Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2007) 1186–1197
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Massively distributed digital implementation of an integrate-and-fire LEGION network for visual scene segmentation

Bernard Girau[a,*], Cesar Torres-Huitzil[b]

[a]LORIA INRIA-Lorraine / University Nancy 2, Cortex team campus scientifique B.P. 239 54506 Vandoeuvre-les-Nancy, France
[b]INAOEP, Computer Science Department, Luis Enrique Erro #1, Tonantzintla, Puebla, Mexico

Available online 12 December 2006

Abstract

Despite several previous studies, little progress has been made in building successful neural systems for image segmentation in digital hardware. Spiking neural networks offer an opportunity to develop models of visual perception without any complex structure based on multiple neural maps. Such models use elementary asynchronous computations that have motivated several implementations on analog devices, whereas digital implementations appear as quite unable to handle large spiking neural networks, for lack of density. Recent results show that this trend is now counterbalanced by FPGA technological improvements and new implementation schemes. In this work, we consider a model of integrate-and-fire neurons organized according to the standard LEGION architecture to segment gray-level images. Taking advantage of the local and distributed structure of the model, a massively distributed implementation on FPGA using pipelined serial computations is developed. Results show that digital and flexible solutions may efficiently handle large networks of spiking neurons.

Keywords: Integrate-and-fire spiking neurons; Local excitatory global inhibitory networks; Image segmentation; FPGA

1. Introduction

Computational neuroscience is a very active field of research. It aims at modelling, simulating and understanding mechanisms that underlie neural processes such as perception, action, learning, memory or cognition. Currently, one of the main lines of research of computational neuroscience studies the properties of the low-level mechanisms that are used by neural systems. In this approach, the computation mode is organized around neuronal events called spikes [6]. Biological arguments indicate that some vision tasks, as well as most olfactory tasks, cannot be satisfactorily performed by standard neural models that use an analog computation mode, where values stand for mean firing rates of natural spiking neurons [28,9,1,15]. The temporal coding that lies in spikes has to be fully exploited, for example through the notion of neuron synchronization [19,11].

On the other hand, the fundamental properties of neurons seen as processing units is a major source of inspiration to adapt processing architectures to algorithms and to embed neural processing in fine grain distributed systems. Artificial neural networks, considered as models of parallel computation, may represent an alternative to standard computation models for machine intelligence and intelligent behavior [4]. This claim is based on both theoretical and practical properties of neural networks. Their ability to deal with strong implementation and application constraints is a major current research domain to find ways to map neural connectivity and functionality onto hardware through architectures inspired by such massively parallel and hierarchical processing. The hardware plausibility is related to a very fine grain parallelism that fits parallel hardware devices, as well as to the emergence of very large digital reconfigurable systems, such as FPGAs (field programmable gate arrays) that become able to handle both adaptability and massive parallelism of neural networks.

These two major trends of computational neuroscience research (spiking models and embedded neural systems) are

*Corresponding author.
E-mail address: Bernaud.Girau@loria.fr (B. Girau).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.11.009

now mixed in several works. One of the projects carried out by our team is to develop a fully neural system for autonomous robotics. In this context we mainly develop two kinds of spiking models: advanced spiking models that model the olfactory bulb, and spiking neural networks that perform a rough real-time analysis of the robot visual environment.

Our spiking models for visual perception are based either on adapted standard spiking models or on principles inspired by neural field theory [20]. Simultaneously, we carry out several attempts to evaluate our preferred type of hardware device as a possible implementation basis for massively distributed spiking models. Following our works about adapted standard spiking models for visual perception, we have first implemented a well-known though criticized spiking model for image segmentation on FPGAs: the integrate-and-fire LEGION model ([21,22], as well as [29,3] for its adaptation to gray-level images). LEGION is criticized due to its rather simple biological plausibility and the mathematical/computational problems for hardware implementation feasibility, but the underlying 2D structure of this model fits the topological constraints of FPGA implementations.

Our work mostly focuses on low-area solutions, based on the use of a standard serial arithmetic within pipelined loops. As a result, a fully parallel implementation of the LEGION network has been mapped onto a Xilinx Virtex FPGA device, large enough to handle low-resolution robot image sequences. This work has confirmed the assertion of [13,8,7]: FPGA parallelism fits neural computation, so that FPGA implementations of neural networks may favor the use of neural approaches that keep fully distributed and localized processing. In our case, steady communication channels, synapses, have been preferred to event-driven systems. Section 2 briefly discusses related works: FPGA implementations of spiking models as well as event-driven implementation methods. We will justify our choice of an embedded reconfigurable system based on FPGAs as the hardware implementation option in Section 3, and we will describe its possible limitations as an implementation device for spiking models. The theoretical foundations of the LEGION model, its derivation using integrate-and-fire neurons, and its adaptation to the segmentation of gray-level images will be described in Section 4. The global architecture of the hardware implementation of the spiking neural model and its different parts will be detailed in Section 5, whereas Section 6 will provide the main features of our hardware mapping onto the Xilinx Virtex family.

2. Related works

Related works are recent and still very few. A single simple neuron is considered in [27]. In [17,24] implementations are still based on sequential processing and parallel arithmetic, so that their lack of scalability makes them unable to take advantage of the strong technological improvements that are a major aspect of FPGA solutions. Our work is closer to an almost simultaneous study that is described in [18]. Both works share common implementation principles: serial operators and massively distributed implementation of spiking neurons. Even results are very close. Yet both works are more complementary than competing. In [18], the authors develop an implementation method for a partially generic model of integrate-and-fire spiking neuron. Their work stands as a very interesting step towards an automated design of spiking models on FPGAs, but it does not take into account several aspects of the computation when real data are considered. Our work is more specific to a given model (integrate-and-fire LEGION), which results in a more compact but less generic implementation, that is able to perform all required computations for the considered application (visual segmentation), including the complex image preprocessing that is mandatory for such a model.

In this paper, as well as in [18], a time-discretized computation has been preferred to an event-driven approach. Event-driven simulations of spiking neural networks have been proposed in [30], and they have been extended in [16]. Instead of using fixed time steps to update neurons, the state of the whole network is computed at event (spikes) occurrence times. Such methods provide a better accuracy in the simulation of the continuous phenomenons that generate spikes. But they are not always easy to implement, and they require that the differential equations of the neuron state lead to an analytical solution when determining the time of the next event. Moreover, in terms of computational efficiency, these methods become inefficient when the average spike activity increases, which might be a problem for real-time applications. Another main drawback makes event-driven methods unsuitable for our work. Despite several attempts with parallel computers, their natural lack of parallelism lies in the process of selecting the next neuron that fires so as to know the next update time for the whole network, whereas our implementation principles are based on a fine-grain parallel mapping of the neurons with local interactions.

