Hardware Radial Basis Function Neural Network Automatic Generation

Lucas Leiva*
INCA/INTIA, UNCPBA
Tandil, 7000, Argentina
and
Nelson Acosta
INCA/INTIA, UNCPBA
Tandil, 7000, Argentina

ABSTRACT

This paper presents a parallel architecture for a radial basis function (RBF) neural network used for pattern recognition. This architecture allows defining sub-networks which can be activated sequentially. It can be used as a fruitful classification mechanism in many application fields. Several implementations of the network on a Xilinx FPGA Virtex 4 - (xc4vsx25) are presented, with speed and area evaluation metrics. Some network improvements have been achieved by segmenting the critical path. The results expressed in terms of speed and area are satisfactory and have been applied to pattern recognition problems.

Keywords: RBF Neural Networks, FPGA, pattern recognition, architecture

1. INTRODUCTION

Artificial neural networks are used as a modeling technique that emulates the human brain. The main characteristic of a neuronal network is its ability to learn internal features by through data sets analysis. A neuronal network is made of a set of simple processing units, each one with a natural capability to store experimentally acquired knowledge together with the ability to use them readily.

The neural networks may be classified in terms of the systems they are intended to be used for [1]. The networks used in pattern recognition processes maybe readily used for classification systems. Classification procedures involve the derivation of a function dedicated to split data into categories, defined from a set of features. This function is activated by a neuron-classifier, trained to use different types of input data along with their categories.

A classification network links any input vector to a well established category, producing an output signal identifying this category. Moreover, the network can set some level of input acceptance within the selected category. This means that the output is not just binary.

The RBF networks [2] [3] [4] are commonly used as neural-classifiers. This type of networks is made of an input layer with branching nodes, a hidden layer and an output layer. Each node of the hidden layer has a special type of activation function located in central vector of the cluster: this function generates a most prominent response for those vectors closest to the center. The nodes of this layer are weighted. The output layer is responsible for producing the sum of the products obtained from the hidden layer weights (Fig. 1.).

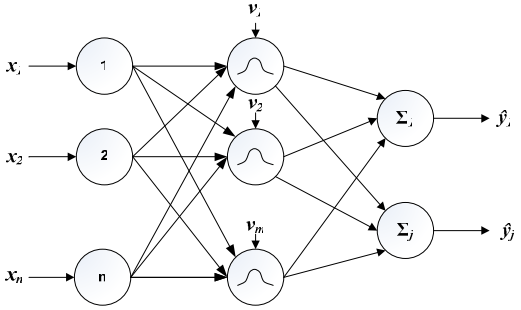

Fig. 1. RBF Neural Network

The hidden layer applies a function depending from the distance between the parametric charge vector (v_i) and the input vector. These functions represent a special class of functions whose main feature is a response decreasing with the increasing distance to the central point. A typical example is the Gaussian function.

Each radial basis function has an influence only in its receptive field, which is a small region of the featured space. The important regions of space are covered by a number *n* of functions.

Each radial basis function is attending to a small convex region called receptive field. A number of these functions cover a large space portion with their receptive fields. The output layer may thus associate some of them to regions with classes not linearly separable. Therefore, the number of RBF´s must be large enough to cover all subclasses that are linearly separable. Fig. 2 shows how the radial basis functions can cover any area of interest, no matter the shape of those regions.

Hardware implementation of neural techniques [5] has a significant number of advantages, mainly in the processing speed. For networks with large numbers of neurons and synapses, the conventional processors are not able to provide real time responses and training capacities, while parallel processing of multiple simple procedures achieves a large increase in speed.

