International Journal of Neural Systems, Vol. 22, No. 4 (2012) 1250014 (11 pages)

DOI: 10.1142/S0129065712500141

World Scientific
www.worldscientific.com

HARDWARE IMPLEMENTATION OF STOCHASTIC SPIKING NEURAL NETWORKS

JOSEP L. ROSSELLÓ, VINCENT CANALS,
ANTONI MORRO and ANTONI OLIVER
Physics Department, Universitat de les Illes Balears
Cra. Valldemossa km. 7.5, Palma de Mallorca
Balears, 07122, Spain
j.rossello@uib.es

Spiking Neural Networks, the last generation of Artificial Neural Networks, are characterized by its bio-inspired nature and by a higher computational capacity with respect to other neural models. In real biological neurons, stochastic processes represent an important mechanism of neural behavior and are responsible of its special arithmetic capabilities. In this work we present a simple hardware implementation of spiking neurons that considers this probabilistic nature. The advantage of the proposed implementation is that it is fully digital and therefore can be massively implemented in Field Programmable Gate Arrays. The high computational capabilities of the proposed model are demonstrated by the study of both feed-forward and recurrent networks that are able to implement high-speed signal filtering and to solve complex systems of linear equations.

Keywords: Stochastic spiking neural networks; neural networks; signal processing; hardware implementation; Gabor filters.

1. Introduction

Artificial Neural Networks are attempting to reproduce the human brain and therefore its amazing properties in terms of perception, signal processing and reasoning. Many efforts have been done to reproduce those systems from the simple perceptron[1] model to the much more realistic and biologically inspired spiking neural model.[2,3] The processing capacities of neural systems arise from the collective behavior of simple processing elements (the neurons). Many research fields demand the use of neural networks due to their great capacity in classification,[4,5] identification[6] associative memory or pattern recognition.[7–10] Neural systems are also present in biological[11] and medical[12–14] applications, control systems,[15,16] prediction and forecasting,[17] optimization,[18,19] and robotics.[20–22] Spiking neural models are also used to understand basic brain behavior.[23,24] This wide range of applications are possible due to the use of different training and evolutionary algorithms.[20,25,26]

The difference between the neural models present in the literature resides in the type of processing of each neural element, ranging from the thresholding of weight-accumulated input spikes to more complex temporal processing and coding based on delta functions (spikes) in spiking neural networks (SNN).[2] The natural evolution of this research field is to find efficient and biologically consistent circuit designs for its hardware implementation since those systems are much more reliable and faster than software simulations. The hardware implementations present in the literature use analog,[27–29] digital[30,31] and also mixed signal circuits,[32,33] but none of them considers the probabilistic nature of information transmission of real biological systems. Spiking Neural Networks are bio-inspired models that use delta functions to represent the Action Potentials of real neurons. The binary (spike) representations lends itself to dense digital circuit implementation and therefore to exploit the processing capabilities of neural systems opening the door to real-time information processing.[34]

An important question about nervous systems is how spike trains are coding the information. There is no question about the use of firing rate codes, but the

Table 1. Main characteristics of timing and rate codes.

	Codification efficiency	Coding/ decoding complexity	Learning complexity
Timing codes	High	High	High
Rate code	Low	Low	Low

existence of more complex codes as delay coding[35] is currently subject of substantial debate.[36] The use of timing codes (dependent on the specific timing of spikes and not only on the firing rate) would require great demands on the spatio-temporal precision with which real neurons need to be connected and the learning process would be quite complex. An advantage of temporal codes is that they can be one or two orders of magnitude more efficient than rate code in terms of information transmission.[37] In Table 1 we compare the two alternative codes showing their shortcomings and advantages.

Depending on the underlying neural activity, spike trains may present a regular[38] or a stochastic behavior.[36] The apparent lack of spike patterns has been one of the principal arguments in favor of firing rate coding with respect to temporal codes. The observed chaotic nature of spike trains is in part due to the mechanism of synaptic transmission that relies on stochastic processes. It have been reported that each synaptic vesicle releases its contents of neurotransmitters from the neuron presynaptic terminal with a given probability.[36] Therefore, the intrinsic stochastic behavior of real biological neurons suggests the need to create nondeterministic stochastic neural models to obtain a more realistic neural behavior.

In this paper we propose a new stochastic-based spiking neural model to be implemented in digital circuitry. The advantage of this approach is the possibility to create large-scale Stochastic Spiking Neural Networks (SSNN) with capacity of parallel computation.

2. Stochastic Spiking Neural Networks

In this section we show how a realistic bio-inspired stochastic neural model can be implemented in hardware to reproduce the observed chaotic behavior of real biological systems. The synaptic transmission of neurons is governed by stochastic processes since each synaptic vesicle releases its contents of neurotransmitter with a given probability once an action potential is generated. If a low number of release sites are present in the pre-synaptic neuron, then the transmission of the pre-synaptic action potential to the other neurons can be modeled by a random variable. We define the probability of signal transmission from the jth to the kth neuron as 'p_{kj}'. If the number of release sites of the jth neuron is high enough, this transmission probability is close to '1' and the connexion is assumed to be reliable. The parameter p_{kj} represents the weight of the connection between two neurons ranging from weak connections (low transmission probability) to strong connections (transmission probability close to one). The inclusion of this probabilistic process creates a stochastic neural behavior in the neural firing. Therefore, probabilistic laws govern spiking signals when they interact with neurons, leading to a wide range of mathematical functionalities.