3. Choosing field programmable gate arrays

3.1. Neural computations on FPGAs

The very fine-grain parallelism of neural networks uses many information exchanges, thus it better fits hardware implementations than conventional processors. Configurable hardware devices such as FPGAs are cheap, flexible, and they offer a good compromise between the hardware efficiency of digital ASICs and the flexibility of a rather simple software-like handling [14,23]. State-of-the-art FPGAs offer high-performance, high-speed and high capacity programmable logic solutions that enhance design flexibility while reducing time-to-market. FPGAs, such as Xilinx FPGAs are based on a matrix of configurable logic blocks (CLBs). Each CLB is able to implement small combinational logical functions (four or

five input functions). A CLB provides a few elementary memory devices (flip-flops or latches) and some multiplexers to implement sequential logic. The Xilinx Virtex series are well- suited for the implementation of the serial arithmetic operators that we use in this work. The CLBs can be connected using a configurable routing structure. In Xilinx FPGAs, CLBs can be efficiently connected to neighboring CLBs as well as CLBs of the same row or column. The configurable communication structure can connect border CLBs to input/output blocks (IOBs) that drive the chip input/output pads. An FPGA-based approach simply adapts to the handled application, whereas a usual VLSI implementation requires costly rebuildings of the whole circuit when changing some characteristics. A design on FPGAs requires the description of several operating blocks. Then the control and the communication schemes are added to the description, and an automatic "compiling" tool maps the described circuit onto the chip. Therefore, configurable hardware appears as well- adapted to obtain efficient and flexible neural network implementations.

Several works show that FPGAs are a real opportunity for flexible hardware implementations of neural networks [8,7]. The main advantages of reconfigurable hardware for neural network implementations are: reprogrammable FPGAs allow prototyping, FPGAs may be used for embedded application, an on-chip learning may be followed by a reconfiguration of the FPGA with an optimized implementation of the learned network, FPGA-based implementations may be mapped onto new improved FPGAs (unlike the use of neuroprocessors, which rapidly become outdated), and FPGAs provide a rather high computational density with high performance application due to the customizable logic gates. However, the 2D-topology of FPGAs does not fit the connection complexity of most standard neural network models in a straightforward way. Moreover, FPGAs still implement designs with a relatively limited number of logic gates, whereas neural computations (multipliers, transfer functions) are area-consuming operators. The implementation of standard neural models raises specific problems to be solved by means of architectural innovations such as in [7].

3.2. Spiking models on FPGAs

Most works on neural network digital hardware implementations use a group of binary digits (words) to implement classical neural models where all interactions are represented by the mean firing rate of the neurons [12,2]. On the other hand, recent research in neuroscience has developed spiking neural models that are much closely coupled with action potential and spikes communication and even more tightly related to the brain abilities than classical neural models. Spiking models suggest that there is significant information encoded in the relative timing between pulses and this information is partially hidden by the inherent averaging in computing firing rates. Spiking neural models are different from classical connectionist models in the sense that information is transmitted by means of pulses or spikes through time rather than average firing rates. The spike-time interrelationship enriches the dynamics of spiking models that exploit the temporal domain to encode and retrieve information in the exchanged spikes.

As shown by recent works [17,24,18], the implementation of spiking neural networks on digital circuits has become an active line of research, whereas previous implementations were mostly taking advantage of the density of analog devices. This change is mainly linked to the flexibility and rapid technological improvements of FPGAs. Advanced implementation methods (such as the one we propose in this paper or the work of [18]) are now necessary to make digital implementations able to handle large networks of spiking neurons. This requires low-area implementations of this kind of neuron, based on the fact that spiking neural networks are well suited to be implemented in digital logic: they are inherently binary and the time of spike can be handled through digital circuits since communication among neurons is handled by spikes, so that single bits might be used for inter-neuron communication, instead of floating point words. That is, a single wire is sufficient for the connection between two neurons and the routing requirements of neural interconnection on a 2D chip can be simplified. Besides, spiking neurons are weighted integrators that can fire and be reset as soon as a threshold value is reached, which avoids area greedy operators in the hardware implementation. Combining spiking neural models with specific hardware implementations on FPGA architectures opens new possibilities to build intelligent autonomous systems.

4. LEGION model description

The choice of the integrate-and-fire LEGION model follows several preliminary works in our team to study how spiking models may be used for visual perception. Moreover, the 2D architecture of this model makes it particularly well-fitted for a straightforward parallel implementation on FPGA.

4.1. Principle

Oscillations of neurons in the cortex are now considered as an important mechanism that is involved in several cognitive functions of the brain. Our research team is more particularly interested in the role that this mechanism plays in perceptual tasks such as vision and olfaction [26,25,1,10,9].

Several studies have shown that oscillations of neurons take part in several perceptual functions of the cortex. These oscillations are used in visual perception to segment features in a visual scene [10,21,3]. Neurons in cortical areas such as primary visual area V1 are known to "see" only a small localized part of the visual field (such areas are

retinotopically organized). And yet synchronized behaviors have been detected for groups of neurons whose visual fields are strictly separated. These neurons bind together thanks to the synchronization of their firing activities. Such behaviors cannot be studied by means of standard neural network models that only take into account the mean firing rate of the neurons. This is why one of our research goals deals with neural elementary mechanisms.

The LEGION network (Local Excitatory Global Inhibitory Oscillator Network) has been proposed in [22] based on biological considerations. This simple model consists of a 2D grid of oscillators (see Fig. 1 where an oscillator is connected to its eight immediate neighbors) that receive visual stimuli. Neighboring oscillators are coupled by excitatory connections, and a global inhibitor gets active when any oscillator jumps up. The excitatory/inhibitory connections result in synchronizations (through excitation) of distinct (through inhibition) groups of oscillators so as to segment the input image. This model has been applied to the segmentation of binary and gray-level images [29].

The original LEGION model uses relaxation oscillators [22] as basic computational units. Each of them is modeled by two coupled units, an excitatory and an inhibitory one. The excitatory unit is connected to both input stimulus and neighboring oscillators, whereas the inhibitory unit interacts with the excitatory one to produce oscillations. Exact equations may be found in [22].

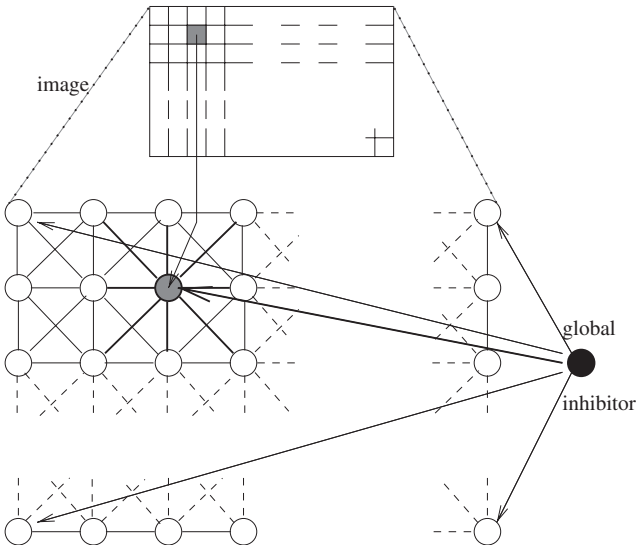

Fig. 1. 2D architecture of the LEGION network: a neural oscillator is connected to its eight immediate neighbors and to a global inhibitor. Each oscillator receives visual stimuli from pixels of the input image.