*Sponsored by CONICET

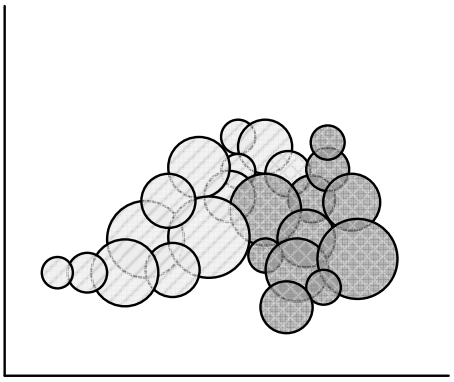

Fig. 2. RBF Contour curves in 2D space

Another advantage of implementing neural computing dedicated hardware is its ability to provide robust solutions for applications where it is not possible to install a PC. Such is the case for toys or autonomous robots for industrial uses or exploration. Numerous neural networks hardware implementation are available, such as the IBM ZISC78 device, Hitachi Digital Chip, Philips L-Neuro 1.0, Nestor Ni 100 [6] [7] [8], ZISC78[9], CM1K[10] devices, which implements RBF neural network, containing 78 neurons.

In this work, generic radial basis function architecture is presented, where certains parameters are instanciated in a VHDL template by a software tool.

The section 2 presents the hardware architecture of a RBF neural network implementation in a FPGA. We proposed an improvement to obtain a better performance through critical path segmentation. The section 3 analyzes the implementation in terms of area and speed. A several number of implementations were made, evaluating these parameters against the prototype size, the neurons number, and segmentation registers number. This section presents also a comparison against commercial neural networks devices (ZISC78, CM1K). Finally, the section 4 describes the conclusions of this work.

2. IMPLEMENTATION

A neural network is composed by a controller and a neuron set (Fig. 3). All the neurons are connected by a inter-neural communication bus. The controller (implementated as a state machine) coordinates the operations of the neuron set to perform a classiffication. Also, performs the feature vector reading and informs the operation status through the *rdy* signal.

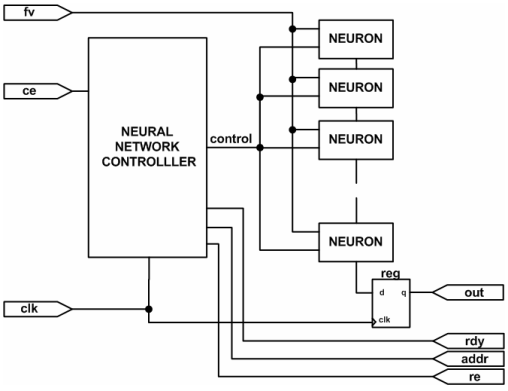

Fig. 3. Neural Network architecture

Each one of the neurons is associated with a category. Its operation basically begins with the occurrence of an input vector from which the neurons calculates the distance of this vector with the prototype stored in each of them. This prototype vector describes the "learning" of the neuron. The distance is compared with the influence field, which describes the receptive field. If the input vector lies in the neuron influence area, then the neuron is fired. Among the fired neurons, one is selected, with the shortest distance. The resulted category is then the one contained in this neuron.

Particular cases exist where the active (fired) neurons have different categories. In these cases, the network must indicate that the result is uncertain, as the system cannot determinate a confident response.

Given this behavioral description of a RBF neural network, this paper proposed an architecture which consists of a set of neurons and a controller responsible for coordinating the operations. Each neuron is composed of a prototype, a register which stores the influence field (*NAIF*), the category to which it belongs (*CAT*) and an additional register storing the distance (*DST*) between the feature vector and the stored prototype (Fig. 4). Both, the influence field and the category status are instantiated in VHDL code.

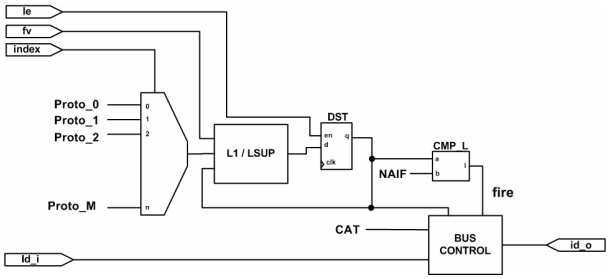

Fig. 4. Artificial neuron architecture

The function of distance is calculated allowing the following distance norms for the input feature vector (*FV*) and the prototype stored in the neuron (*P*):

$$L: DST = \Sigma \mid FV_i - P_i \mid \quad (1)$$

$$LSUP: DST = Max(\mid FV_i - P_i \mid) \quad (2)$$

The distance (*DST*) is updated when a load enable signal occurs. Then, one neuron is fired (*Fire = 1*) if its influence field (*NAIF*) is less than the calculated distance. Feature vectors are used because it is recommendable to eliminate unnecessary information.

