FPGA Implementation of the Huber-Braun Neuron Model

Marcel Beuler[1], Alexander Krum[2], Werner Bonath[3] and Hartmut Hillmer[1]
[1]*Department of Electrical Engineering and Computer Science, University of Kassel, Wilhelmshöher Allee 71-73, D-34121 Kassel, Germany*
[2]*School of Computer Science and Engineering, University of New South Wales, NSW 2052, Sydney, Australia*
[3]*Department of Electrical Engineering and Information Technology, University of Applied Sciences, Wiesenstr. 14, D-35390 Giessen, Germany*

Keywords: Huber-Braun, Neuronal Network, FPGA, VHDL.

Abstract: The Hodgkin-Huxley model (HH) describes the initiation and propagation of action potentials in neurons closed to the biological conditions, but it is not well suited for large scale simulation of neuronal networks. In this paper, an implementation of the Huber-Braun model is presented. It is a simplified HH-type model and able to reproduce a wide variety of spiking patterns. An FPGA is selected as a reconfigurable hardware implementation platform to simulate the network functionality of the neurons. The 32-bit floating-point format and computation techniques (i.e. CORDIC) instead of LUTs are used to avoid loss of physiological information. We validated our design with a C++ program and report the synthesis result based on Xilinx Virtex 6 FPGA.

1 INTRODUCTION

In physiological research computer simulations play an important role in analyzing the neuronal information processing in the central nervous system. Basic elements are mathematical descriptions of neurons and synapses with different complexity. Hodgkin-Huxley-type neuron models generate action potentials by voltage-gated ion channels, therefore, they closely follow the biological concept. The original approach has four nonlinear differential equations and postulates three gating variables to model the dynamics of a sodium and potassium channel (Hodgkin and Huxley, 1952). Its solution requires a great deal of computing power and is thus not well suited for larger neuronal networks. That's why several simplified models with reduced computing effort have been developed in the past, and their use depends on each specific problem (Izhikevich, 2004).

For example, the Hindmarsh-Rose model has three nonlinear differential equations and can be used to investigate the spiking-bursting behavior of two electrically coupled neurons (Xia and Qi-Shao, 2005). The FitzHugh-Nagumo model approximates the HH system by only two differential equations (FitzHugh, 1961), but it doesn't generate bursts, which can be important for neuronal transmission (Izhikevich et al., 2003). One of the simplest models of a neuron's electrical properties is the Integrate-and-Fire model. It is based on a threshold approach instead of generating action potentials and allows large network simulations, but the model is unrealistic from a physiological point of view (Gerstner et al., 2014). In this paper, we have chosen the Huber-Braun neuron model which is also a simplified HH-type model. With tonic, bursting, and chaotic spike patterns it can reproduce a large amount of neuronal activity. In contrast to the Hindmarsh-Rose or FitzHugh-Nagumo model, each parameter has a clearly defined biological correlate.

In general, analog and digital approaches can be used to implement neuron models. The latter are more popular as they have the advantage of higher accuracy, lower noise sensitivity, better testability, higher flexibility, etc. (Muthuramalingam et al., 2008). In this context, an analog circuit of a Huber-Braun neuron has problems with the first period-doubling bifurcation (Hermida et al., 2012).

Target hardware for digital implementations can be conventional CPUs, GPUs, ASICs, and FPGAs. Conventional CPU-based execution is much slower in comparison to specialized hardware because of its serial nature. GPUs can exploit the parallelism of neuronal networks better and can be a powerful alternative to general purpose processors. However, they may not be able to meet real-time requirements due to high rates of data exchange between the neurons (Du Nguyen, 2013). ASICs provide the best performance regarding chip area and clock frequency, but they are expensive and their functionality cannot be changed after chip manufacturing. Although FPGAs are not

Beuler M., Krum A., Bonath W. and Hillmer H.
FPGA Implementation of the Huber-Braun Neuron Model.
DOI: 10.5220/0006499102470254
In *Proceedings of the 9th International Joint Conference on Computational Intelligence (IJCCI 2017)*, pages 247-254
ISBN: 978-989-758-274-5

as powerful as ASICs, they have the great benefit of reconfigurability during prototyping.

Several FPGA implementations of the HH neuron model have been implemented so far (Graas et al., 2004); (Zhang et al., 2009); (Bonabi et al., 2014). In this paper we present an FPGA-based processor architecture for the Huber-Braun neuron model. It is programmed in VHDL and can calculate up to 1600 neurons including electrical coupling in real-time on the Xilinx ML605 development board at 200 MHz clock frequency. To validate the design, two coupled neurons as well as a 20x20 network are tuned to different synchronization states, and these results are compared with a C++ simulation.

