Population Coding and SpikeProp Hardware Accelerator for Spiking Neural Networks

Marco Aurelio Nuño-Maganda[1], Cesar Torres-Huitzil[2], and Miguel Arias-Estrada[3]

[1] Universidad Politécnica de Victoria (UPV), Av. Nuevas Tecnologías S/N, Parque Científico y Tecnológico de Tamaulipas, C.P. 87137, Ciudad Victoria, Tamaulipas, México
[2] CINVESTAV-Tamaulipas, Information Technology Laboratory, Parque Científico y Tecnológico de Tamaulipas, C.P. 87137, Ciudad Victoria, Tamaulipas, México
[3] National Institute for Astrophysics, Optics and Electronics (INAOE), Luis Enrique Erro #1, San Andrés Cholula, C.P. 72840, Puebla, México

Abstract. Spiking Neural Networks (SNNs) have become an important research topic due to new discoveries and advances in neurophysiology, which suggest that information among neurons is coded, interchanged and processed via pulses or spikes. Two important aspects of SNNs are coding and learning as means to represent information and acquire knowledge by experience, respectively. The SpikeProp algorithm has been proposed as a learning algorithm for SNNs with good results in classification and pattern discrimination. SpikeProp algorithm requires a coding scheme called Gaussian Receptive Fields (GRFs), which converts real data to firing times. In this paper we explore the feasibility of using FPGAs for developing specialized hardware modules for GRF coding and for large scale simulations of SNNs. Modularized processing elements were designed to evaluate different implementation tradeoffs and to promote scalability of the model to larger FPGAs. Results and performance statistics are presented, as well as a discussion of implementation trade-offs using pattern classification problems as a case study.

1 Introduction

Spiking Neural Networks (SNNs), classified as neural models of third generation [1], have become an important research area due to its biological plausibility. Roughly speaking, the behavior of ideal spiking neurons is described as follows: A typical neuron can be divided into three functionally distinct parts, called dendrites, soma and axon. The dendrites play the role of the "input device" that collects signals from other neurons and transmit them to the soma. The soma is the "central processing unit" that performs a nonlinear processing step. If the total input exceeds a certain threshold, then an output signal is generated. Then, the output signal is converted by the "output device", the axon, which delivers the signal to other neurons. The biological neuronal signals consist of short electrical pulses. The pulses, the so-called action potentials or spikes, have an amplitude of about 100 mV and typically a duration of 1-2 ms [2]. It is recognized

Advances in Computer Science and Engineering.
Research Computing Science 45, 2010, pp. 145-156

Received 30/04/10
Final version 19/05/10
Accepted 14/05/10

that SNNs are capable of exploiting time in a sophisticated manners a resource for information coding and computation. Learning in traditional, artificial neural networks is usually performed by gradient ascendant/descendant techniques to find the network parameters to perform a given task. For SNNs these techniques are extrapolated and the SpikeProp algorithm, has been proposed for learning [3] [4] [5] [6] [7] for neurons modeled by the Spike Response Model (SRM) [8].

The motivation of digital implementations of SNNs is as diverse as the background of the researchers, but most implementation look for performance speedup and/or a large scale simulations. Spiking models are less hardware-greedy than classical models, and accuracy results compared to classical neural networks are similar [9]. Due to programmability, digital hardware offers a high degree of flexibility and provides a platform for simulations on neuron level as well as on network level [10]. In [11], several hardware platforms used for simulation of SNNs are reported, where the network sizes range from 8K to 512K neurons. The presented results concluded that only supercomputers can cope with the computer processing demands of these type of networks, however other alternative exist, such as FPGA-based computing, for such kind of processing.

This paper is divided in 5 parts. Section 2 presents an overview of coding schemes and recall phase of feedforward SNNs. Section 3 describes the proposed architecture for each stage of the SNN processing. In Section 4, experimental results about performance of hardware blocks are presented, with a tradeoff discussion. Finally, in section 5, conclusion and future work are presented.