The inter-neural communication bus is composed by four signals. The first of them, indicates the pattern identification *(Id)*, the next one indicates whenever a pattern has to be associated with two or more categories, making the classification uncertain (*Unc*). The other two signals consist of the smaller distance (*Dst*) together with the category (*Cat*) which involves this distance. Depending on the neuron state in front of an output enable signal, the bus controller behaves according to Table 1, where i_{n-1} is an input bus signal while i_n is an output bus signal, $i = \{Id, Unc, Cat, Dst\}$.

Table 1. Inter-neuron communication bus behavior

	Fire=0	Fire=1				
		Id_{n-1} = 0	Id_{n-1} = 1			
			CAT = Cat_{n-1}		CAT <> Cat_{n-1}	
			Dst_{n-1} < DST	Dst_{n-1} >= DST	Dst_{n-1} < DST	Dst_{n-1} >= DST
Id_n	Id_{n-1}	1	1	1	1	1
Unc_n	Unc_{n-1}	0	Unc_{n-1}	Unc_{n-1}	1	1
Cat_n	Cat_{n-1}	CAT	Cat_{n-1}	CAT	Cat_{n-1}	CAT
Dst_n	Dst_{n-1}	DST	Dst_{n-1}	DST	Dst_{n-1}	DST

The inter-neural controller bus behavior is synthesised as shown in the Fig. 5.

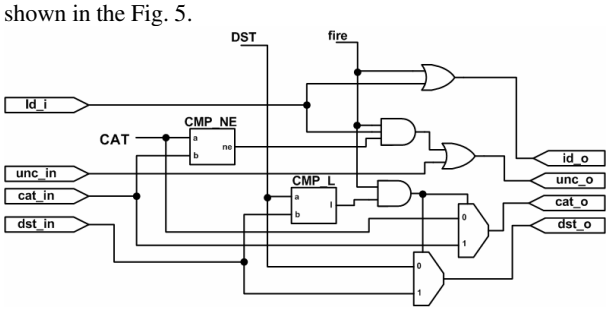

Fig. 5. Inter-neuron communication bus implementation

If the neuron is not fired, the input bus is mapped directly onto the output bus. However, if the neuron is fired the identification signal of the output bus will be 1 ($Id_n = 1$). On the other hand, if the input bus indicates an earlier identification ($Id_{n-1} = 1$) while the neuron is set active, an uncertainty arises about the equality between the category stored in the neuron and that of in the input bus. If they diverge, an uncertain signal will be activated on the output bus ($Unc_n = 1$), otherwise the uncertainty output will be that of the input ($Unc_n = Unc_{n-1}$). Actually, although no difference exists between both inputs, the bus contains the category of smallest distance found, up to the moment some uncertainty is carried out.

For the distance to the output bus, the controller simply verifies that the distance from the neuron (*DST*) is less than that of the input bus (Dst_n). If so, the distance of the output bus will be the one calculated by the neuron while the output category (Cat_n) wills the one that corresponds to the neuron. If the distance from the neuron is greater than the input bus distance, these signals will be mapped along with the category of the output bus.

For the first neuron in the network, identification signal and uncertainty signal must be 0 while the distance must be the greatest possible (Table 1).

The interconnection neuron buses ease the linkage of several neurons in cascade. Those buses impose a critical path in what concerns the system design, directly associated with the number of neurons in the network. It is well known that these critical paths can be resolved through the use of registers, in such a way to divide paths with greater latency into shorter ones. For this sake, one proposes to include intermediate registers, uniformly distributed on the inter-neuron communication bus.