In order to segment images, the LEGION model groups oscillators that receive their input from similar features in an image. Oscillators group together by synchronization of their phase thanks to excitatory connections, and they get desynchronized from other groups of oscillators by means of global inhibition (Fig. 2 shows a similar phenomenon with the integrate-and-fire LEGION network described below: in this image taken by our robot camera, pixels in white correspond to a group of simultaneously firing neurons, thus segmenting a distinct part of the wall, resp. floor). When an oscillator jumps up, it excites its neighbors, which immediately jump up if they were about to. Excitation propagates and groups get formed. Global inhibition ensures that only one group can get active at a time.

4.2. Integrate-and-fire LEGION

After having been applied to binary image segmentation, and then to gray-level image segmentation [22,29], the LEGION architecture has been chosen in [3] as an interesting model to study synchronization and desynchronization of integrate-and-fire oscillators (or spiking neurons). Beside the interest of this approach so as to study the complex properties of assemblies of such oscillators, it provides us with a model that keeps synchronization properties while being suited for digital VLSI implementation.

There are four computational components of LEGION that describe its behavior: group participation, group formation, group segregation, and group suppression. Each computational component is implemented by the following network components: input stimulus, excitatory coupling, global inhibitor, and neuron potential, respectively. The dynamics of a LEGION network of integrate-and-fire neurons is defined according to the following equation:

$$\frac{dx_i}{dt} = -x_i + I_i + \sum_{j \in N(i)} \frac{\alpha_{ij}}{Z_i} P_j - G, \quad (1)$$

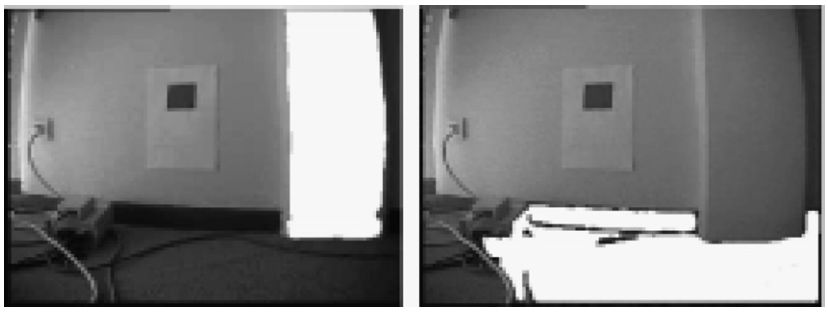

Fig. 2. Image segmentation by an integrate-and-fire LEGION network: simultaneously spiking neurons correspond to the white pixels.

where the sum is over the neurons in a neighborhood, $N(i)$, around neuron i, x_i is the potential of neuron i, Z_i is the number of nearest neighbors that neuron i has (to take into account boundaries problems). α_{ij} is the coupling strength. P_j is a neighbor neuron that fires at a given time producing an excitation pulse to neuron i. G is an instantaneous inhibitory pulse to the entire network when any neuron in the network fires. When $x_i = 1$ the neuron is said to fire; its potential is instantly reset to 0, and it sends excitation to its neighbors.

Parameter I_i is the external stimulus given to neuron i and when applied to image segmentation it depends on the input image. In order to segment gray level images, parameter I_i is computed as follows [29]. Let p_i be the intensity of pixel i. If $|p_i - p_j|$ is less than a given threshold, then the two pixels are said to satisfy the pixel difference test. Two neurons have a nonzero coupling strength only if they are neighbors and if their pixels satisfy the pixel difference test. The weights of the connection strengths α_{ij} are determined by the number n_i of neighboring pixels of i that pass the pixel difference test. $\alpha_{ij} = \alpha/n_i$ if neurons i and j satisfy the pixel difference test, where α is a constant. If neurons i and j do not satisfy the pixel difference test, α_{ij} is set to zero. If half of the pixels in $N(i)$ satisfy the pixel difference test, then I_i is set to a value I_L greater than 1 (leader value). If no neighboring pixel satisfies the pixel difference test then I_i is set to zero. Otherwise, I_i is given a stimulus I_N (near-threshold value), which is less than but near 1.

It has been demonstrated that integrate-and-fire LEGION maintains its segmentation capabilities since their synchronizing properties are similar to relaxation oscillators [3]. Due to the oscillatory characteristic in LEGION, a large amount of differential equations need to be solved, which induces a high computational load and power consuming task in conventional processors. The use of integrate-and-fire neurons are attractive for digital VLSI implementation for perceptual organization in embedded systems.

5. LEGION hardware implementation

5.1. General architecture

Due to the high computational requirements of LEGION, the highly dense interconnectivity and the area greedy operators for a fully parallel implementation, we have chosen a serial hardware implementation on massively fine grain parallel structures as a suitable architectural option for embedded connectionist processing for visual perception.

5.1.1. Arithmetic and precision

The fine grain pipelined internal structure of the proposed digital architecture was implemented using fixed point arithmetic. The pixels of the input image were coded in 8 bits for gray level images. Analysis and experimental results were carried out to determine a suitable wordlength in terms of performance and cost for the coding of the values involved in the computation of neuron potentials. Considering the various ranges of handled data (internal states, update values, etc.), a minimum precision of 10 bits (with two bits for the integer part) is required. We used time-discretized software simulations of the LEGION network using fixed-point computations to estimate the precision required to perform a segmentation similar to our event-driven software implementation using floating-point computations (same groups of neurons simultaneously firing for small images).

For this FPGA LEGION implementation, it appeared that a serial arithmetic with a 12-bit fixed point representation should be chosen in order to favor a high density of neurons per silicon area while preserving accuracy and performance of the model. A 2-complement representation with 2 bits for the integer part and 10 bits for the fractional part is used for the internal representation of neuron potentials. Larger wordlengths can result in a more accurate synchronization and convergence at the expense of a slightly different control and a slower implementation. It must be pointed out that our architecture may handle up to a 16-bit fixed point representation without any architectural change (only control signals must be updated). Larger precisions are also possible at the cost of a more complex handling and storage of the neuron potential and constant values.

5.1.2. Architecture of the network

Fig. 3 illustrates the general architecture of the implementation for a 4×4 LEGION network. It consists of a grid of 4×4 identical neuron modules, a global inhibitor module, and a ROM module to store several constants. All neurons use the same implementation, but at the border each neuron module adapts to the number of neighbors as required by Eq. (1).