2 NEURON MODEL

The four-dimensional HH equations with transfer rates can be converted in a simplified HH model without the need for a power function and, finally, in a two-dimensional system as shown in (Postnova et al., 2012). These simplifications are acceptable, because the exact form of an action potential is less important than the modulation of the fire rate and impulse patterns. To consider subthreshold membrane potential oscillations which operate independently of the spike generation, the two-dimensional system is extended by two additional ion channels.

Within the model, spike generation is done by an HH-type component with a fast depolarizing sodium ion current (index d) and a fast repolarizing potassium ion current (index r) with voltage- and time-dependent conductances g_d and g_r. A slow depolarizing sodium ion current (index sd) and a slow repolarizing potassium ion current (index sr) with voltage- and time-dependent conductances g_{sd} and g_{sr} are responsible for subthreshold oscillations. Finally, a leakage current (index l) already known from Hodgkin-Huxley is included. Figure 1 (A) shows the equivalent circuit diagram of the neuron model with all ion currents as well as two current sources for an additional noise level J_{noise} and the synaptic process J_{ext}. The latter one consists of a current injected externally in the cell (J_{inj}) and a gap-junction current (J_{gap}) which is explained further on.

A positive hyperpolarizing current $J_{ext} > 0$ decreases the fire rate, since the membrane potential gets more negative. With the current-voltage-relation of the membrane capacity and Kirchhoff's Law we get Eq. (1). Leakage current and non-linear currents are modeled as Eq. (4) and Eq. (5), with g_i the conductances, a_i the voltage- and time-dependent activation variables, and V_i the related Nernst potentials.

$$C_m \frac{dV}{dt} = \underbrace{-\sum J_m - J_{ext}}_{f(V)} + J_{noise} \tag{1}$$

$$\sum J_m = J_l + J_d + J_r + J_{sd} + J_{sr} \tag{2}$$

$$J_{ext} = J_{inj} + J_{gap} \tag{3}$$

$$J_l = g_l (V - V_l) \tag{4}$$

$$J_i = g_i a_i (V - V_i) \qquad i = \{d, r, sd, sr\} \tag{5}$$

Similar to the HH gating mechanism, the activation variables of the ion channels are described by differential equations first order. Steady-states are modeled by a sigmoid curve and time constants replace the voltage-dependent delay functions. The fast sodium channel J_d is instantaneous active, so $a_d = a_{d\infty}$ can be estimated. Furthermore, J_{sd} is directly coupled with a_{sr} in Eq. (9) and the related steady-state activation $a_{sr\infty}$ is omitted. Both factors ρ and φ are responsible for temperature scaling of conductances g_i and time constants τ_i with T_0 the reference temperature in °C.

$$a_{i\infty} = \frac{1}{1 + \exp[-s_i (V - V_{0i})]} \quad i = \{d, r, sd\} \tag{6}$$

$$a_d = a_{d\infty} \tag{7}$$

$$\frac{da_i}{dt} = \frac{a_{i\infty} - a_i}{\tau_i} \qquad i = \{r, sd\} \tag{8}$$

$$\frac{da_{sr}}{dt} = \frac{-\eta \cdot J_{sd} - k \cdot a_{sr}}{\tau_{sr}} \tag{9}$$

$$g_i = g_{i0}\,\rho = g_{i0} \cdot 1.3^{(T - T_0)/10^\circ\mathrm{C}} \tag{10}$$

$$\frac{1}{\tau_i} = \frac{\varphi}{\tau_{i0}} = \frac{3.0^{(T - T_0)/10^\circ\mathrm{C}}}{\tau_{i0}} \tag{11}$$

Network simulations are performed by bidirectional gap-junction coupling via J_{gap} with g_{gap} the coupling strength, $V_{i,j}$ the membrane potential of an individual neuron at the position (i, j) in Figure 1 (B), and $V_{i+n,j+m}$ the membrane potentials of neighboring neurons. The summation is taken over all pairs (m, n) with $m, n \in \{-1, 0, 1\}$ (Postnova et al., 2009):

$$J_{gap_{i,j}} = \sum_{m,n} g_{gap} (V_{i,j} - V_{i+n,j+m}) \tag{12}$$

Normally, border neurons are coupled with neurons from the opposite border to get a torus-like network:

$$i + n = -1 \rightarrow i + n = 39 \quad j + m = -1 \rightarrow j + m = 39$$
$$i + n = 40 \rightarrow i + n = 0 \quad j + m = 40 \rightarrow j + m = 0$$

All differential equations are solved numerically by using the Euler integration method with $\Delta t = 0.1\,\mathrm{ms}$:

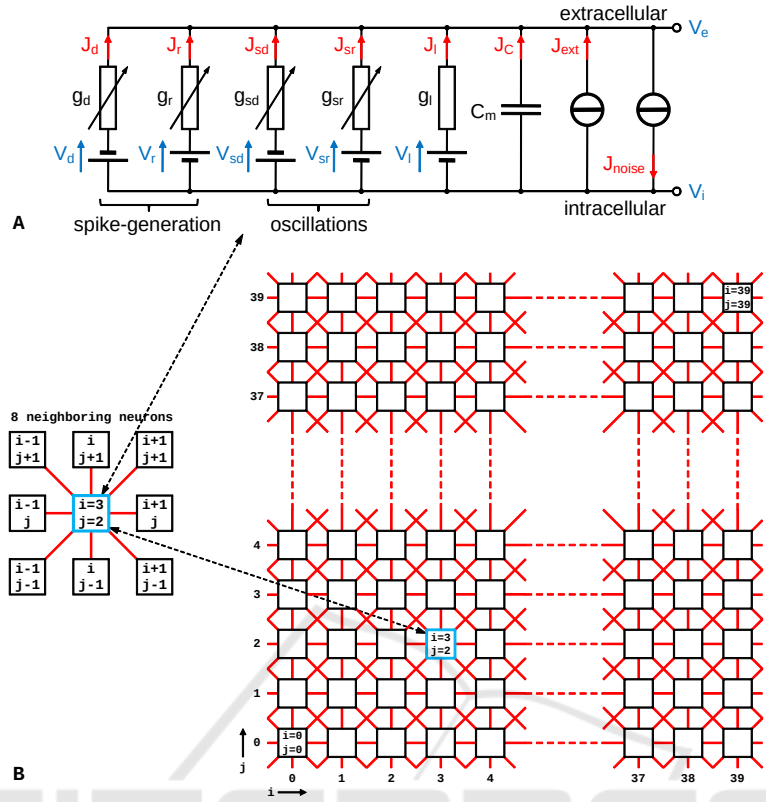

Figure 1: Simulated neuronal network. A) Equivalent circuit diagram of a single cell. B) Gap-junction coupling in a 40×40 network.

$$V^{(t+\Delta t)} = V^{(t)} + \frac{\Delta t}{C_m} f(V^{(t)}) + g_W \qquad (13)$$

$$a_i^{(t+\Delta t)} = a_i^{(t)} + \varphi \frac{\Delta t}{\tau_i} (a_{i\infty}^{(t)} - a_i^{(t)}) \qquad (14)$$

In Eq. (13), g_W is a white Gaussian noise process given by the Box-Mueller transform (Fox et al., 1988), where a and b are two uniformly distributed random numbers in the interval $[0,1]$. The noise intensity can be adjusted with the parameter d.

$$g_W = \sqrt{-4\,d\,\Delta t \ln(a)} \cdot \cos(2\pi b) \qquad (15)$$

Analyzing the distribution of phase differences between spike times enables us to distinguish between different states of synchronization in a system of especially two coupled neurons (Postnova et al., 2007). A spike time is taken when the membrane potential exceeds a $-35\,\mathrm{mV}$ threshold. The phase difference is calculated by Eq. (16), where t_1 and t_2 are the times of subsequent spikes of the first neuron and τ is the spike time of the second neuron (Pikovsky et al., 2001):

$$\Delta\varphi = 2\pi \frac{\tau - t_1}{t_2 - t_1} \qquad t_1 < \tau \leq t_2 \qquad (16)$$

Typical values of the parameter in the above equations can be taken from (Postnova et al., 2009).

3 PROCESSOR ARCHITECTURE

Figure 2 shows the basic design of the implemented architecture with two separated modules for the Huber-Braun model equations and the electrical synapses. It contains all necessary hardware to compute 1600 neurons including gap-junction coupling in real-time at 200 MHz clock rate. Operations of the Huber-Braun model are allocated to 9 pipelined arithmetic units (AUs), where "ADD_SUB" is a combined 12-stage adder/subtractor and "MULT" an 11-stage multiplier. All the functions (ln, cos, sqrt, and exp) and, in contrast to (Bonabi et al., 2014; Zhang et al., 2009), the divisions are calculated by the iterative CORDIC algorithm (Walther, 1971). The remaining arithmetic is designed to ensure an optimum utilization of the 50-stage CORDIC pipeline. To solve the limited convergence domain problem of the algorithm we use a unified division-free argument reduction method proposed in (Hahn et al., 1994).