The controller of Fig. 6 allows the existence of multiple subnets to be triggered in serial form; in this case, multiple neural networks can be contained in a common architecture, avoiding an extra cost in area. In figure 5, *net* is the active subnet, *nets_number* the number of subnets contained, *fv_count* corresponds to the position vector with entries belonging to the subnet currently analyzed, *fv_size(i)* is the size of the i^{th} subnet contained in the device, *NR* is the number of delay cycles (number of neuronal interconnection bus registers) needed to validate the output data of the inter- neuron communication bus and *nc_count* the actual delay cycle.

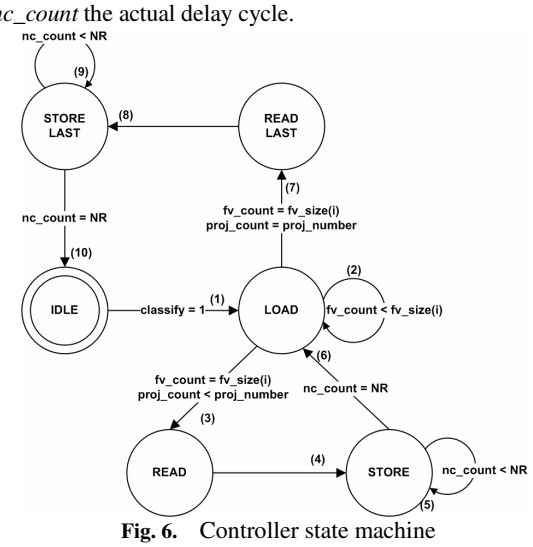

Fig. 6. Controller state machine

The controller operational sequence is presented in Table 2, where *(k)* is the set of operations of the transition *(k)* of the preceding state machine, $re_{j(i)}$ is the reading request for the j^{th} characteristic vector position, $le_{j(i)}$ represents the load enable signal for the j^{th} position of the characteristic vector, oe_j is the output enable signal, CAT_j indicates the category storage and DST_j the stored distance, for the j^{th} subnet. The table shows an instruction overlap reducing the number of the required cycles for the operation.

Table 2. Controller operational sequence

Cycle	Re	le	oe	DST	CAT	
0	$re0_{(0)}$					(1)
1	$re0_{(1)}$	$le0_{(0)}$				(2)
…	…	…				(2)
N	$re0_{(N)}$	$le0_{(N-1)}$				(2)
N+1	$re1_{(0)}$	$le0_{(N)}$				(3)
N+2	$re1_{(1)}$	$le1_{(0)}$	oe0			(4)
...	...	...	...			(5)
N+NR+2	$re1_{(NR+1)}$	$le1_{(NR)}$	oe0			(5)
N+NR+3	$re1_{(NR+2)}$	$le1_{(NR+1)}$		$DST_{(0)}$	$CAT_{(0)}$	(6)
N+NR+4	$re1_{(NR+3)}$	$le1_{(NR+2)}$				(2)
...	...	...				(2)
N+M	$re1_{(M)}$	$le1_{(M-1)}$				(2)
N+M+1		$le1_{(M)}$				(7)
N+M+2			oe1			(8)
…			...			(9)
N+M+NR+2			oe1			(9)
N+M+NR+3				$DST_{(1)}$	$CAT_{(1)}$	(10)

There are so many control signals (*oe, we, re*) as subnets into the device.

Assuming that a neural network contains multiple subnets, the number of cycles associated to its classification ($C_{Classify}$), is given by

$$C_{classify} = 3 + NR + \sum_{j=0}^{nets\_number} \dim(FV_j) \qquad (3)$$

Where *NR* is the inter neuron communication bus registers number and FV_j the characteristic vector corresponding to the j^{th} subnet.
As it can be observed, the number of cycles is independent of the classification results.
This implementation does not support embedded (on-the-chip) learning. To carry out this feature it would be necessary to adopt a strategy for receptive field centers selection [11], incorporating a number of empty (untrained) neurons. Reconfigurable logic could be a valuable approach.