2 Background

2.1 Population Coding using Gaussian Receptive Fields (GRFs)

The application of GRFs for supervised learning was introduced in [3], where a set of analog variables is fed into the GRFs to convert them in pulse streaming suited for the SNNs processing and learning. A Gaussian function is defined by:

$$f(x) = ae^{\frac{(x-b)^2}{2\sigma^2}} \tag{1}$$

for real constants $a > 0$, the height of the Gaussian peak, $b > 0$, the position of the center of the peak and $\sigma > 0$ controls the width.

A real input value is encoded by an array of receptive fields. The range of the data is first calculated, and then, each input feature is encoded with a population of neurons that cover the whole data range. For a variable with a range $[I^n_{min}, ..., I^n_{max}]$, a set of m GRF neurons are used. The center b_i of each GRF neuron is determinated by:

$$b_i = I_{min} + \frac{1}{m-2}\frac{2i-3}{(I_{max} - I_{min})}, \tag{2}$$

And the width σ of each GRF neuron i is determinated by:

$$\sigma = \frac{1}{(m-2)}\frac{1}{\beta(I_{max} - I_{min})}, \tag{3}$$

where the proposed value for β belongs to the range $[1, 2]$.

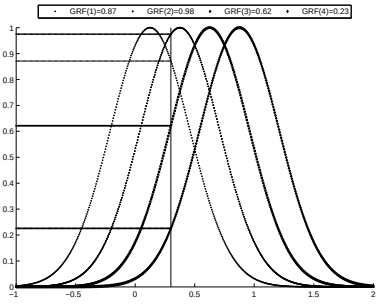


Fig. 1. Example of GRFs coding

The process of coding an analog variable is shown in figure 1, where the total number of GRFs is four, and the overlapped GFs are plotted together. The value to be coded is 0.3 (shown as a vertical line), and the evaluation of that value in each GRF is shown by a horizontal line.

2.2 Spiking processing in recall phase

A neuron j, having a set Γ_j of immediate predecessors ("pre-synaptic neurons"), receives a set of spikes with firing times $t_i, i \in \Gamma_j$. Any neuron generates at most one spike during the simulation interval, and fires when the internal state variable reaches a threshold ϑ. The dynamics of the state variable $x(t)$ are determined by the input spikes, whose impact is described by the spike-response function $\Delta(t)$ weighted by the synaptic efficacy ("weight") w_{ij}:

$$x_j(t) = \sum_{i \in \Gamma_j} w_{ij} \varepsilon(t - t_i) \tag{4}$$

A network consists of a fixed number of m synaptic terminals (or connections), where each terminal serves as a sub-connection that is associated with a different delay d and weight w. The unweighted contribution of a single synaptic terminal k to the state variable y is given by:

$$y_i^k = \varepsilon(t - t_i - d^k) \tag{5}$$

where $\epsilon(t)$ is the spike response function, with $\varepsilon(t) = 0$ for $t < 0$. The time t_i is the firing time of a previous neuron i, and d^k is the delay associated with the synaptic terminal k. The spike response function is given by:

$$\varepsilon(t) = \frac{t}{\tau} e^{(1 - \frac{t}{\tau})} \tag{6}$$

Extending (4) to include multiple synapses per connection and, substituting in (5) the state variable x_j of neuron j receiving input from all neurons i, the network can be described as the weighted sum of the pre-synaptic contributions:

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} (w_i j^k y_i^j(t)) \tag{7}$$

The target of learning through the SpikeProp algorithm is to get a set of firing times t_j^k, at the output neurons $j \in J$ for a given set of input patterns $P[t_1...t_h]$, where $P[t_1...t_h]$ denotes a single input pattern described by single firing times for each neuron $h \in H$. The error-function is defined by:

$$E = \frac{1}{2} \sum_{j \in \Gamma_j} (t_j^a - t_j^d)^2 \tag{8}$$

where t_j^d are the desired firing times and t_j^a are the actual firing times. The details of the SpikeProp algorithm are out of the scope of this paper, but it is being considered for further on-chip implementation. See [3] for further details on SpikeProp.