Each neuron uses loop computations within an internal pipeline scheme. The control of these computations are synchronized in the whole network so that neurons may serially communicate their excitations to their neighbors. In order to simplify the block-diagram of Fig. 3, only few buses and wires are shown: the excitation computed by the neuron in position (2,2) are sent to all its eight neighbors and to the global inhibitor module, the input pixel signal received by neuron (2,2) is also received by its eight neighbors, the tests of difference computed (and sequentially stored during the pixel difference test phase) by neuron (2,2) are sent to the corresponding neighbors, and finally neuron (2,2) receives the global inhibition signal and some constants from the ROM module. All other neurons handle similar IOs not shown on Fig. 3. In the following subsections, the general hardware architecture for the LEGION neuron model and its main components will be described in some detail.

5.1.3. Architecture of a neuron

The general architecture for the LEGION hardware neuron model is shown in the simplified block diagram of

Fig. 4. The hardware neuron model has 1-bit inputs and outputs to exchange data among neurons and to the external world. Inputs are mainly used to receive image pixels and excitation spikes from neighboring neurons and outputs are used for outgoing spikes. All integrate-and-fire neurons used in the network are architecturally identical although they may be functionally different on the image boundaries since some inputs are nonexistent and no wrap-around is used. All the required logic for controlling the behavior of a neuron is built inside each neuron, though it is based on received external signals, such as reset and clock, for synchronization and data exchange. The dynamics of every integrate-and-fire neuron in the LEGION network is tuned by constant parameters, such as the α_{ij} and I_i parameters in Eq. 1. These parameters are received from global registers and memories through a lookup table approach, that avoids the hardware implementation of area greedy operators, multiplication and division.

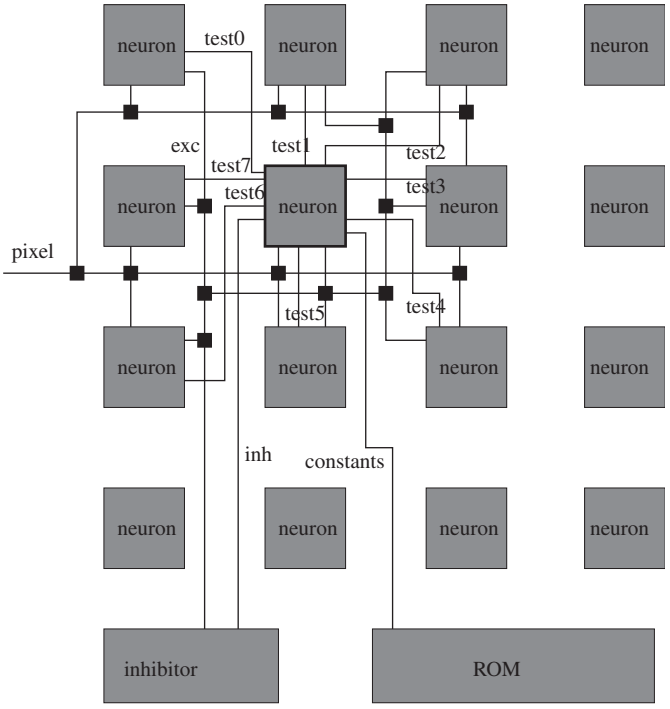

Fig. 3. Simplified architecture of a whole 4×4 integrate-and-fire LEGION network.

The proposed hardware neuron model is constituted by four main modules: a test of difference, an arithmetic, a RAM and a leader-excitation detector module. The key idea for the design of the proposed neuron architecture relies on the association of basic operators to specific arithmetical computations of Eq. (1), the use of some previously stored constants to avoid hardware complexity of nonlinear operators and in the transformation of the differential equation into an equation of differences using the Euler's method. The functionality of the main modules and their implementation details are described below.

Pixel difference test module: The difference test module determines the number of neighbors that fulfill the requirements of the difference test. It is essentially constituted by a multiplexer, a subtractor, a counter and a comparator optimized for serial arithmetic.

The difference test is applied between the pixels of two neighbor neurons, the central pixel and the current pixel selected by the multiplexer. The absolute difference of these values is compared to a constant threshold, 16 for the current implementation (any other power of 2 value would lead to a similar architecture). If the absolute difference is below the threshold, then pixels satisfy the test of difference. The main goal of applying the test of difference is to define groups of neurons with non-zero coupling connections. A neuron that satisfies the difference test with most of its neighbors is part of a homogeneous region and tends to be synchronized through time with neurons of this region. The data flow is controlled by signal

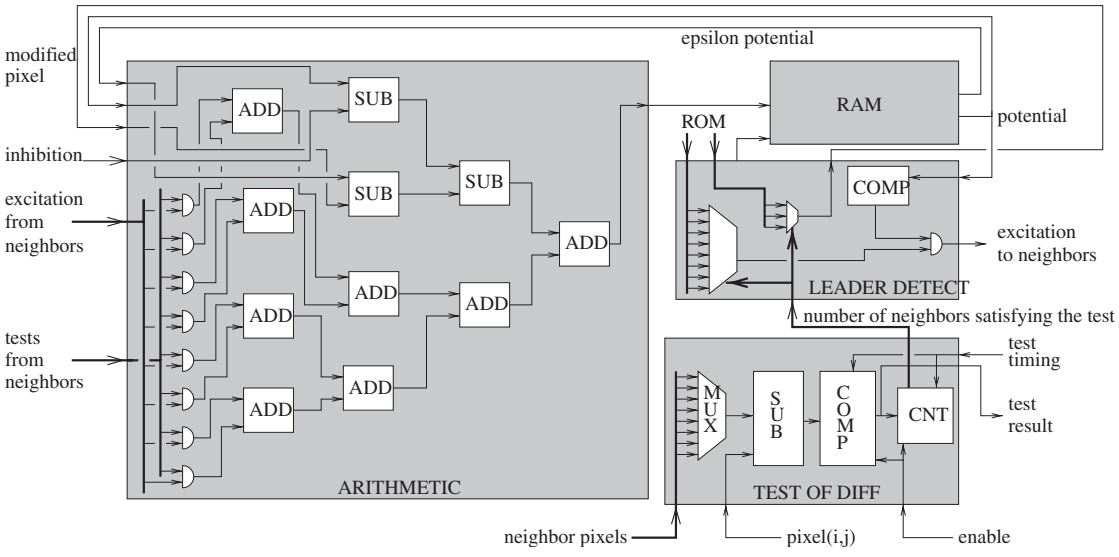

Fig. 4. Architecture of a single neuron.

`Enable` that starts the module functionality and by signal `Test timing` that provides the timing slot to perform the difference test on each neighbor pixel. The module operates sequentially with all its neighbors and then it sends the number of neighbors that satisfy the test to the leader detector module for further utilization.