Each AU processes 32-bit floating-point values to avoid undefined effects due to significant loss of physiological information (Zhang et al., 2009). It has an own program memory for individually pro-

gramming of the operations, up to four FIFOs to store the computed results and a ROM (a RAM used as ROM after initialization) to access model-specific parameters like g_l and V_{0r}. Two adders/subtractors also have a RAM for the temporary storage of model variables. These variables are the membrane potential $V^{(t)}$ (RAM of ADD_SUB_A) and three activation values $a_i^{(t)}$ (RAM of ADD_SUB_B), see Figure 2. OPA and OPB represent both operands for each AU selectable by the machine instructions. To ensure an efficient programming, the RAM content of ADD_SUB_A is provided simultaneously to a second ADD_SUB module. FIFOs can store calculated results in cases where a subsequent operation on the same or another AU cannot be started immediately. First of all, each stored FIFO result is available as an operand for the own AU as well as all the others. After the arithmetic operations are allocated to the individual AUs, this wiring complexity is reduced to the required connections in order to meet the timing specification. "PRNG" generates pseudo random numbers via linear feedback shift registers. Here, a program memory determines how many random numbers are generated per time interval.

At 200 MHz clock rate, the step size of 0.1 ms is divided into 20 identical subneuron cycles with 1000 clocks in each case. Furthermore, all subneuron cycles are divided into 2 blocks with 500 clocks as the basic execution time, i.e. all program memories with a size of $2^9 = 512$ cells can store up to 500 executable instructions. In such a block the operations of 40 neurons are allocated to the existing hardware components. During a neuron cycle each program memory is run through forty times with actualized index-registers to access variables, so that $40 \cdot 2 \cdot 20 = 1600$ neurons per core are computable in real-time.

Before a simulation can start, a host software has to generate the machine instructions for each AU as well as the ROM contents and initial states for all variables by means of the user settings. This enables us to test different neuron models without changing the FPGA configuration. All generated data will then be transferred to the processor during the initialization sequence.

Electrical Synapses. A neuron at the position (i, j) receives 8 partial gap-junction currents from neighboring neurons with $J_{gap_{i,j}}$ the total current. In contrast to Eq. (12), not each potential difference is weighted by the coupling strength g_{gap}, but only its summation. During a block calculation of 40 neurons all synaptic operations are allocated to two ADD_SUB modules and a multiplier. Because of direct further processing, both ADD_SUB modules store their results in registers. The synaptic process is independent of the neuron model and doesn't change, so the related program memories have a fixed content and cannot be modified after the FPGA is configured. Membrane potentials are stored in two RAMs having the same content as the ADD_SUB_A-RAM of the model equations when the initialization sequence has finished. In each new neuron cycle, both RAMs operate alternately by a toggle-bit, that means while one RAM provides the membrane potentials for the gap-junctions, the other stores the updated potentials of the current neuron cycle parallel to ADD_SUB_A-RAM of the model equations and vice versa.

The module SET_ADDR determines the RAM addresses of the actual neuron (Addr0) as well as all neighboring neurons (Addr1 to Addr8) which are identical to the neuron identification numbers (NIDs, $0 \cdots 1599$). These addresses are then used in the ADD_SUB_A module to select the membrane potentials $V_{i,j}$ and $V_{i+n,j+m}$ with $m,n \in \{-1,0,1\}$. $V_{i,j}$ is stored in the register RegV as operand A during a separate load instruction and the other potentials represent operand B, see Figure 2. Due to the additional load instruction, both RAMs can be implemented as single-port memories. The size of the neuronal network can easily be adjusted by the user settings from 1×1 to 40×40. Eight flags define whether the positions of the neighboring neurons are located inside the network borders and they've to be taken into account or not. All gap-junction currents are stored in FIFO FGAP. Finally, these values can be accessed as operands by the ADD_SUB_D module of the model equations.

Main-Controller. The processor has a main-controller as higher level control which analyzes incoming initialization data from the host and relays them to the related memories and registers. The packet-based data transfer with the host is realized by a USB 3.0 connection. A FIFO-Management-Unit (FMU) with own buffers in both communication directions controls all packet transfers between main-controller and peripheral USB controller. Processor-generated packets always have a length of 512 bytes and are called PPackets to distinguish them from USB packets (UPackets) which can have a length of 512 or 1024 bytes depending on a high-speed and super-speed connection, respectively.