For the creation of large-scale spiking neural networks, we consider the use of the Integrate-and-fire (IF) neural model due to its simplicity. A perfect integrate-and-fire analogue model consists of a single capacitance that integrates the charge delivered by the synaptic input in addition to a fixed and stationary threshold voltage u_{th} for spike initiation (see Fig. 1). The IF model does not reproduce in detail the time course of action potentials but includes the essential mechanism of neural behavior. Therefore, it is assumed that no information is present in the action potential waveform envelope but on its timing characteristics (inter-spike intervals). An advantage of the *IF* model is that it can be implemented using digital circuitry. In Fig. 2 we show a possible implementation by using a universal shift register. All the bits of the register are stated to a low value except of one single bit that represents the state potential of the neuron. If an excitatory (inhibitory) signal arrives at the input, then the register performs a shift-right (shift-left) change. Once the active bit of the register arrives to the rightmost position it is assumed that the membrane potential of the neuron is surpassing the threshold voltage and the register returns to the resting position. The behavior of the circuit of Fig. 2 can be modeled by a cyclic finite state machine where the presence of a pulse at the excitation (inhibition) signal increases (decreases) the state. All the states provide a zero value (see Fig. 3)

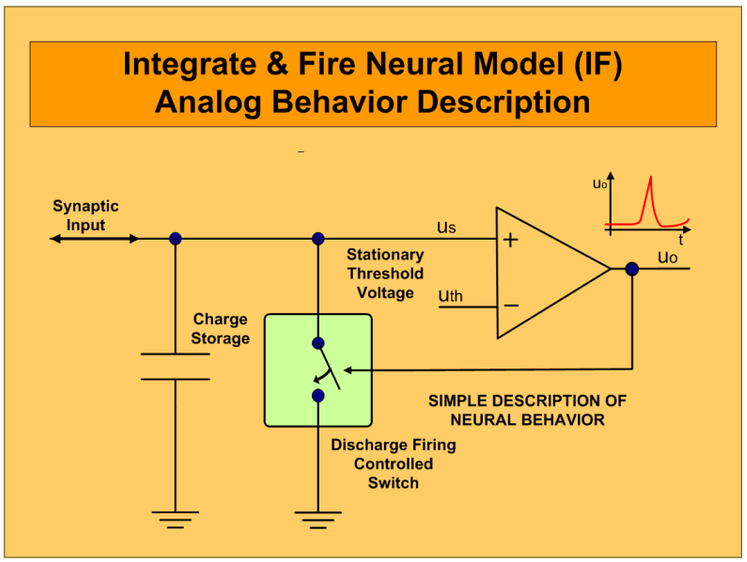

Fig. 1. The Integrate-and-Fire Neural model (*IF*) is one of the simplest descriptions of neural behavior.

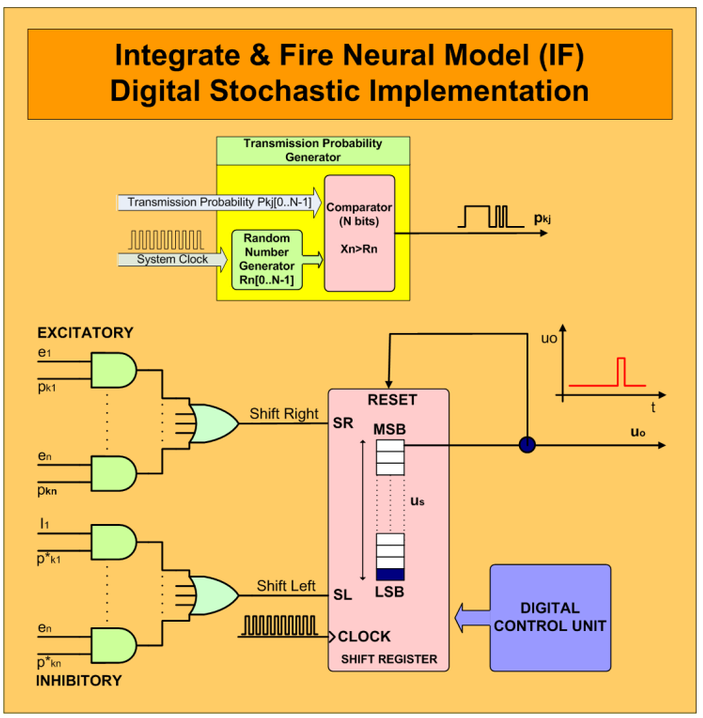

Fig. 2. Hardware implementation of an *IF* model considering stochastic processes.

at the output except the last state (u_7 in Fig. 3) that activates u_{out}. Once the upper state is reached the next is the resting state. The rightmost bit of the register (u_{out}) codifies the presence or absence of an action potential. This bit indicates when the membrane potential is under ($u_{\text{out}} = 0$) or over ($u_{\text{out}} = 1$) the threshold value. If the shift register is simplified to one single bit (replacement of the shift register

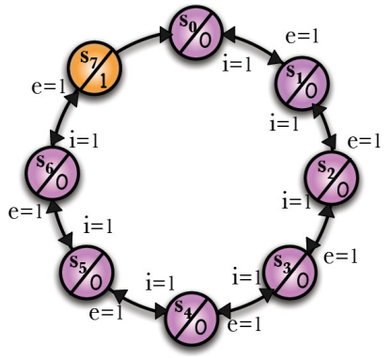

Fig. 3. State machine describing the behavior of the proposed stochastic digital neuron.

by a single D-Flip-Flop) we obtain the most simple implementation of this IF digital model.

