limit their use. Accordingly, as most communications happen between neighboring FPGAs, the FPGAs are connected in a ring based topology (Fig. 4). Furthermore, the ring topology generation and administration of the routing tables is less complex. For synchronization between the clusters, one of the FPGAs contains a controller that handles all the synchronization packets. In large systems this, however, results in a large impact on the time needed to complete the iteration. Hence, to prevent this, we use one of the FPGAs as a master. All FPGAs in the system are connected to the master via two wires; the signal does not have to cross multiple stages, the run time is constant for any number of cells, and signal can finish iteration immediately. The master FPGA, in turn, issues the new round signal when adequate.

III. Experimental Results

The system is automatically generated using a human-readable configuration file, which contains all the relevant parameters of the system and can be easily modified allowing exploration of different fan-out values, different cell communication schemes, etc. All simulations are performed with cycle-accurate SystemC, including all calculation and communication latencies, both on- or off-chip. Three different connection schemes are used to simulate the system behavior; all-to-all connections, normal distributed distance based connections, and neighbor based connections.

Fig. 5a) shows the required buffer sizes for the worst-case, the all-to-all connection scheme. The two factors counteract each other: low fan-out results in fast routers but more routers are required in the system; however, slower routers with large fan-out result in less routers in the system, however, each router requires a bigger delayed buffer. A fan-out of 2 results in the lowest size for the delayed buffer. Fig. 5b) highlights the different fan-out in respect to cluster sizes. While small clusters with small routers provide the overall best performance, bigger clusters catch up with big fan-out. The extreme cases with high or low fan-out perform better than the cases in the middle. The cluster size choice is illustrated in Fig. 5c). Due to the fact that all the *PhCs* of a cluster time-share a memory, large clusters are expected to be slower than smaller clusters. However, clusters with more *PhCs* perform worse, especially at a higher number of cells. The difference for 2048 cells between 2 *PhC* clusters and 18 *PhC* clusters is 30%. Comparison between different cluster sizes for different amounts of cells is illustrated in Fig. 6a). At 512 cells the difference between the optimal approach and the 2 *PhC* system is 9.8% and 5.9% for normal and neighbor connection schemes, respectively.

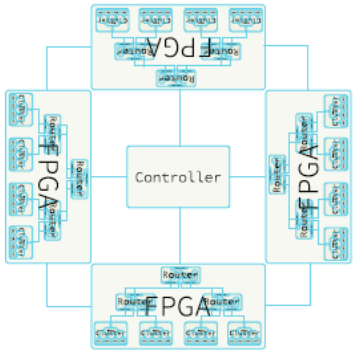

Fig. 4: The single FPGA implementations are connected using a ring topology network. The FPGA are synchronized via a central controller with two wires. One of the wires indicates when the corresponding FPGA complete the operation and the other wire is used by the controller to indicate a new iteration.

The absolute difference in execution time for 32 to 512 cells is 354 cycles, and 191 cycles in the worst case (2 *PhC* system). Correspondingly, this result shows that the proposed system is feasible for one chip systems. To evaluate the scalability of the proposed system to multi-chip systems, the multi-FPGA system is compared to the single chip variant, as well as the optimal approach. The comparisons are done using both the dedicated wire, as well as the packet based synchronization methods. In contrast to the packet based approach, in a dedicated controller method that is connected with each chip, no packet has to cross multiple chip boundaries. The dedicated controller system is 3.84 times faster than the packet based approach at 8 chips with very slow communication (Fig. 6b and Fig. 6c including calculation time). For faster connection speeds on the other hand the difference in communication time is in the order of 1 %.

In Table I, the estimated hardware utilization numbers for the main components of the system are shown in terms of flip-flops (FF) and look-up tables (LUT); smaller components, like the synchronization circuits, are omitted for clarity. The minimum simulation interval to achieve a realistic representation of the neuron-cell behavior is determined as in [8]. Consequently, since each *PhC* can be reused multiple times within the brain real-time boundary[2], and to take advantage of the high level of parallelism and performance of the FPGA, the *PhC*s are time-multiplexed. All results reported are for real-time simulations with double floating point precision for the most biologically accurate representation of a neuron cell behavior.

Component	FF	FF %	LUT	LUT %
Available	692800	100	346400	100 %
PhCs	20488	2.96 %	30235	8.73 %
Router	132	0.01905 %	187	0.05398 %

TABLE I:
Hardware utilization of the most important components of the system on a Xilinx Virtex 7 XC7VX550 FPGA board. The *PhC*s numbers are from [8] and are used to estimate the cluster sizes of the proposed system. Routers sizes are generated by synthesizing a SystemC model of a router using Vivado HLS 2013.4.

[2] The maximum number of cell states that can be computed within the model (in the case of the evaluated, high-detail inferior-olive model, the simulation time step is 50 µs) [10].