Because the processor is a 32-bit system, a 4-byte word is processed per clock cycle. A PPacket always consists of 4 header and 124 data words. The header contains information such as "Packet Type" and "Packet Number". For example, the field "Packet Type" indicates if a certain program memory has to be initialized or the simulation has finished. User settings for core registers are summarized in a SETUP

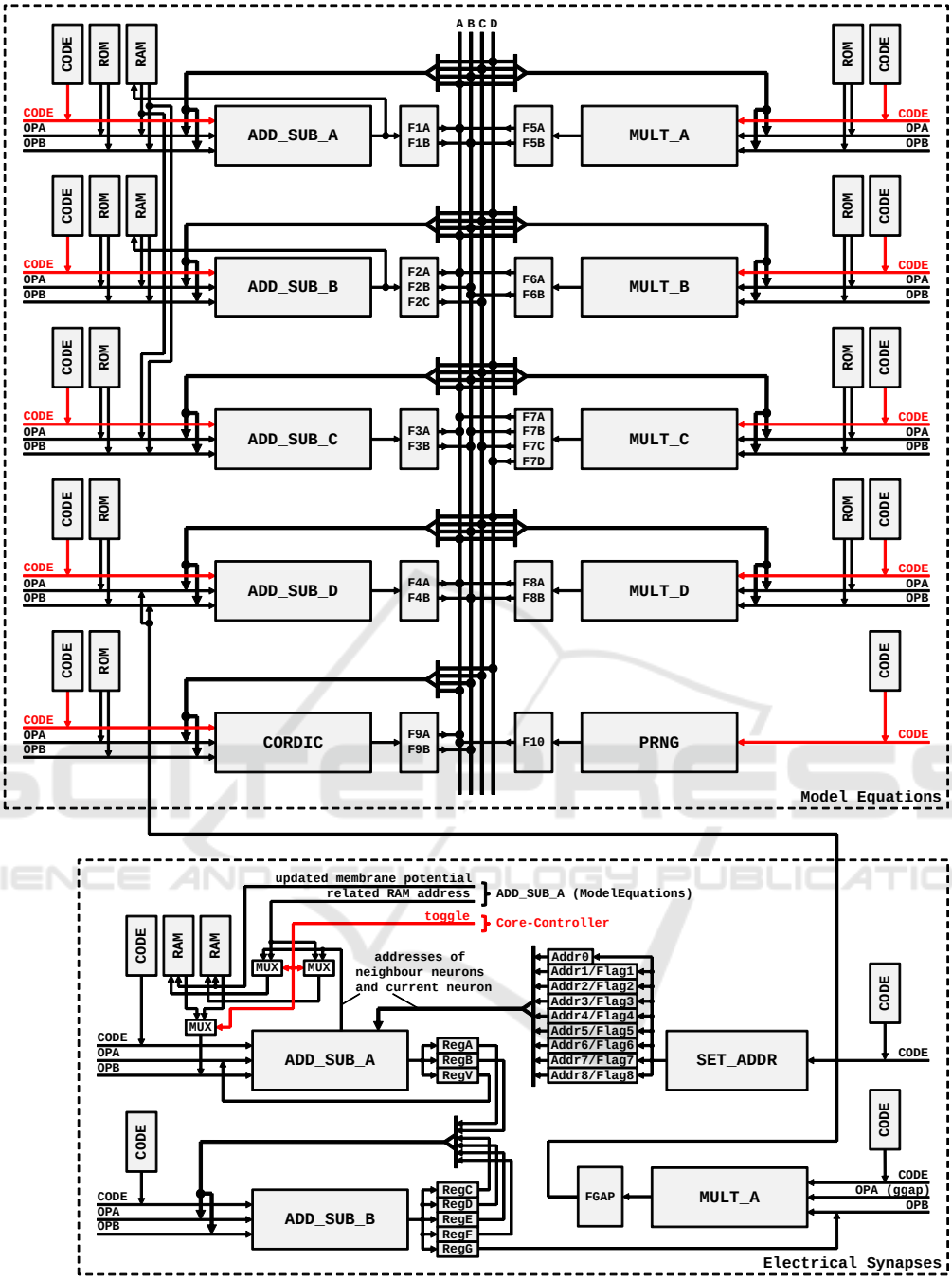

Figure 2: Block diagram of the main arithmetic components with two modules for the actual equations and the electrical synapses. In contrast to the synapses, all program memories for the Huber-Braun equations are programmable by means of user settings. A main-controller as higher level control and the core-controller to generate data packets for the host are not shown.

packet. It mainly includes the mode, the selected neuron model, the gap-junction networking (torus / no torus), the network size, the threshold for spike detection, as well as initial values for the ramp function g_{gap}, T, and J_{inj}. During run-time, each ramp function enables an automatic incrementation of its parameter until maximum is reached. The parameter interval is divided into 1000 steps, this means $\Delta P = (P_{max} - P_{init})/1000$ with $P \in \{g_{gap}, T, J_{inj}\}$ and $P_{max} \geq P_{init}$. All ramp functions determine important tuning parameters for network analysis.