3 Proposed Architecture

The dataflow for the architecture is shown in figure 2. The input data set is organized in rows and columns. The columns are the dataset attributes, while the rows are the dataset samples. The class column is stored in a separated memory region. The input dataset is passed through the GRFs, which main function consists of codifying the input data into firing times. Once this transformation has been carried out, the firing times are passed to the SNN module, which computes the output as firing times. Then, the output firing times are passed to a class decoder, which obtains the class assigned by the network to the input pattern. Finally, the network performance can be evaluated, comparing the class obtained by the network with the original class assigned to that pattern. Additionally, a hardware-based learning module can be implemented for adjusting weights and delays for learning from a dataset for a specific application.

3.1 Architectural Overview

The proposed hardware for carrying out the dataflow previously explained is shown in figure **??**. A system with 2 data buses is proposed. The Global Data Bus connects external memory elements with internal routers. The internal routers send data to each one of the processing elements. The processing elements read from the bus their input data, process them and output the results on other data bus, which feed an output router. The output router can send data to memory, or to other router (the input router of other process).

The control bus contains all the control signals generated by the global control unit. The control unit generates the synchronization signals required for each

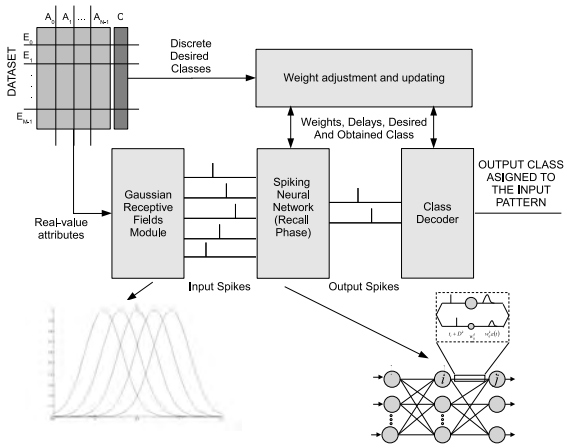


Fig. 2. Architectural dataflow

module in the proposed architecture. There are two type of processors. The first one is the Gaussian Receptive Fields Processor (GRPF), which transform the input data into firing times. The second one is the Spiking Neural Layer Processor (SNLP), which obtains the network output for a given input firing times pattern. These processors are explained more in detail in this section.

The proposed architecture reads the first k-columns of the dataset, and the GRF coding is applied to that input data (using one or more GRFPs) for obtaining the input firing times corresponding to those input data. Later, the architecture takes the input firing times and generates the network output (or output firing times). Depending of the layers implemented in the SNN, is the number of implemented SNLPs (at least one SNLP must be implemented for 2-layer SNN). This process is repeated until all the patterns in the dataset (or a number of patterns previously established) have been evaluated.

Gaussian Receptive Field Processor In figure 4, the main components of the GRFP are shown. The main components are described below:

- *Internal Control Unit.* This component receives control signal from Global Control Unit and generates the appropriate control signal for the components of the GRFP.
- *Maximum-Minimum-Range Computation Module (MMRCM).* This module accesses to the dataset and computes maximum, minimum and range for each data column of the dataset. This module can be excluded from the architecture or not synthesized if these parameters are known or were precomputed before coding. The results obtained by this module are stored in the Maximum-Minimum-Range Memory(MMRM).
- *Centroid Computation Module (CCM).* This module computers the gaussian centroids as defined by equations 2 and 3. The total amount of centroids

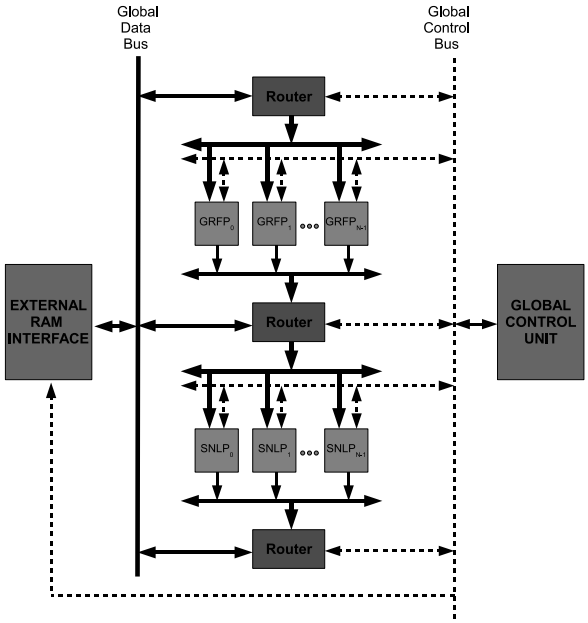


Fig. 3. Complete architecture with GRFPs and SNLPs modules

required depends of the number of gaussian fields required. The results obtained by this module are stored in the Centroid Memory (CM).