Different inputs (excitatory and inhibitory bits e_j and i_j) can be connected to the register by using OR gates. These OR gates implements the addition of all incoming inputs and the membrane potential fluctuates depending on the value of these signals. In addition to the excitatory and inhibitory bits we also include stochastic signals to reproduce the non-deterministic transmission of action potentials (p_{kj}). Each of these stochastic bits fluctuates between 0 and 1 with a given probability. For the generation of these stochastic signals we use a pseudo-random number generator that can be modeled by the finite state machine shown in Fig. 4. The state machine is composed of N states that are randomly connected between them.

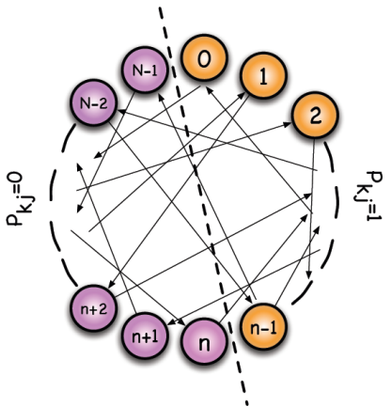

Fig. 4. State machine describing the behavior of the module that generates the stochastic bit p_{kj}.

The signal p_{kj} is activated if and only if the actual state is under the input 'n'. Since the connection between states is chosen randomly, the signal p_{kj} is also random with an activity being proportional to the ratio n/N. Therefore, the transmission of information is dependent on the excitation and inhibition probabilities (p_{kj} and $p*_{kj}$) of incoming signals. The generation of stochastic bits can be obtained by using a pseudorandom (LFSR) or random number generator[39,40] and a comparator (see Fig. 2). At each moment, the output of the comparator (signal p_{kj}) will be equal to zero (if a fixed binary number $\boldsymbol{P_{kj}}$ is lower than the current random number $\boldsymbol{R_n}$) or one. Therefore, when using LFSR systems, the signal p_{kj} switches between one and zero with a probability equal to $p_{kj} = \boldsymbol{P_{kj}}/(2^n - 1)$ (being n the number of bits of binary signals $\boldsymbol{P_{kj}}$ and $\boldsymbol{R_n}$). The main advantage of implementing those stochastic processes is that incoming signals (that also are fluctuating chaotically since they are generated by other stochastic neurons) are weighted by a probability factor (through the AND gates). This is due to the fact that at the AND gate output a switching signal is generated with a probability of change equal to the collision probability of the two inputs. The random number generator can be implemented by using a LFSR or a chaotic circuit.[39,40] A chaotic circuit can be implemented easily inside a FPGA by using digital gates.[40] The methodology proposed in Ref. 40 provides a chaotic bit stream to the output of the circuit. From this chaotic bit stream we can create a random bit stream following the methodology shown in Ref. 39. At the same time the resting voltage value u_o can be fixed or variable.

In Fig. 5 we show the measurements of the digital neural model implemented in a Field Programmable Gate Array (FPGA). It can be appreciated that the state signal u_s increases (decreases) its value each time an excitatory (inhibitory) signal arrives (the two top signals in Fig. 5). Due to the probabilistic nature of input transmission the output signal generated is also stochastic, thus reproducing the Poisson distribution observed in the firing activity of real biological neurons. The proposed stochastic model can be used as a useful tool for exploring the computation capabilities of real biological systems with the added advantage of exploiting the parallelism of hardware implementations (in contrast to software simulations where the models are executed sequentially).

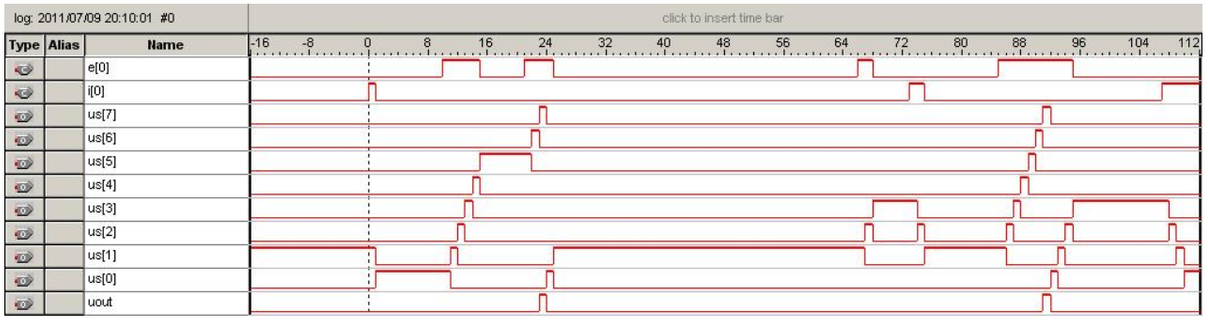

Fig. 5. Measurements of the digital model of SSN. Membrane potential is described by internal state $u_s[7..0]$ while $u_s[7]$ is the response of the neuron (u_{out}). As can be appreciated, excitatory (inhibitory) connexions (the two upper signals) increases (decreases) the state u_s.

The proposed IF digital model of Fig. 2 includes the use of a clock signal. The clock period can be understood as being related to the time response of neurons. Since current FPGAs devices can operate with frequencies of the order of 500 MHz, it implies neural responses of the order of a few nanoseconds.

Other important aspect of the model is that the membrane potential (signal u) is not using a binary representation since only one single bit is activated at each time. This fact implies a higher simplicity since it is not necessary to implement a binary comparator to verify that the potential is over the threshold voltage. Although this method provides a lower precision than a binary representation we obtain a very simple neural model that can be implemented with a low number of digital gates. In the case of the neuron of Fig. 5 the number of logic elements used in an FPGA is of the order of 10 (current FPGA circuits can include to the order of 10^6 logic elements). This fact implies the possibility to implement thousands of neurons working in a parallel mode in a single chip of today's technology.