3. EXPERIMENTAL RESULTS

Several networks were generated with different parameters (number of neurons, prototype vector size and inter neurons register number). All the networks were composed by a single subnet. The prototype vectors stored in the neurons were generated randomly.
These architectures were implemented on Xilinx Virtex4 (xc4vsx25-11-ff668) FPGA [12]. The synthesis has been achieved with XST (Xilinx Synthesis Technology) [13] while the physical implementation used Xilinx ISE (Integrated Software Environment) version 9.1.03i [14], using default options in both cases. In this presentation no mention is made about the input data stream. These data can be stored in LUTs, block RAMs or external memory.
The behavior of an architecture without inter neuronal registers has been analyzed with different prototype vector sizes (16, 32, 64) in a range of 6 to 30 neurons. This size relies on the application field. It has been observed that the influence of the vector size is practically zero both on area (Fig. 7) and latency (Fig. 8).

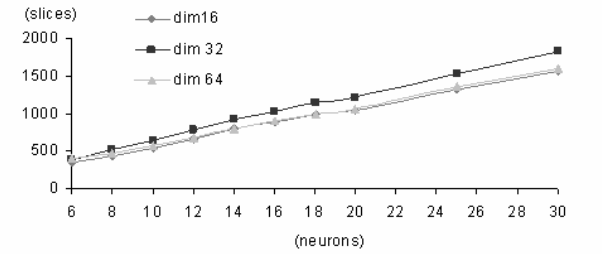

Fig. 7. Prototype vector size effect on area

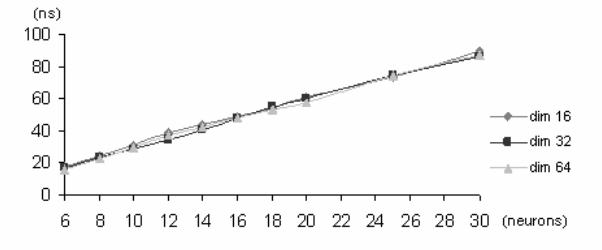

Fig. 8. Prototype vector size effect on latency

Figures 7 and 8 emphasize a growing proportion of both parameters with regard to the number of neurons in the network. According to the results it is possible to roughly estimate the area (number of slices) and the latency from the number of neurons (Eq.4, Eq. 5) by linear approximation:

$$\#\ Slices \cong \quad 59.53 \times Neurons + 54.57 \qquad (4)$$

$$Latency \cong \quad 2.92 \times Neurons + 2.019 \qquad (5)$$

The other important aspect to be looked at is the benefit of incorporating inter-neuron registers. A number of neural networks have been automatically generated with a number of neurons between 4 and 20, with prototype vector size fixed to 16, and inter-neuron registers number fixed to 0, 1 and 2. Fig. 9 and 10 display the results obtained in function of area and latency.

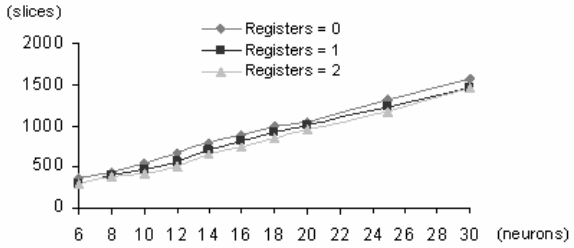

Fig. 9. Inter-neural registers effect on area

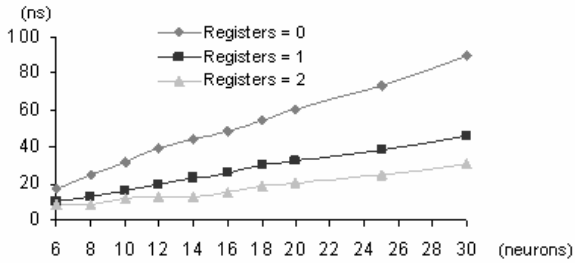

Fig. 10. Inter-neural registers effect on latency

These figures show that the number of intermediate registers has a significant impact in both area and maximum operating frequency. Incorporating one interneuron register an 89% acceleration is achieved; incorporating two registers, the acceleration reaches 169% figure with respect to an unsegmented architecture.
Nevertheless, the maximum number of characteristic vectors analyzed per second decreases with the size of the vector, as the number of cycles needed to achieve a classification directly depends upon the size (Fig. 11).