Arithmetic module: This module performs the arithmetical operations that define the dynamics of the neuron model, as stated in Eq. 1. The module is composed of a tree of serial adders and subtractors, and AND gates. The arithmetic module receives excitation signals from its neighbors, the global inhibition, and the previous internal potential stored in the RAM module. An integration step $\varepsilon = 1/4$ is used to solve the difference equation through the Euler's method. To avoid multipliers, the potential and ε times the potential are simultaneously provided by the RAM module using an appropriated addressing scheme. A set of AND gates verifies if a neuron has produced a spike and if it satisfies the difference test, and it labels this neuron to compute its contribution to the potential. The arithmetic module outputs the current neural potential to the leader detector module.

RAM module: This module is basically constituted by a 16×1 dual port memory RAM that stores the neuron internal potential produced by the arithmetic module on each clock cycle. The two bus addresses are driven by two global counters, delayed four clock cycles one of each other, to provide the current neuron potential and the potential scaled with the integration step ε.

Leader detector and excitation module: The leader detector module determines the modified value of a pixel associated to a neuron, and generates the excitation potential contribution at a given time, and the excitation spike. Three possible values for the modified pixel may be assigned (see Section 4.2): $I_L = 1.25$, $I_N = 0.95$, and 0. Also, the excitation potential contribution depends on the number of neighbors satisfying the difference test. The module computes this strength, selecting values previously stored in an internal memory in order to avoid multipliers and divisions. The module generates an excitation pulse and a reset signal for the internal state of the neuron, produced by the comparator component, if the internal potential is greater than the unit.

5.1.4. Complementary modules

Fig. 5 shows a block diagram of the global inhibitor used in the LEGION network. The inhibitor receives the excitation signals from neurons as inputs. If at least one excitation is present, then the inhibitor outputs an inhibition signal which is applied to all neurons. As the detection of excitations must be done in a proper time, an enable signal stores the output of the OR gate into a latch. The latch output is sent to an AND gate which is enabled for the time required to serially transmit the fixed point value of the global inhibition value stored in a global memory. If no excitation is generated, then the inhibition is set to zero.

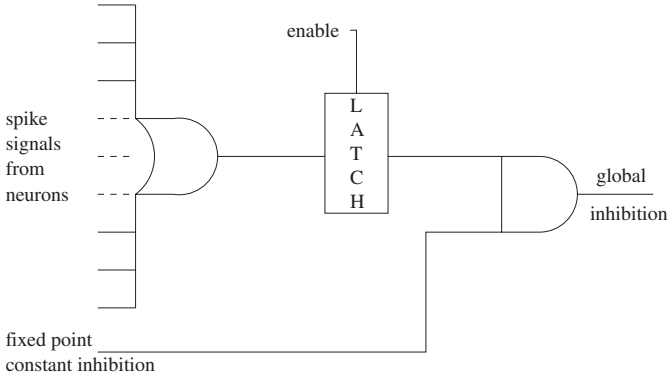

Fig. 5. Block diagram of the global inhibitor which produces a global inhibition for the neurons if any excitation/spike is generated.

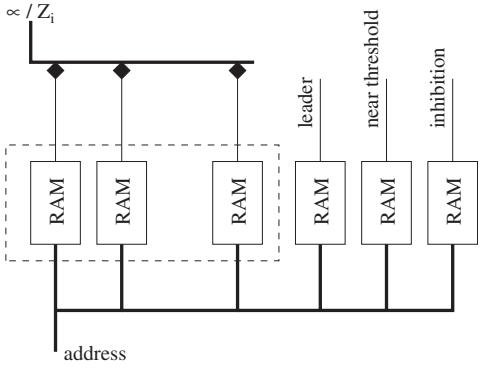

Fig. 6. Block diagram of the ROM module to store constant parameters of LEGION and the values of α times the reciprocal of the number of neighbors to avoid divisions.

Several LEGION parameters are stored in a global ROM memory module organized in several banks as shown in Fig. 6. RAM memories of 16×1 store the fixed point representation of parameters such as inhibition, leader and near-threshold values. A set of eight RAMs, enclosed by the broken line rectangle, are used to store the values of α times the reciprocal of the number of neighbors which are required to compute the excitation contribution to the neuron potential. The use of this approach allows reducing hardware complexity avoiding division in the neural oscillator model at the cost of storage and the use of global communication. All the RAMs use the same address bus which is generated by a counter in the global control module in LEGION (the constant values must be sent serially).

6. FPGA implementation results

The proposed hardware model for the neural oscillator and LEGION has been successfully modeled in very high speed integrated circuit hardware description language (VHDL) and implemented in a FPGA device. The VHDL model for the LEGION is configurable and it may be built up for different gray level image sizes.

6.1. Synthesis results

The synthesis results for the neuron hardware model and for different configurations of LEGION targeted to a Xilinx Virtex XC2V6000-4FF1517 device are summarized in Table 1. For each kind of FPGA resource and for each implemented model, the global usage ratio is given as a percentage. The design was synthesized, placed and routed automatically in Xilinx Foundation ISE 7.1i. It is important to point out that when the LEGION size increases then the maximum clock frequency is reduced considerably since more routing resources are required which limits the performance.

The hardware resource utilization summary highlights that the massively parallel and pipelined implementation is highly attractive for FPGA implementation. According to the post layout timing information in Table 1, a 26×26 LEGION can reach a maximum clock frequency of 65 MHz for segmentation times in the order of microseconds. This is a consequence of the use of pipelining registers that shorten the interconnect delay in the critical path. Due to the interconnect centric nature of LEGION, the hardware performance depends on the FPGA family chosen since routing is expensive in terms of interconnect delay. In this sense, a segmented routing FPGA architecture is beneficial [14,23]. As can be inferred from the presented results, the proposed digital approach for LEGION implementation is particularly efficient from the device utilization point of view.

6.2. Simulation environment

To validate the functionality of the proposed digital hardware architecture for LEGION, both functional and timing simulations were performed. The timing simulation data produced in the place and route phase was used for the timing simulation of the design. A set of testbenches were developed to validate each component in the neuron model and the LEGION network. A simulated hardware environment for the complete LEGION network model was used. This environment is shown in Fig. 7. An input image file used as stimulus is read on simulation time and data are loaded into a RAM VHDL model of an input memory bank. Then, pixels are sent by a data distributor to a distributed multiport RAM model to provide parallel access to all pixels of the input image required by LEGION network. After processing, excitation spikes are collected by the data collector module and stored in a distributed multiport RAM and then sent to an output memory bank for further analysis.