3. Applications of the Model

3.1. *Feed-forward systems of SSNN can be applied to fast signal filtering*

A well-known property of neurons in the mammalian visual cortex is their ability to implement 2D linear filters to input visual objects.[41,42] Simple receptive fields are able to implement 2D Gabor filters so that neural circuitry is able to respond to images in a given orientation while filtering the others. This kind of filtering has special applications on image processing for edge detection. The main assumption is that this filtering is the first step of a series of neural actions devoted to recognize objects and shapes.

In Fig. 6 we show how a Gabor filter can be implemented using a simple feed-forward spiking neural network with only one layer of neurons. The input image creates a neural activity proportional to the light intensity that stimulates the neural output layer. The weights of the connections (represented by the probability of stimulus transmission) are equal to the impulse-response of the Gabor filter. Therefore, the feed-forward network implements the convolution of the image with this unit impulse response. In the example that is implemented in this work, each neuron at the output layer is connected to nine neurons at the input (that provide the image information). The distribution of these weights is shown in Fig. 7. The values used in Fig. 7 are obtained from the unit impulse response of a Gabor filter.

The main functionality of the filter is to enhance the edges of the image in a pre-defined orientation while attenuating any other characteristic.

To prove the capacity of the proposed system to implement linear filters we implement a simple SSNN with one output layer of 100 neurons. This network is processing a black and white input image of 10×10 pixels. The image to be filtered is shown in Fig. 8 (left image). This image is related to the activity of the input synapses of each neuron. These inputs are assumed to be the output signal of the photoreceptors of the image. All these inputs are random in nature and the switching activity is directly related to the pixel intensity.

In the right image of Fig. 8 we show the activity induced at the output layer of the network. These

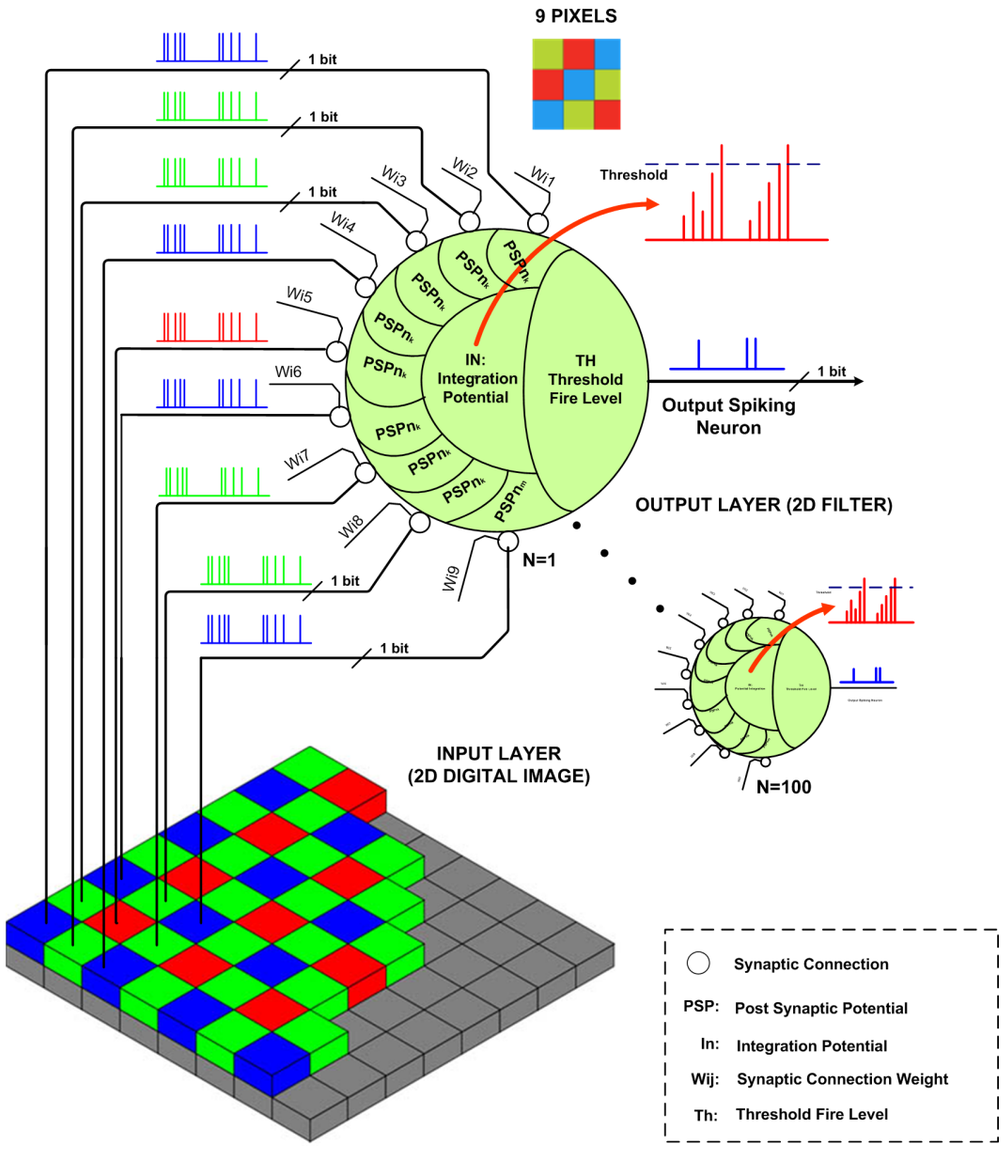

Fig. 6. Application to 2D image filtering. Each neuron of the output layer implements a linear function of an input image.

activities also present a stochastic behavior. The firing rate of the output layer highlights the horizontal edges of the input image while other image features are filtered.