- *Parameters Memory (PM).* The statistics and centroids are stored in this memory and later sent to each GP for the Gaussian Field Computation.
- *Temporal Registers (TR).* Data from the input dataset are stored in an array of temporal registers. Each one of the GPs accesses to this register for obtaining the corresponding firing times according to the number of gaussian fields processor outputs.
- *Gaussian Processor (GP).* This module performs the computation of each gaussian function of the gaussian array. Each GP reads data from all the TRs and obtains the Gaussian codification of that value, and later the results are stored in the corresponding FT.
- *Firing Times Memory (FTM).* The firing times generated for each GP are stored in this memory. Later, the data are sent to the router for storing in external memory or to be sent to the SNLP.

The proposed architecture for the GRFP is designed to be flexible and modular depending of the required degree of parallelism. The parameters for architecture compilation that can be set for this processor are the number of gaussian fields, the width of the gaussian fields and the gaussian fields separation.

Spiking Neural Layer Processor (SNLP) This processor performs the recall phase in SNNs, which consists on the firing time computation of both hidden

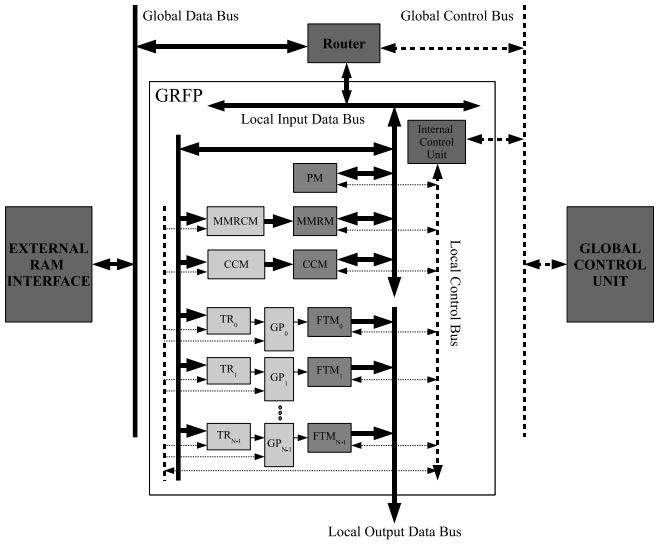


Fig. 4. Main components of a GRFP

and output neurons layers. The neural computation takes as input the firing times generated by the GRFPs. In the actual implementation, this processor is designed for working only with feed-forward SNNs, but it can be modified for supporting other connectivity schemes. The quality and precision of the GRF coding performed by the GRFPs is a critical factor for the network performance (related to the classification accuracy, not performance). The processor contains a set of modules called Neural Processors (NPs), which are grouped in layers depending of the requirements of the implemented SNN. A SNLP contains a set of fixed NPs, which are assigned for computing the neuron firing time for exactly one network layer. As reported in several applications, for the proposed architecture only two layers of Spiking neurons are defined (and only 2 SNLPs are implemented), but the proposed architecture is designed for implementing more neuron layers by defining more than one SNLP.

4 Implementation and Results

4.1 FPGA implementation

The proposed architecture was implemented and synthesized on an Alphadata ADM-XPL board, which hosts a Virtex- II PRO FPGA. The hardware resources for the target FPGA device are shown in table 1. The target platform has a PCI interface, thus it is hosted on a desktop PC. The proposed architecture is fully modeled using the Handel-C Hardware Description Language.