This simulation environment was developed keeping in mind the common hardware platform organizations that are used for physical implementations where the two main components are an FPGA device and external memory banks for data storage. For on chip testing, the major modification of the simulation environment is the acquisition and transfer of real image data from a camera to the memory banks. In this sense, some modules should be developed according to hardware platform constraints to acquire images from an attached camera. Data transferring to the input memory bank and result transferring to a visualization module or to a host should be developed as well. These aspects are currently being addressed for an experimental setup based on a vision-oriented Celoxica RC340 board (using a Virtex XC4VLX160) we are about

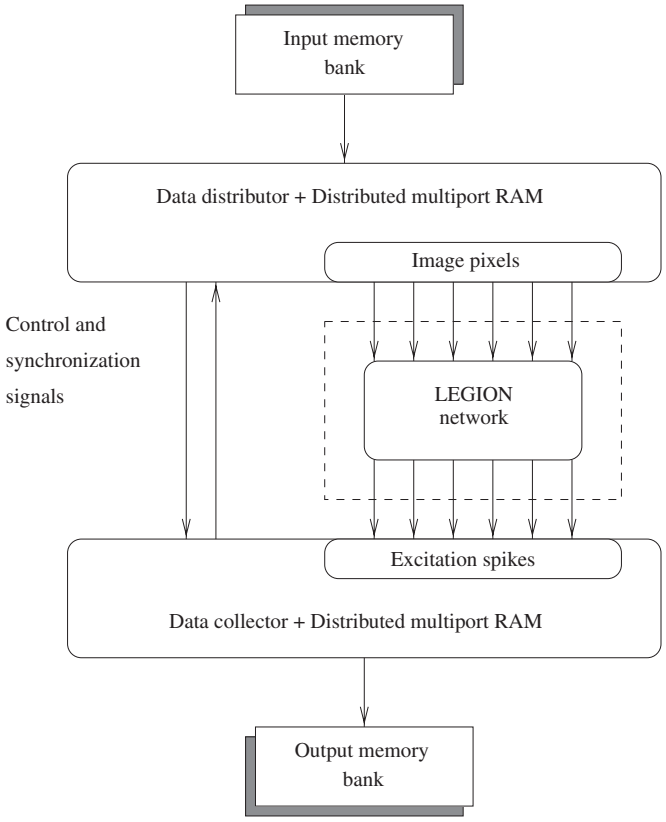

Fig. 7. Generic view of the simulation/hardware environment for LEGION network.

Table 1
Virtex post-place and route synthesis results for a VIRTEX-II device implementation (XC2V6000-4FF1517)

Parameter	One neural oscillator	8×8 LEGION	16×16 LEGION	26×26 LEGION
Number of slices	44/33792 (1%)	2615/33792 (7%)	10700/33792 (31%)	25760/33792 (76%)
Number of 4-input LUTs	67/67584 (1%)	4163/67584 (6%)	17019/67584 (25%)	46710/67584 (69%)
Number of flip-flops	22/67584 (1%)	1361/675840 (2%)	5617/67584 (8%)	15549/67584 (23%)
Maximum clock frequency	112.38 MHz	100.26 MHz	77.22 MHz	64.45 MHz

to purchase. Additional results and considerations will be available with this board.

6.3. Experimental results

The produced results were compared to the software implementation of LEGION providing satisfactory results in the tests performed on the hardware model. The values used for LEGION parameters in this experiment are: 0.2, 0.25, and 0.01 for α, the integration step and inhibition, respectively (see Eq. (1)). Following Section 5.1.1, we recall here that this hardware implementation exactly computes the same results as the fixed-point simulation software using the same precision, but these results may be different from the floating-point version.

Fig. 10 shows a small 3×5 input image used to exemplify the LEGION architecture operation (top-left) and the pixel-neuron number convention as well (top-right). Fig. 8 shows the difference test timing for two neurons associated to two pixels in the input image, P7 and P8. At the end of the difference test, indicated by a low value of signal Enable as shown in the figure, the number of neighbors satisfying the test for P7 and P8 are 4 and 3, respectively. Fig. 9 shows a snapshot of the time evolution of the oscillatory and synchronizing activity in the network. For explanation purposes and space considerations, only the excitation signals from neurons, the global inhibition signal and the main clock are shown for a short interval of time. The number associated to the excitations corresponds to the same pixel number in the input image as shown in Fig. 10. The clock signal appears continuous since the scale of time is too large to show the transitions details. The neuron synchronization can be observed for example if the time between the vertical lines on the left of the time diagram is considered. In

0	1	120	249	250
2	122	121	125	251
3	4	116	253	245

P_0	P_1	P_2	P_3	P_4
P_5	P_6	P_7	P_8	P_9
P_{10}	P_{11}	P_{12}	P_{13}	P_{14}

Fig. 10. 3×5 input image to test the LEGION architecture: gray-level values on the left, associated pixel numbers on the right.

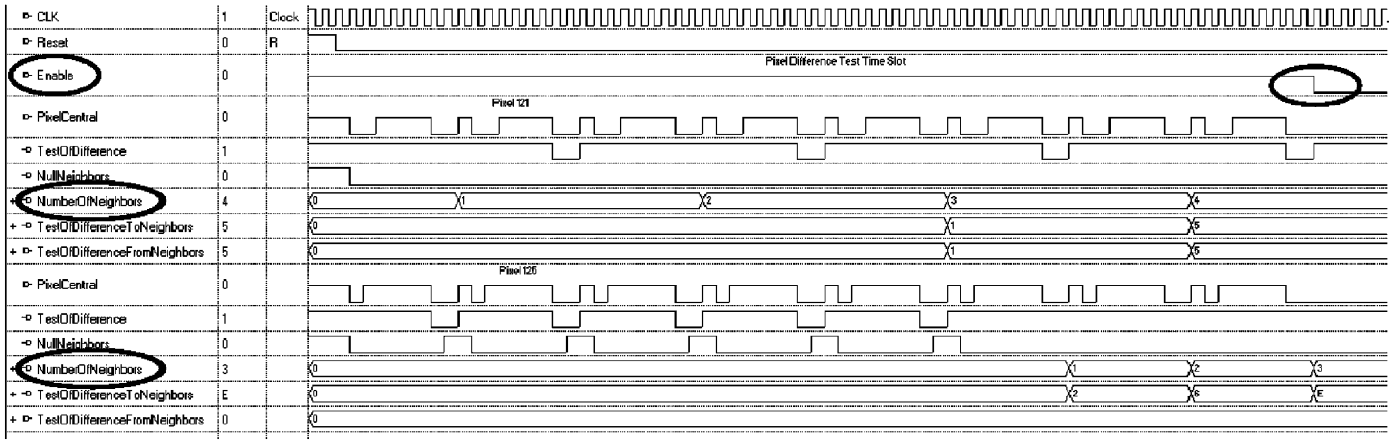

Fig. 8. Difference test timing for two neurons.