Using similar methodologies any kind of linear filtering (as a complete edge detection system) can be built using a simple feed-forward neural network. Due to the low number of layers used (in this case 1 single layer) the computation speed of the network is quite high since only one single synapse is needed to filter the image. For the implementation of the digital circuit we used an ALTERA Cyclone III FPGA (EP3C16F484C6N) that includes the order of 16,000 logic elements. The computation speed of the network was about 40 ns (that implies a capacity of the order of 25 millions of computations per second). This high computation speed is due to the intrinsic parallelism of neural systems and enables its use

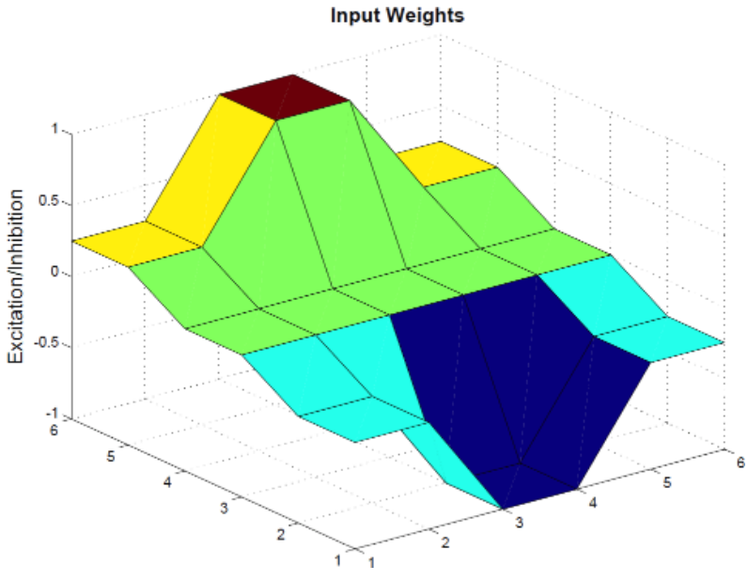

Fig. 7. Weights used for the connection between the input and the output layer. Positive (negative) values represent excitatory (inhibitory) connections.

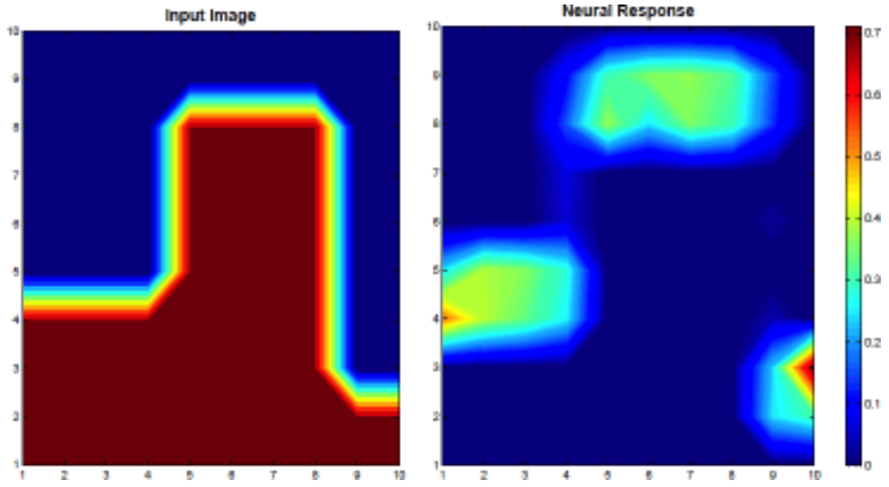

Fig. 8. Neural response (right image) of the feed-forward stochastic spiking neural network to an input image (left).

for ultra high-speed signal processing. Compared to software implementations of Gabor filters, the proposed system is several orders of magnitude faster.[43]

3.2. *Recurrent SSNN are able to solve complex cross-correlated equations*

In the previous subsection we studied the capacity of feed-forward SSNN to implement linear filters (in particular, the implementation of a Gabor filter). In general, feed-forward SSNN are able to implement both spatial and temporal filters to input stimuli. For the case of recurrent SSNN the computation capabilities of networks are more complex. Far from providing an exhaustive study on recurrent SSNN we analyze here the simplest IF digital neural model that is characterized by a shift-register composed of one single bit.

Consider the case in which we have a network composed of n simple stochastic neurons connected in a complete topology. We define u_j and p_{jk} as the switching activity of the jth neuron and the weight of connection from the kth to the jth neuron, respectively. We also define an external excitation i_{j_0} to the jth neuron representing the influence of external environment to the network.

Assuming low values for i_{j_0} and p_{jk} we can estimate the activity of each neuron (u_j) as:

$$u_j \cong i_{j0} + \sum_{k=1}^{n} u_k p_{jk}. \tag{1}$$

This equation can be expressed in a matrix form as:

$$G\mathbf{u} = \mathbf{i}, \tag{2}$$

where $\mathbf{u}$ and $\mathbf{i}$ are n-dimensional vectors representing the internal activity and the external excitation, respectively. $\mathbf{G}$ is a square matrix (that we define as the conductance matrix) being equal to:

$$\mathbf{G} = \mathbf{I} - \mathbf{P}, \tag{3}$$

where $\mathbf{I}$ and $\mathbf{P}$ are the identity and weight matrix, respectively.