Core-Controller. During run-time, the core-

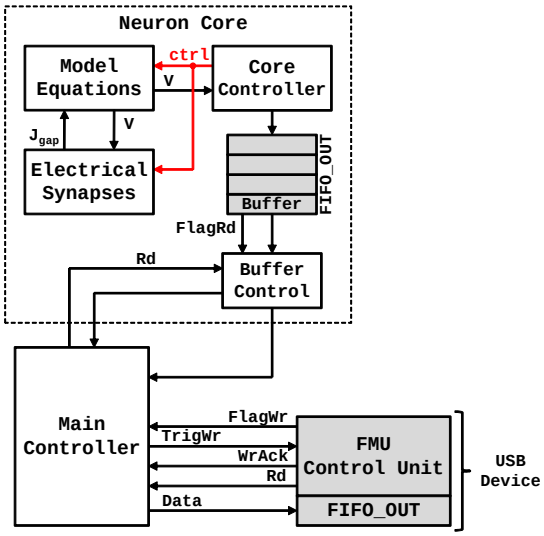

Figure 3: Block diagram of the processor architecture. All arithmetic components and the core-controller are summarized to a neuron core.

controller increments the program counters of all AUs, updates the ramp functions, alternates the RAM access for gap-junctions via a toggle-bit, and composes all PPackets to be transferred to the host in a small buffer which can be read by the main-controller, see Figure 3. Here, both modules for the actual equations and the electrical synapses as well as the core-controller are summarized to a neuron core. Such a core includes all necessary hardware for 1600 neurons and supports several modes defining how the calculated results are further processed in the core-controller. In this paper, we mostly use an event-driven mode called Preselected-Neuron-Spike-Time (PNST). It contains the spike time and the related NIDs of up to 124 individually selectable neurons. With these spike times the host software can determine interspike intervals (ISI) and phase differences according to Eq. (16), since the necessary computing power is limited. The mode Preselected-Network-Integer (PNI) includes, among other things, the mean membrane potential of all interconnected neurons (mean field potential, MFP) as an indicator for synchronization. Finally, the fire state of all neurons at any time is composed in the mode Full-Network-Spike-Time (FNST). This mode is used to plot the dotted spike times of the entire network.

4 RESULTS

The presented architecture is written in VHDL and synthesized on the Xilinx ML605 development board. Buffer registers of memory outputs can be absorbed by the related RAM during the synthesis process with high logic delay of approximately 2 ns. In cases where the timing failed, this absorbtion can be avoided by the keep attribute in the VHDL code. Together with a minimum reset path we are able to meet the timing constraints with a minimum period of 4.996 ns. The synthesis results in Tab. 1 show that the processor with one implemented neuron core takes 16 % of all LUT resources including the FMU. Therefore, the used FPGA has enough resources for further development like chemical synapses and multi-core operation.

Due to the block-wise calculation of always 40 neurons, a neuron core is optimized for network sizes of 40x40 in real-time and 20x20 in quadruple speed. However, the simulation of two coupled neurons is not efficient and should be done software-based, since the results of the remaining neurons are rejected and the high parallelism is left unexploited. For such cases we have expanded our hardware to include a speedup option which only works in PNST mode and defines the number of calculated blocks per neuron cycle. This is still worse than a software-based solution, but it helps us in the evaluation period to reduce the real simulation time for interspike intervals (ISI) from several hours to less than 10 minutes.

To validate our design, the FPGA results are compared with a C++ simulation. Figure 4 summarizes the results of two electrically coupled neurons as well as a 20x20 network at $T = 6\,^{\circ}\mathrm{C}$. All simulations use the coupling strength as tuning parameter from 0 to its maximum value. The two neuron system is well suited to study transitions between asynchronous and different types of synchronous states. Figure 4 (A) and (B) illustrate the interspike interval and the corresponding phase difference with $J_{inj} = 0.045\,\mu\mathrm{A/cm^2}$ and $g_{gap,max} = 0.03\,\mathrm{mS/cm^2}$ for the FPGA in PNST mode (left) and the C++ program (right). A comparison of these plots shows a good agreement between both implementations. The injection current is selected in a way that uncoupled neurons are closed to the first period-doubling. This is the region where the greatest variety of synchronous activity can be seen for coupled neurons (Postnova et al., 2007). At first, the neurons are in an out-of-phase synchronization state with a constant phase difference ($\Delta\varphi \neq 0, 2\pi$)

Table 1: Device utilization summary.