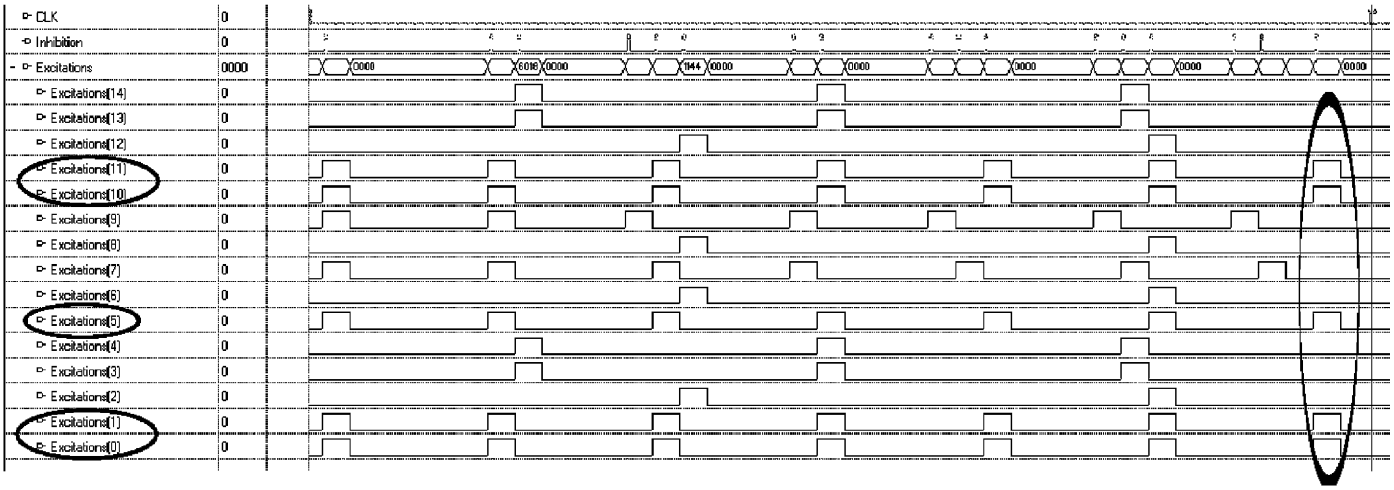

Fig. 9. Time evolution of the oscillatory and synchronizing activity in the network.

this time interval, excitations from neurons 11, 10, 5, 1 and 0, are generated which implies that pixels P11, P10, P5, P1, and P0 are part of the same object in the input image.

Another example is based on the 16×16 image depicted in Fig. 11 (left). As shown in Fig. 11 (right), this example image is particularly difficult to segment, and the LEGION network mostly succeeds in segmenting the borders of the spiral. This is an expected result because the structure of the image produces no "leader" pixel in the body of the spiral. Therefore the network correctly segments the image by separating the leader pixels from the others.

Regarding performance, the image segmentation time is in the order of microseconds for the current implementation and it is mainly related to two main processing stages: the image pre-processing stage to perform the test of difference and leader detection, and the time to reach synchrony. Due to serial processing the difference test between two neighboring pixels requires nine clock cycles. As neighbors are processed sequentially, eight difference test cycles are required. On the other hand, synchrony is easily achieved after a few oscillation cycles ($10-10^2$). For the current implementation, each oscillation cycle requires 15 clock cycles to update the neuron potential and to produce an excitation spike if needed.

According to the implementation results in Table 1, a maximum clock frequency of 77 MHz, the segmentation time for the 16×16 test image is about 3 μs, which is similar to the processing times reported for analog neuromorphic implementations. This time is based on the assumption that the input image is already in the input memory bank and no overhead for image acquisition and transferring is considered. On the other hand, the software implementation on a microprocessor based computer, Pentium 4, 2 GHz and 512 MB, requires around ten milliseconds to segment the image. Thus, the architecture provides a speed factor up to 1000×, that would even increase with the number of neurons (sequential vs parallel implementation). A more fair comparison is difficult to establish since some other factors should be taken into account such as the semiconductor technology used for both platforms. The results demonstrate that the architecture performs segmentation at a higher speed while providing additional advantages as area and power consumption in comparison to a simulation on a microprocessor based computer.

6.4. Modeling and architecture scaling

The experiments described above show good results in the test image segmentation with the specified values for the LEGION parameters. However, more experiments have to be done both for test and real images to analyze the effects of parameter tuning in the image segmentation, as well as the effects of image resolution on the segmentation quality. This problem appears for example when we try to apply a 26×26 integrate-and-fire LEGION network to the image sequence from which Fig. 2 has been extracted. It requires first that the images are reduced to a 26×26 resolution. Segmentation results are then far from being as good as in Fig. 2. The problem appears to be that a lot of details are lost when the image resolution is reduced. It should be pointed out that the 26×26 limit of Table 1 mostly corresponds to synthesis tool limits (the VHDL code is generic but the synthesis is limited to that configuration for most current devices). However, the size of the reported LEGION implementation is in the average for neuromorphic implementations (analog computations).

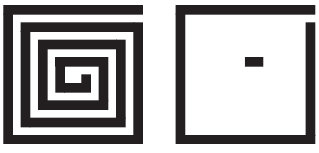

Fig. 11. 16×16 input image to test the LEGION architecture: initial image on the left, an example of segmentation on the right.

If we want to target larger images, the scalability of our architecture must be discussed. The HDL model of the architecture is generic for different LEGION sizes and it could be synthesized for larger neuron arrays if enough hardware resources are available. On the largest current Virtex FPGAs, approximately up to 50×50 neurons might be implemented. The main problems in scaling the architecture in a single FPGA device are internal distributed memory resources for stimulus and neuron potential storage and routing congestion which slows down the architecture performance. For the current implementation, a set of 8×1 RAM memories are required inside the FPGA to store the 8-bit image pixels, as well as an equal number of 16×1 dual port RAMs to store the potentials of neurons, which rapidly becomes a limiting factor for neuron density due to the limited distributed and block RAM resources in the FPGA. This drawback is similar to that found on its analog neuromorphic counterparts. Moreover, as the number of neurons in the network increases, the routing resources demand increases as well. However, due to the local nature of most connections it is not a limiting factor for neuron density but it slows down performance due to global connections (mainly required by the inhibitor module).

One possible approach to cope with architecture scaling is the partition and the implementation of the architecture in multiple FPGA devices. The main problem to overcome under this approach is the I/O bottleneck of FPGA chip packing technology to physically connect the boundary signals among chips (for example the excitation spikes from neighboring neurons among different chips). With a multi-chip implementation where $n \times n$ neurons would be implemented on each FPGA, an FPGA would need at least $18n$ I/O pads for each side having an adjacent FPGA, and such data exchanges would significantly slow down performance.