Note that (2) is simply a system of n linear equations with n unknown values (internal activities u_j) given an external activity i_{j_0}. The network is able to solve the system (2) since the neural activity (u_j) follows the equation:

$$\mathbf{u} = G^{-1}\mathbf{i}. \tag{4}$$

Therefore, from a known matrix $\mathbf{G}$ (that is dependent on the values of the connection weights) the network is able to quickly obtain its inverse $\mathbf{G}^{-1}$. In Fig. 9 we show the measurements obtained in a recurrent SSNN of 10 neurons connected in a complete topology (thus implying a total of 100 synapses). Results from Fig. 9 have been obtained from a

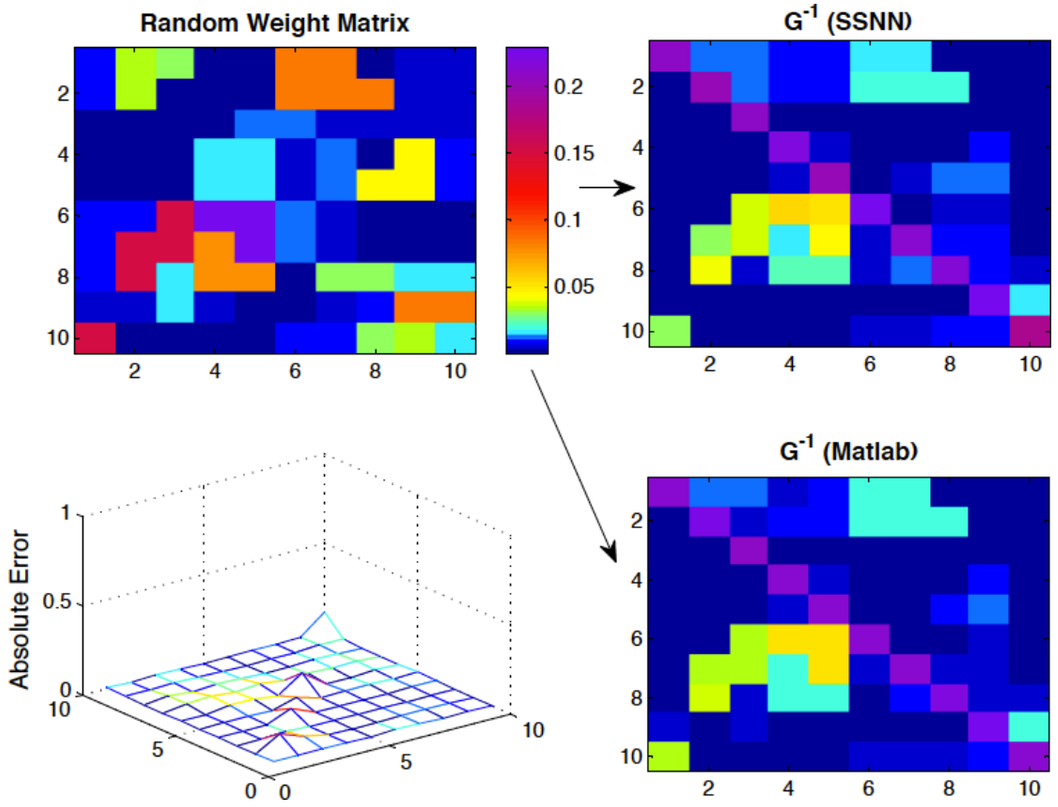

Fig. 9. Estimation of the inverse matrix ($\mathbf{G}^{-1}$) from a random configuration (a 10×10 weight matrix $\mathbf{P}$). The results obtained with the SSNN are nearly the same than the real values (MATLAB simulations).

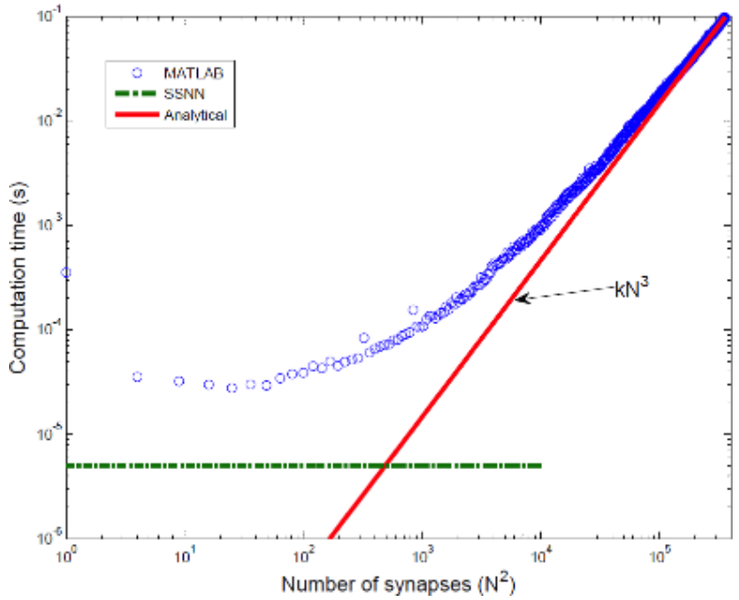

Fig. 10. Performance comparison between SSNN and MATLAB.

random connection matrix $\mathbf{P}$ and the inverse matrix $\mathbf{G}^{-1}$ is taken from the measured neural activity in the FPGA.

The comparison of the performance of neural systems and processor-based computers is quite complex due to the different nature of both systems. The results provided here regarding the capacity of solving n-dimensional linear systems opens an opportunity of comparing the two systems. The capacity of neural networks relies in the high parallelism that can be achieved with those systems while processor-based methods are intrinsically sequential processes.