Slice Logic	Used	Available	Utilization
Slice Registers	19,676	301,440	6 %
Slice LUTs	24,253	150,720	16 %
Occupied Slices	8,720	37,680	23 %
RAMB36E1	58	416	13 %
RAMB18E1	42	832	5 %

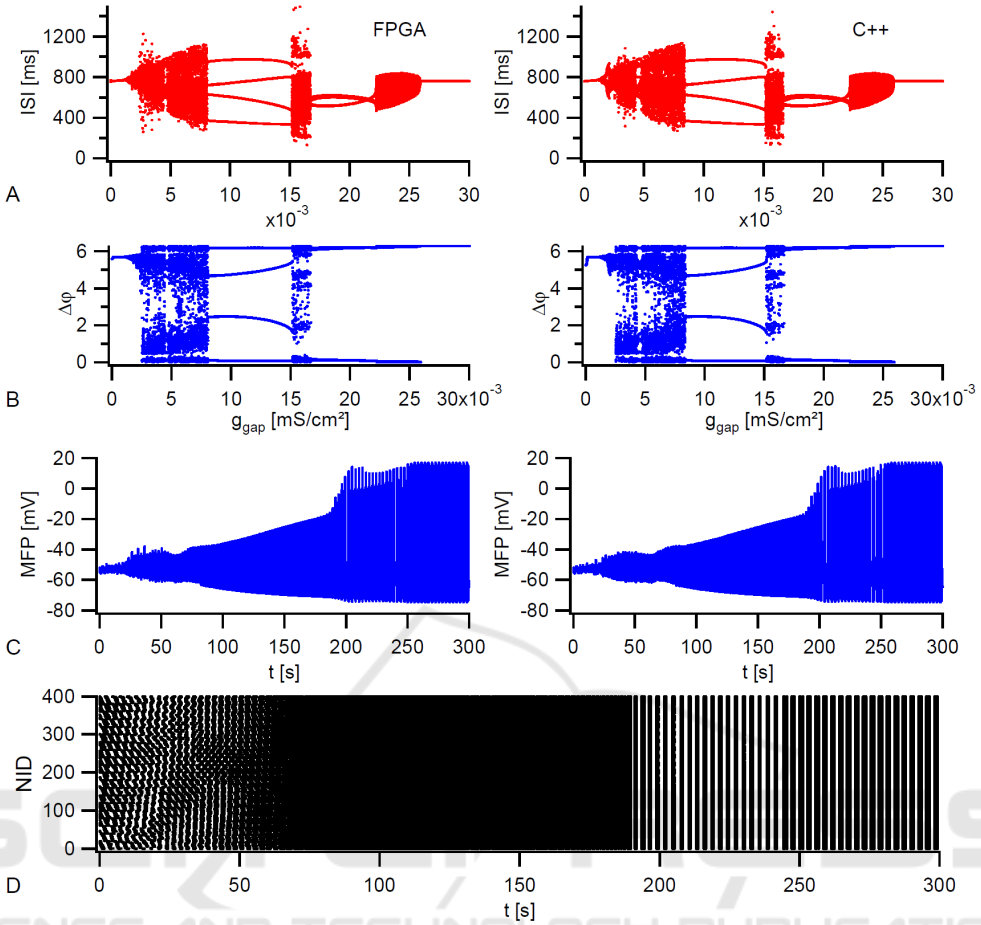

Figure 4: Simulation of two electrically coupled neurons as well as a 20×20 network at $T = 6\,°C$ and tuning g_{gap}. A-B) FPGA (left) and C++ (right) results for interspike intervals (ISI) and phase differences of two coupled neurons. Both simulations run 20 000 s with $J_{inj} = 0.045\,\mu A/cm^2$ and $g_{gap,max} = 0.03\,mS/cm^2$. C) FPGA (left) and C++ (right) result for the MFP of the network with $J_{inj} = 0.65\,\mu A/cm^2$ and $g_{gap,max} = 0.15\,mS/cm^2$. D) Dotted spike times of the FPGA network.

and a regular tonic activity. An increasing coupling strength leads to an abruptly asynchronous behavior with an irregular interspike interval distribution and then to an out-of-phase state of period 4. The latter means four lines in both diagrams. If the coupling strength in further increased, two additional regions with an asynchronous behavior can be seen until the neurons go into in-phase synchronization. In general, the simulated transitions are identical to (Postnova et al., 2007), where more information on physiological details can be found.

Figure 4 (C) shows, for the FPGA in PNI mode (left) and the C++ program (right), the mean field potential of a 20×20 network with $J_{inj} = 0.65\,\mu A/cm^2$ and $g_{gap,max} = 0.15\,mS/cm^2$. At this injection current, the uncoupled neurons are in the bursting regime. Here, an increasing coupling strength leads to a two step-like transition from the unsynchronized to the fully synchronized state with a MFP plateau at an almost synchronized state between 0.1 and $0.125\,mS/cm^2$. This transition was already published for a 10×10 network in (Postnova et al., 2009) and makes clear that the processor works accurately. At the end, the dotted spike times of the FPGA network taken in FNST mode with synchronized bursts at $t = 250\,s$ are illustrated in Figure 4 (D).