A possible hardware solution for inter-chip communication is the use of the address event representation (AER) protocol which uses a time-multiplexing technique on

high speed interchip digital buses with a reduced number of pins ensuring that only information carrying neurons consume communication bandwidth. However, high rates of spikes produced at a time interval could compromise performance. A detailed analysis of architecture scaling using a multichip approach is currently being addressed, using both an AER protocol and the principle of duplicate excitation values on each FPGA for the boundary neurons of adjacent FPGAs, so as to reduce performance losses.

7. Conclusion

This paper presents a massively distributed digital implementation of a spiking neural network for image segmentation based on the time oscillatory correlation theory. The results show that efficient implementations of large neural networks can be achieved through the use of specialized datapaths with serial arithmetic. Fast computations supported by lookup tables combined with parallel processing and pipeline increase the performance of the proposed implementation.

Our results confirm that digital devices now appear as promising solutions for embedded systems using spiking models. With respect to previous works, significant performance improvements may be noticed: for example, the results of table 1 outperform those presented in [5] where a simplified architecture is used with binary connection weights and local inhibition. Our results also reach slightly better performances than current works such as [18], but this comparison mostly indicates that common implementation principles appear as key concepts for future implementations, and that complementary approaches of neuron genericity and integrated processing of input and output data should be considered to build the next significant improvement steps.

Our implementation is able to handle low and moderate image resolutions for mobile applications with low area cost in most up to date FPGA technology. The modularity, scalability and high efficient implementation would be beneficial in an embedded system environment for visual perception. The implementation principles presented in this paper bring new perspectives in the digital hardware implementation of spiking neural models and goes forward in the analysis and design of digital VLSI neuromorphic circuits.

References

[1] M. Abeles, Local Cortical Circuits, Springer, Berlin, 1982.
[2] D. Anguita, S. Boni, S. Rindella, A digital architecture for support vector machines: Theory, algorithm, and FPGA implementation, IEEE Trans. Neural Networks 14 (5) (2005).
[3] S. Campbell, D. Wang, C. Jayaprakash, Synchrony and desynchrony in integrate-and-fire oscillators, Neural Computation 11 (1999).
[4] B. Colwell, Machine intelligence meets neuroscience, IEE Computer 38 (1) (2005) 12–15.
[5] D. Fernandes, J. Stedile, P. Navaux, Architecture of oscillatory neural network for image segmentation, in: Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing, 2002.
[6] W. Gerstner, W. Kistler, Spiking Neuron Models - Single Neurons, Populations, Plasticity, Cambridge University Press, Cambridge, 2002.
[7] B. Girau, FPNA: Interaction between FPGA and neural computation, Int. J. Neural Systems 10 (3) (2000).
[8] B. Girau, Neural networks on FPGAs: a survey, in: Proceedings of the Neural Computation, 2000.
[9] H. Hugues, D. Martinez, Encoding in a network of sparsely connected spiking neurons: application to locust olfaction, Neurocomputing (2005) 65–66.
[10] T. Lindblad, J.M. Kinser, Image Processing Using Pulse-coupled Neural Networks, Springer, Berlin, 1998.
[11] W. Maas, Computation with spiking neurons, in: A.M. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, 2003.
[12] N. Mehrtasch, D. Jung, H.H. Hellmich, T. Schonauer, V.T. Lu, H. Klar, Synaptic plasticity in spiking neural networks (sp2inn): A system approach, IEEE Trans. Neural Networks 14 (5) (2003) 980–992.
[13] T. Nordstrom, B. Svensson, Using and designing massively parallel computer for artificial neural networks, J. Parallel Distrib. Comput. 14 (3) (1992).
[14] S.W. Olridge, S.J.E. Wilton, A novel FPGA architecture supporting wide, shallow memories, IEEE Trans. VLSI Systems 13 (6) (2005).
[15] F. Rieke, D. Warland, R. Steveninck, W. Bialek, SPIKES: Exploring the Neural Code, MIT Press, Cambridge, MA, 1998.
[16] O. Rochel, M. Martinez, An event-driven framework for the simulation of networks of spiking neurons, in: Proceedings of ESANN, 2003.
[17] D. Roggen, S. Hofmann, Y. Thoma, D. Floreano, Hardware spiking neural networks with run-time reconfigurable connectivity in an autonomous robot, in: NASA/DoD Conference on Evolvable Hardware, 2003.
[18] B. Schrauwen, J. van Campenhout, Parallel hardware implementation of a broad class of spiking neurons using serial arithmetic, in: Proceedings of ESANN, 2006.
[19] W. Singer, Synchronization, binding and expectancy, in: A.M. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, 2003.
[20] J.G. Taylor, Neural bubble dynamics in two dimensions: foundations, Biol. Cybern. 80 (1998) 5167–5174.
[21] D. Terman, D.L. Wang, Global competition and local cooperation in a network of neural oscillators, Physica D 81 (1995).
[22] D. Terman, D.L. Wang, Locally excitatory globally inhibitory oscillator networks, IEEE Trans. Neural Networks 6 (1) (1995).
[23] L. Ting, R. Woods, C. Cowan, Virtex FPGA implementation of a pipelined adaptive LMS predictor for electronic support measures receivers, IEEE Trans. VLSI Systems 13 (1) (2005).
[24] A. Upegui, C.A. PenaReyes, E. Sanchez, An FPGA platform for online topology exploration of spiking neural networks, Microprocessors and Microsystems 29 (2005) 211–223.
[25] C. von der Malsburg, J. Buhmann, Sensory segmentation with coupled neural oscillators, Biol. Cybern. 67 (1992).
[26] C. von der Malsburg, W. Schneider, A neural cocktail-party processor, Biol. Cybern. 54 (1986).
[27] J. Waldemark, M. Millberg, T. Lindblad, V. Becanovic, Implementation of a pulse coupled neural network in FPGA, Int. J. Neural Systems 10 (3) (2000) 171–177.
[28] D.L. Wang, The time dimension for scene analysis, IEEE Trans. Neural Networks 14 (6) (2005).
[29] D.L. Wang, D. Terman, Image segmentation based on oscillatory correlation, Neural Comput. 9 (1997).
[30] L. Watts, Event-driven Simulation of Networks of Spiking Neurons. in: NIPS, vol. 6, 1993, pp. 927–934.

Bernard Girau graduated from the Ecole Normale Supérieure de Lyon in 1999. He is currently an assistant professor of computer science at Université Nancy 2, and a researcher in the Cortex team of the LORIA INRIA-Lorraine laboratory. His research interest includes embedded parallel connectionism and bio-inspired neural models for visual perception.

Cesar Torres-Huitzil received the Doctor degree in computer science from the National Institute for Astrophysics, Optics and Electronics (INAOE), Puebla, Mexico in 2003. Currently he is a professor/researcher in the Computer Science Department, INAOE. His current research interests include bio-inspired systems and VLSI and FPGA architectures for embedded visual perception.