In Fig. 10 we compare the computation time needed to solve an N-dimensional linear system between a SSNN and MATLAB. We compare with MATLAB since this is a mathematical tool that is commonly used by scientists so it can be used as a first reference. In the graph we show the computation time with respect to the number of synapses needed to solve the system by using a recurrent SSNN (N^2). It is found that MATLAB needs approximately a computation time proportional to N^3 while the computation time of neural networks is held constant due to the parallelism. The difference is higher than one order of magnitude although the operation frequency of both systems is quite different (50 MHz for the FPGA used to implement the SSNN and 2.26 GHz used by an Intel Core i5-430M for the MATLAB results).

4. Discussion

Neural Networks are very attractive computational systems due to their intrinsic parallelism and their ability to implement any kind of computation. To study those systems it is interesting to use biologically inspired digital models due to the capability of current technology to implement dense digital circuits in FPGAs. In this paper we propose a simple digital implementation of a stochastic Integrate-and-Fire spiking neuron that is able to mimic some of the most significant properties of biological neurons. The inclusion of the intrinsic probabilistic nature of signal transmission enables the introduction of weights in neural connections. It has been shown that feed-forward and recurrent Neural Networks are able to implement signal filters and to solve large systems of cross-correlated equations. The proposed digital circuit can be used to experiment with complex neural systems in order to understand the behavior of real neural circuitry. It is also a useful way to implement ultra high-speed image processing systems.

5. Conclusion

In this paper we proposed a new spiking neural model that can be implemented directly in hardware to exploit the intrinsic parallelism of neural systems. This represents an advantage with respect to experiments based on simulations. The novelties of this work are:

(1) Presentation of the SSNN that is based on the Integrate-and-fire model but including the intrinsic stochastic processes of real biological neurons. Therefore the model has a biologically inspired nature.
(2) Two modes of operation are studied:
   - Feed-Forward SSNN that allow its use as linear filters. This filtering capacity is measured with an FPGA implementation. The results presented here are related to Gabor filters (that we know that the mammalian cortex performs, as shown in Refs. 41 and 42).
   - Recurrent SSNN. In this case we show a different mathematical property of SSNN. In the recurrent mode they are able to solve linear systems of equations.

Acknowledgments

This work was supported in part by the European Development Funds (FEDER) and in part by the Spanish Ministry of Economy and Competitiveness under the project TEC2011-23113

References

1. F. Rosenblat, The perceptron — A probabilistic model for information-storage and organization in the brain, *Physiol. Rev.* **66**(6) (1958) 386–408
2. W. Maas, Networks of spiking neurons: The third generation of neural network models, *Neural Networks* **10**(9) (1997) 1659–1671.
3. S. Ghosh-Dastidar and H. Adeli, Spiking neural networks, *Int. J. Neural Syst.* **19**(4) (2009) 295–308.
4. H. N. A. Hamed, N. Kasabov, S. M. Shamsuddin, H. Widiputra and K. Dhoble, An extended evolving spiking neural network model for spatio-temporal pattern classfication, in *Proc. Int. Joint Conf on Neural Networks* (2011), pp. 2457–2464.
5. J. J. Wade, L. J. McDaid, J. A. Santos and H. M. Sayers, SWAT: A spiking neural network training algorithm for classification problems, in *Proc. Int. Joint Conf on Neural Networks* (2010), pp. 1817–1830.
6. A. Khashman, Blood cell identification using a simple neural network, *Int. J. Neural Syst.* **18**(5) (2008) 453–458.
7. S.-Y. Fu, G.-S. Yang and Z.-G. Hou, Spiking neural networks based cortex like mechanism: A case study for facial expression recognition, in *Proc. Int. Joint Conf. Neural Networks* (2011), pp. 1637–1642.
8. S. Soltic and N. Kasabov, Knowledge extraction from evolving spiking neural networks with rank order population coding, *Int. J. Neural Syst.* **20**(6) (2010) 437–445.
9. E. J. Agnes, R. Erichsen Jr. and L. G. Brunnet, Model architecture for associative memory in a neural network of spiking neurons, *Phys. A* **391**(3) (2012) 843–848.
10. T. Natschläger and B. Ruf, Spatial and temporal pattern analysis via spiking neurons, *Netw.: Comput. Neural Syst.* **9**(3) (1998) 319–332.
11. F. Rasheed, M. AlShalalfa and R. Alhajj, Adapting machine learning technique for periodicity detection in biological sequences, *Int. J. Neural Syst.* **19**(1) (2009) 11–24.
12. S. Ghosh-Dastidar and H. Adeli, Improved spiking neural networks for EEG classification and epilepsy and seizure detection, *Integrated Comput.-Aided Eng.* **14**(3) (2007) 187–212.
13. S. Ghosh-Dastidar and H. Adeli, A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection, *Neural Networks* **22**(10) (2009) 295–308.
14. N. Kasabov, R. Schliebs and H. Kojima, Probabilistic comutation neurogenetic modelling: From cognitive systems to alzheimer's disease, *IEEE Trans. Auton. Mental Develop.* **3**(4) (2011) 300–311.
15. D. Theodoridis, Y. Boutalis and M. Christodoulou, Indirect adaptive control of unknown multi-variable nonlinear systems with parametric and dynamic uncertainties using a new neuro-fuzzy system description, *Int. J. Neural Syst.* **20**(2) (2010) 129–148.
16. P. Arena, L. Fortuna, M. Frasca and L. Patané, Learning anticipation via spiking networks: Application to navigation control, *IEEE Trans. Neural Netw.* **20**(2) (2009) 202–216.
17. H. Y. Yamin, S. M. Shahidehpour and Z. Li, Adaptive short-term electricity price forecasting using artificial neural networks in the restructured power markets, *Int. J. Electr. Power Energy Syst.* **26**(8) (2004) 571–581.
18. H. S. Park and H. Adeli, Optimization of space structures by neural dynamics, *Neural Networks* **8**(5) (1995) 769–781.
19. N. Kasabov and H. N. A. Hamed, Quantum-inspired particle swarm optimisation for integrated feature and parameter optimisation of evolving spiking neural networks, *Int. J. Artif. Intell.* **7**(11A) (2011) 114–124.