The proposed architecture was designed for supporting as many processors as possible, limiting the implementation to the hardware resources available on the

target FPGA. In order to explore the parallelism tradeoffs, different implementations were synthesized. In table 2, the hardware resource utilization for each implementation is shown. For validating the GRF coding, only one GRFP module was synthesized (In table 2, second column). For validation the GRF coding and network performance, several NPs (4,8, and 16 NPs) per SNLP (only 2 SNLPs) were synthesized (In table 2 from third to fifth column).

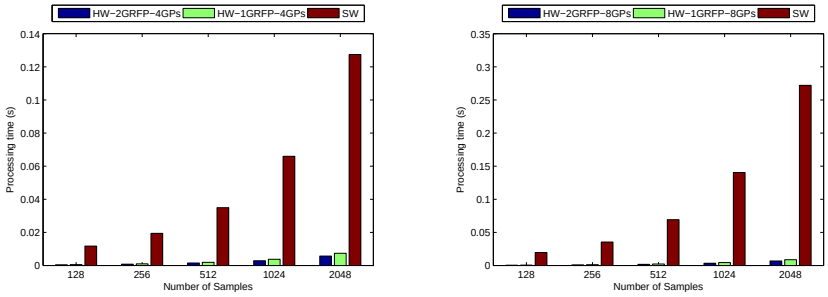
Table 1. Characteristics of the target FPGA device

FPGA	4-Input LUTs	Slice-FFs	Total Slices	Total BRAMs	Total MULT18x18s
xc2vp30-6ff896	27,392	27,392	13,696	136	136

4.2 Performance results

The implemented architecture was tested using several variations of the input parameters. The input simulation parameters that can be set are:

- Number of examples (rows) and variables (columns) of the input dataset.
- Number of gaussian fields used for each variable (the number of input neurons is given by the product of total variables by the number of gaussian fields associated to each variable).



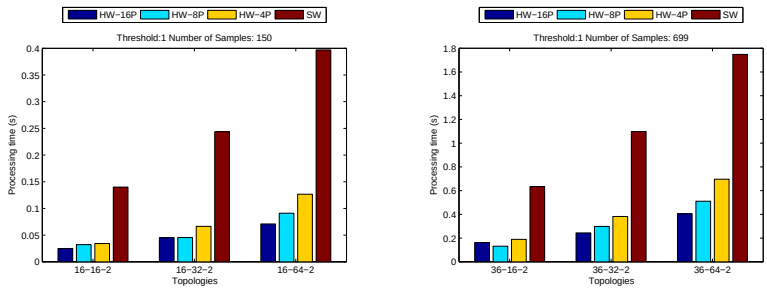
(a) 4 GRFs (b) 8 GRFs

Fig. 5. HW versus SW execution time comparison for the GRFPs

Execution time comparison for the coding module is shown in figure 5, where the "SW" label means execution time in software, on a PC with a Pentium-4 processor running at 2.66 GHz. The "HW-2GRFP-4GPs" label means execution

time in hardware using 2 GRFPs with 4 GPs, the "HW-1GRFP-4GPs" label means execution time in hardware using 1 GRFP with 4 GPs, the "HW-2GRFP-8GPs" label means execution time in hardware using 2 GRFPs with 8 GPs and the "HW-1GRFP-8GPs" label means execution time in hardware using 1 GRFP with 8 GPs. The number of computed GRFs is 4 GRFs for figure 5(a) and 8 GRFs for figure 5(b).

Execution time comparison for the overall architecture is shown in figure 6, where the "SW" label means execution time in software, on a PC with a Pentium-4 processor running at 2.66 GHz. The HW-XP means execution time in hardware with 2 SNLPs with X NPs each. The experiments were performed with real data obtained from standard datasets used in machine learning applications. For the first experiment, the Iris dataset from the UCI Repository of Machine Learning Databases [12], with 150 instances and 4 discrete attributes was used. The performance for several topological variations of the implemented SNN for the Iris dataset is shown in figure 6(a). For the second experiment, the Wisconsin Breast Cancer (WBC) dataset from the UCI Repository of Machine Learning Databases [12], with 699 instances and 9 attributes was used. The performance for several topological variations of the implemented SNN for the WBC dataset is shown in figure 6(b). The reported execution time is the total amount of time required for both hardware and software implementations to perform the network output computation for the entire dataset.