Finally, we have compared the real simulation time of our processor for the 20x20 network with an Intel Core i7 (6500U) with 3.0 GHz clock rate, 2 CPU-cores, and full utilization. The CPU-based simulation takes about 2.9 times longer.

5 CONCLUSIONS

We have presented a new processor architecture for the Huber-Braun neuron model, a simplified HH-type model. It is optimized for network sizes of 40x40

in real-time and 20x20 in quadruple speed. Synaptic connections are realized by bidirectional gap-junction coupling. The synthesized design runs on the Xilinx ML605 development board at 200 MHz clock frequency and takes 16 % of all LUT resources. A comparison with C++ simulations under physiological conditions shows that the processor works accurately. Furthermore, it is about 2.9 times faster than an Intel core i7 with 2 CPU-cores. Our hierarchical structure with a main-controller as higher level control and a neuron core with all arithmetic components including a core-controller allows the implementation of physiologically more relevant chemical synapses in a special module and can easily be extended to a multi-core architecture. This will drastically increase the computing power and is one of the next targets of this research.

REFERENCES

Bonabi, S., Asgharian, H., Safari, S., and Ahmadabadi, M. (2014). FPGA implementation of a biological neural network based on the Hodgkin-Huxley neuron model. *Frontiers in Neuroscience*, 8.

Du Nguyen, H. (2013). GPU-based simulation of brain neuron models. Master's thesis, Delft Technical University.

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1:445–466.

Fox, R., Gatland, I., Roy, R., and Vemuri, G. (1988). Fast, accurate algorithm for numerical simulation of exponentially correlated colored noise. *Physical Review A*, 38(11):5938–5940.

Gerstner, W., Kistler, W., Naud, R., and Paninski, L. (2014). *Neuronal dynamics - from single neurons to networks and models of cognition*. Cambridge University Press.

Graas, E., Brown, E., and Lee, R. (2004). An FPGA-based approach to high-speed simulation of conductance-based neuron models. *Neuroinformatics*, 2:417–435.

Hahn, H., Timmermann, D., Hosticka, B., and Rix, B. (1994). A unified and division-free CORDIC argument reduction method with unlimited convergence domain including inverse hyperbolic functions. *IEEE Transactions on Computers*, 43:1339–1344.

Hermida, R., Patrone, M., Pijuán, M., Monzón, P., Oreggioni, J., and Braun, H. (2012). An analog circuit implementation of a Huber-Braun cold receptor neuron model. *Conference of the IEEE Engineering in Medicine and Biology Society*, pages 3376–3379.

Hodgkin, A. and Huxley, A. (1952). A quantitative description of membrane current and its application to conductance and excitation in nerve. *Journal of Physiology*, 117:500–544.

Izhikevich, E. (2004). Which model to use for cortical spiking neurons. *IEEE Transactions of Neural Networks*, 15(5):1063–1070.

Izhikevich, E., Desai, N., Walcott, E., and Hoppensteadt, F. (2003). Bursts as a unit of neural information: selective communication via resonance. *Trends in Neurosciences*, 26(3):161–167.

Muthuramalingam, A., Himavathi, S., and Srinivasan, E. (2008). Neural network implementation using FPGA: issues and application. *International Journal of Electrical, Computer, Energetic, Elecronic and Communication Engineering*, 2(12):2802–2808.

Pikovsky, A., Rosenblum, M., and Kurths, J. (2001). *Synchronization - a universal concept in nonlinear sciences*. Cambridge University Press.

Postnova, S., Finke, C., Huber, M., Voigt, K., and Braun, H. (2012). Conductance-based models for the evaluation of brain functions, disorders, and drug effects. In Mosekilde, E., Sosnovtseva, O., and Rostami-Hodjegan, A., editors, *Biosimulation in biomedical research, health care and drug development*, chapter 5, pages 97–132. Springer-Verlag.

Postnova, S., Finke, C., Jin, W., Schneider, H., and Braun, H. (2009). A computational study of the interdependencies between neuronal impulse pattern, noise effects and synchronization. *Journal of Physiology*, 104:176–189.

Postnova, S., Voigt, K., and Braun, H. (2007). Neural synchronization at tonic-to-bursting transitions. *Journal of Biological Physics*, 33:129–143.

Walther, J. (1971). A unified algorithm for elementary functions. *Proc. Spring Joint Computer Conference*, pages 379–385.

Xia, S. and Qi-Shao, L. (2005). Firing patterns and complete synchronization of coupled Hindmarsh-Rose neurons. *Chinese Physics*, 14(1):77–85.

Zhang, Y., Nunez-Yanez, J., McGeehan, J., Regan, E., and Kelly, S. (2009). A biophysically accurate floating point somatic neuroprocessor. *Proceedings of the 2009 International Conference on Field-Programmable Logic and Applications (FPL)*, pages 26–31.