20. N. R. Luque, J. A. Garrido, R. R. Carrillo, S. Tolu and E. Ros, Adaptive cerebellar spiking model embedded in the control loop: Context switching and robustness against noise, *Int. J. Neural Syst.* **21**(5) (2011) 385–401.
21. S. P. Johnston, G. Prasad, L. Maguire and T. M. McGinnity, An FPGA hardware/software co-design towards evolvable spiking neural networks for robotics applications, *Int. J. Neural Syst.* **20**(6) (2010) 447–461.
22. E. Nichols, L. J. McDaid and N. H. Siddique, Case study on a self-organizing spiking neural network for robot navigation, *Int. J. Neural Syst.* **20**(6) (2010) 501–508.
23. R.-M. Memmesheimer, Quantitative prediction of intermittent high-frequency oscillations in neural networks with supralinear dendritic interactions, *Proc. Natl. Acad. Sci. USA* **107**(24) (2010) 11092–11097.
24. A. Vidybida, Testing of information condensation in a model reverberating spiking neural network, *Int. J. Neural Syst.* **21**(3) (2011) 187–198.
25. T. J. Strain, L. J. McDaid, T. M. McGinnity, L. P. Maguire and H. M. Sayer, An STPD training algorithm for a spiking neural network with dynamic threshold neurons, *Int. J. Neural Syst.* **20**(6) (2010) 463–480.
26. S. Schliebs, N. Kasabov and M. Defoin-Platel, On the probabilistic optimization of spiking neural networks, *Int. J. Neural Syst.* **20**(6) (2010) 481–500.
27. Q. Sun, F. Schwartz, J. Michel, Y. Herve and R. Dal Molin, Implementation study of an analog spiking neural network for assisting cardiac delay prediction in a cardiac resynchronization therapy device, *IEEE Trans. Neural Netw.* **22**(6) (2011) 858–869.
28. A. van Schalik, Building blocks for electronic spiking neural networks, *Neural Networks* **14** (2001) 617–628.
29. G. Indiverdi, E. Chicca and R. Douglas, A VLSI array of low-power spiking neurons and biestable synapses with spike-timing dependent plasticity, *IEEE Trans. Neural Netw.* **17**(1) (2006) 211–221.
30. T. Hishiki and H. Torikai, A novel rotate-and-fire digital spiking neuron and its neuron-like bifurcations and responses, *IEEE Trans. Neural Netw.* **22**(5) (2011) 752–767.
31. L. Tigaeru and G. Bonteanu, A neuron model for FPGA spiking neuronal network implementation, *Adv. Electr. Comput. Eng.* **11**(4) (2011) 29–36.
32. J. L. Rosselló, V. Canals and A. Morro, Chaos-based mixed signal implementation of spiking neurons, *Int. J. Neural Syst.* **19**(6) (2009) 465–471.
33. S. Cawley, F. Morgan, B. McGinley, S. Pande, L. McDaid, S. Carrillo and J. Harkin, Hardware spiking neural network prototyping and application, *Genet. Programm. Evol. Mach.* **12**(3) (2011) 257–280.
34. B. Schrauwen, M. D'Haene, D. Verstraeten and J. V. Campenhout, Compact hardware liquid state machines on FPGA for real-time speech recognition, *Neural Networks* **21**(2–3) (2008) 511–523.
35. T. Natschlager and B. Ruf, Pattern analysis with spiking neurons using delay coding, *Neurocomputing* **26–27** (1999) 463–469.
36. C. Koch, *Biophysics of Computation, Information Processing in Single Neurons* (Oxford University Press, New York, 1999).
37. F. Rieke, D. Warland and R. R. D. Van Steveninck, *Exploring the Neural Code* (MIT Press: Cambridge Massachusetts, London, 1999).
38. A. F. Jahangiri and D. M. Durand, Phase resetting of spiking epileptiform activity by electrical stimulation in the CA3 region of the Rat Hippocampus, *Int. J. Neural Syst.* **21**(2) (2011) 127–138.
39. J. L. Rosselló, V. Canals, I. de Paúl, S. Bota and A. Morro, A simple CMOS chaotic integrated circuit, *IEICE Electron. Express* **5**(24) (2008) 1042–1048.
40. R. Zhang *et al.*, Boolean chaos, *Phys. Rev. E* **80**(4).
41. J. G. Daugman, Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters, *J Opt. Soc. Am.* **2**(7) (1985) 1160–1169.
42. J. P. Jones and L. A. Palmer, An evaluation of the two-dimensional gabor filter of simple receptive fields in cat striate cortex, *J. Neurophysiol.* **58**(6) (1987) 1233–1258.
43. H. Wu, Y. Yoshida and T. Shioyama, Optimal gabor filters for high-speed face indentification, in *Proc. Int. Conf. Pattern Recognition* **16**(1) (2002) 107–110.