(a) IRIS dataset (b) WBC dataset

Fig. 6. HW versus SW execution time comparison (real datasets)

4.3 Discussion

The architecture described in this paper is an extension of work reported in the past. A first attempt for implementing an FPGA-based architecture for Spike-Prop is reported in [13], using the original Spikeprop algorithm reported in [3]. The proposed architecture was validated by obtaining the same classification results as the original application. In a second attempt [14], several improvements

Table 2. Hardware resources for the complete architecture

Resources	GRFs ONLY	8 NPs	16 NPs	24 NPs
Slices	4,166 30%	6,843 50%	8,225 60%	9,711 71%
Max clock freq (Mhz)	66,8	66,84	65,17	64,22
Gate Count	458,161	2,523,393	4,155,493	5,791,707

to SpikeProp were integrated into the architecture previously proposed. It is specially interesting the reduction of the number of synaptic terminal from 16 synaptic connections in the original SpikeProp algorithm to 2 synaptic connections reported in [15], and the adaptations of RProp and QuickProp algorithm established in [4]. The recall phase is fully implemented, but the learning is validated only by an off-chip execution. In the first and second attempts, the coding problem was not explored. In a third attempt [16], a hardware architecture for the gaussian receptive fields coding is reported. Tests with randomly generated datasets are reported. In a fourth attempt [17], a generic hardware core for obtaining the network output for multilayer SNNs based on SpikeProp algorithm is reported. A previous version of the learning implementation is reported, as well as an estimation of hardware resources and performance. In this work, an integrated architecture and new improvements are reported. The novelty of the present work consists on the flexibility and scalability of the proposed architecture, as a compact core that can be used for a wide variety of classification problems, allowing the tunning of several parameters for both coding and recall phases of SNNs and improving in performance.

In the first part of the chain of processing of the proposed architecture, the GRF coding is performed using GRFPs. When synthesizing the proposed architecture with only GRFPs (the SNLPs were excluded), execution time results were obtained, and the obtained performance improvement falls in the range from 4X to 16X depending of the number of implemented GPs. The maximum number of GRFs that can be computed is established to 4, as proposed in [4], since 4 if good enough for obtaining an acceptable network performance, but if more GRFs are required, the architecture can be synthesized for fitting that requirement. These improvement rates are obtained because each GRF is mapped in one separated module, and several patterns (when adding more GRFPs) can be processed in parallel. About 30% of the target FPGA device is used when only GRF coding is implemented. The implemented GRFP uses only 4 GPs. If more GRFPs and GPs are required, then the architecture can be extended, but this involves more hardware resources for the GRFs coding. Using the implemented cores only for coding, at least a 2X performance improvement is obtained, and

for the performed experiments, a maximum performance improvement of 16X is obtained.

In the second part of the processing chain, the SNLP are added to the implemented architecture. The obtained performance improvement is from 4X to 20X depending of the number of NPs. About 50% of the FPGA device is used, with 1 GRFP and 2 SNLPs implemented. This core combination can be considered as the "minimal" implementation that functionally can achieve all the computations required for the SNNs described in this work. When using the "minimal" architecture, at least a 3.5X performance improvement is obtained for the smallest tested network, and a maximum 9.5X performance improvement for the large network tested in this work. When using more that 4 NPs for SNLP, the area required for the design is increase by a 1.5 factor. The limitation for the implementation of more SNLPs is the target FPGA available resources. The used hardware resources scale linearly with the number of neurons, and the performance decreases linearly but with a small slope compared with the hardware resources increasing. The performance improvement is very good (in the order of 7X to 9X), when using regular topologies (networks with the same number of neurons in each layer), and not too good (about 2X - 4X) when using an unbalanced number of neurons in the layers (see figures 6(a) and 6(b) for an example).