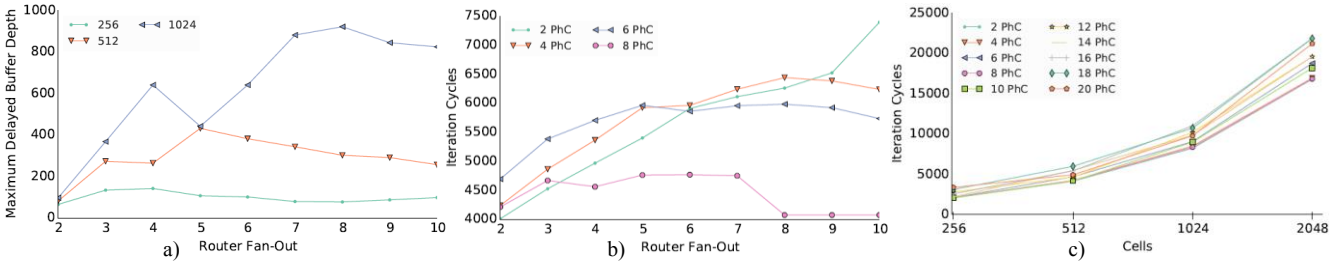

Fig. 5: *a*) Comparison of the system's iteration performance for router input FIFO depths of 4, 8, 16, and 32 elements, *b*) Comparison of iteration performance for 512 cells and different cluster sizes; except for large clusters with eight cells small routers perform better than large routers; the overall best performing configuration consists of small clusters and small routers at two *PhC* per cluster and a fan-out of two, *c*) comparison of systems with different cluster sizes; the routers are kept constant at fan-out of two; clusters with two, four and eight *PhCs* achieve the best iteration times while larger clusters are generally slower especially at 512 cells.

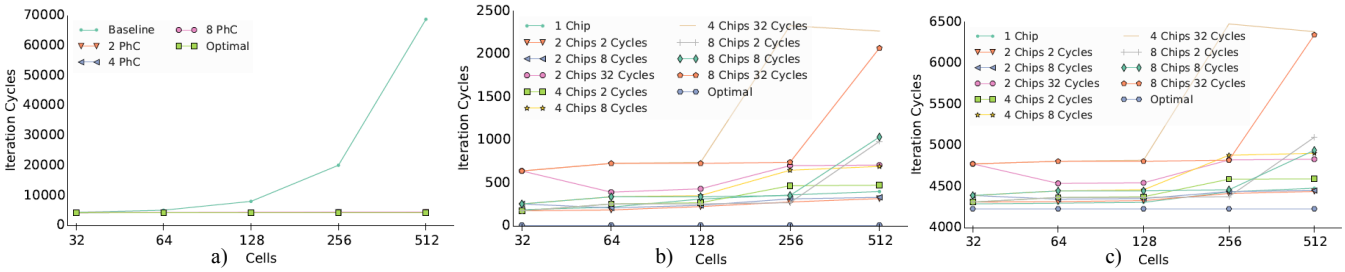

Fig. 6: *a*) Comparison between different cluster sizes for different amounts of cells; the neighbor connection scheme is used and each cell calculation takes 534 cycles [8]; the systems use a router fan-out of two; the baseline design (implemented with a shared bus) is from [8]; all presented configurations scale linearly with the number of cells; the baseline on the other hand scales exponentially, *b*) Comparison between different system configurations utilizing between one and eight chips to simulate a cell amounts between 32 and 512; calculation times are excluded; the neighbor connection scheme is used; multichip systems utilize the dedicated wire based synchronization method, *c*) Similar to *b*) except that calculation times are included as 534 cycles per iteration.

IV. Conclusion

Current neuron simulators, which are precise enough to simulate neurons in a biophysically-meaningful way, are limited in amount of neurons to be placed on the chip, the interconnect between the neurons, run-time configurability and the re-synthesis of the system. In this paper, we propose a system that is able to bridge the gap between biophysical accuracy and large numbers of cells (19200 cells for neighbor connection mode and over 3000 cells in normal connection mode in comparison to 161 neurons in [4] and 400 in [7]). While perfect localization of communication is not possible due to physical constraints, the cells can be grouped around a shared memory in clusters to allow for instantaneous communication. Clusters that are close communicate using only one hop in the network; clusters that are further away communicate less frequently and, consequently, the penalty for taking multiple hops is less severe. Added advantage is that the system can be extended over multiple chips without significant performance penalty. This combination of clusters and a tree topology network-on-chip allows for almost linear scaling of the system. To provide run-time configurability, a tree-based communication bus is used, which enables the user to configure the connectivity between cells and change the parameters of the calculations. As a result, re-synthesizing the whole system just to experiment with a different connectivity between cells is not required. The user has to enter the amount of neurons in the system as well as the desired connectivity scheme. From this information, all required routing tables and topologies are automatically generated, even for multi-chip systems.

References

[1] W. Gerstner, W.M. Kistler, Spiking neuron models: single neurons, populations, plasticity, *Cambridge University Press*, 2002.

[2] E.M. Izhikevich, "Which model to use for cortical spiking neurons?", *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063-1070, 2004.

[3] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons", *Neural Information Processing Systems*, pp. 211-217, 1996.

[4] H. Shayani, P.J. Bentley, A.M. Tyrrell. "Hardware implementation of a bio-plausible neuron model for evolution and growth of spiking neural networks on FPGA", *NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 236-243, 2008.

[5] K. Cheung, S.R. Schultz, W. Luk, "A large-scale spiking neural network accelerator for FPGA systems", *International Conference on Artificial Neural Networks and Machine Learning*, pp. 113-120, 2012.

[6] Y. Zhang, *et al*. "Biophysically accurate floating point neuroprocessors for reconfigurable logic", *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 599-608, 2013.

[7] M. Beuler, *et al*., "Real-time simulations of synchronization in a conductance-based neuronal network with a digital FPGA hardware-core", *International Conference on Artificial Neural Networks and Machine Learning*, pp. 97-104, 2012.

[8] M. van Eijk, *et al*., "ESL design of customizable real-time neuron networks", *IEEE International Biomedical Circuits and Systems Conference*, pp. 671-674, 2014.

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

[10] J.R. de Gruijl, *et al*., "Climbing fiber burst size and olivary subthreshold oscillations in a network setting", *PLoS Computation Biology*, vol. 8, no. 12, pp. 1-10, 2012.

[11] Xilinx, http://www.xilinx.com/