5 Conclusions and future work

A scalable and modular hardware core for SNNs has been proposed. The architecture is based in two phases: one preprocessing phase (GRF coding) and one processing phase for multilayer SNNs: the recall phase. For each one of these phases, a hardware core is proposed. For the coding phase, the GRFs is performed by a set of processors called GRFPs, a performance improvement of at least 4X is obtained. For the GRFs combined with the Recall Phase, a performance improvement of at least 3.5 X is obtained. The improvement to the first hardware core is reported in this work, while the second hardware core was left unmodified from the original version reported previously in [17]. The integration of both cores is fully documented in this work, as well as performance and resource utilization statistics. The proposed results shown an important improvement in performance with respect to SW-based implementation, and the modularization of the proposed architecture allows to implement the proposed architecture using larger FPGA devices.

As future work, the integration of both learning and recall blocks with the implemented modules is proposed. The reuse of hardware resources can improve the execution time, allowing to have a better network performance. If all the stages of the neural processing are implemented on-chip, the compactness of this core can allow the implementation of high-performance classifying systems. The testing of the proposed architecture with more standard datasets used in Machine Learning algorithms is required, and the integration of the core with more challenging neural networks applications, like speech recognition is needed.

References

1. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Transactions of the Society for Computer Simulation International **14**(4) (1997) 1659–1671
2. Gerstner, W., Kistler, W.: Spiking Neuron Models: An Introduction. Cambridge University Press, New York, NY, USA (2002)
3. Bohte, S.M., Kok, J.N., Poutre, H.L.: Spikeprop: Error-backpropagation for in multi-layer networks of spiking neurons. Neurocomputing **1-4**(48) (November 2002) 17–37
4. S. McKennoch, D.L., Bushnell, L.G.: Fast modifications of the spikeprop algorithm. IEEE World Congress on Computational Intelligence (WCCI) (July 2006)
5. Wu, Q.X., McGinnity, T.M., Maguire, L.P., Glackin, B.P., Belatreche, A.: Learning under weight constraints in networks of temporal encoding spiking neurons. Neurocomputing **69**(16-18) (2006) 1912–1922
6. Moore, S.C.: Back-Propagation in Spiking Neural Networks. Master's thesis, University of Bath, United Kingdom (2002)
7. Benjamin, S., Jan, V.C.: Backpropagation for population-temporal coded spiking neural networks. In: Proceedings of the 2006 International Joint Conference on Neural Networks, IEEE (1 2006) 3463–3470
8. Gerstner, W.: Populations of spiking neurons. Maass, W. and Bishop, C., editors, MIT-Press, Cambridge (1999)
9. Johnston, S., Prasad, G., Maguire, L.P., McGinnity, T.M.: Comparative investigation into classical and spiking neuron implementations on fpgas. In: ICANN (1). (2005) 269–274
10. Maass, W., Bishop, C.M., eds.: Pulsed neural networks. MIT Press, Cambridge, MA, USA (1999)
11. Jahnke, A., Schönauer, T., Roth, U., Mohraz, K., Klar, H.: Simulation of spiking neural networks on different hardware platforms. In: ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks, London, UK, Springer-Verlag (1997) 1187–1192
12. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
13. Nuño-Maganda, M., Arias-Estrada, M., Torres-Huitzil, C.: An efficient scalable parallel hardware architecture for multilayer spiking neural networks. In: SPLCONF 2007. (2007) 167–170
14. Nuño-Maganda, M., Arias-Estrada, M., Torres-Huitzil, C.: High performance hardware implementation of spikeprop learning: Potential and tradeoffs. In: ICFPT07, Kytakyushu, Japan (2007) 129–136
15. Benjamin, S., Jan, V.C.: Extending spikeprop. In Campenhout, J.V., ed.: Proceedings of the International Joint Conference on Neural Networks, Budapest (7 2004) 471–476
16. Nuño-Maganda, M., Arias-Estrada, M., Torres-Huitzil, C., Girau, B.: A population coding hardware architecture for spiking neural network applications. In: SPLCONF 2009, Saõ Carlos, Brazil (2009) 83–87
17. Nuño-Maganda, M., Arias-Estrada, M., Torres-Huitzil, C., Girau, B.: Hardware implementation of spiking neural network classifiers based on backpropagation-based learning algorithms. In: IJCNN09, Atlanta, U. S. (2009) 2294–2301