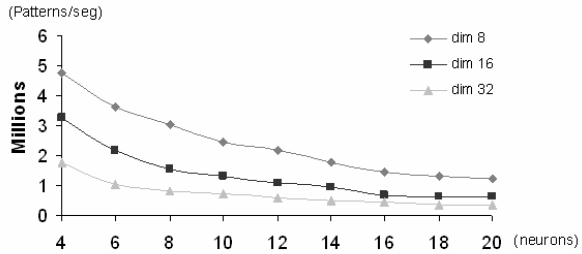

Fig. 11. Analysis of the dimension effect on the operating frequency

The ZISC78 devices needs at least *feature vector size+34*cycles to perform a classification, with 33 MHz clock frequency. This performance is constant with any number of neurons, being the maximum per chip of 78. The chip allows connecting multiple chips in chain to form greater neural networks. Comparing the performance obtained by the implementation and the ZISC78 devices,

the implementation has a greater throughput at least for a small number of neurons (Fig. 12).

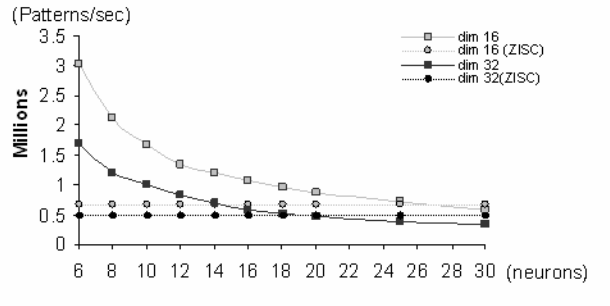

Fig. 12. ZISC78 vs. implementation throughput

On the other side, comparing the results with the CM1K device, is composed for 1024 neuron and it is also chainable. This chip operates with 27 MHz clock frequency, and requires 2 cycles to write the last component, 1 cycle for each other component, 18 cycles for read the category and 16 for the distance. The total number of cycles is *feature vector size + 34* cycles. In comparission with the proposed architecture, the performance of the implementation is better than the CM1K for a small number of neurons too (Fig. 13)

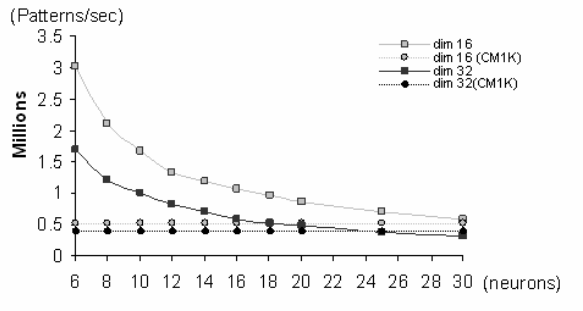

Fig. 13. CM1K vs. implementation throughput

Finally, a post-place&route implementation of a neural network is presented in the Fig. 14. This architecture contains 12 neurons with a protype size of 16 and 3 inter-neuron communication bus registers. The number of occupied slices is 470 out of 10240 (4%), allowing up to 50.725MHz clock frequency (2305698 patterns classified per second).

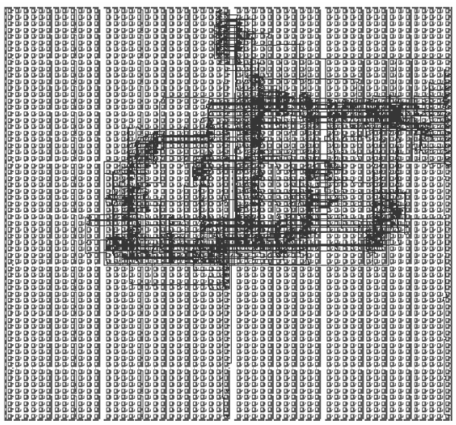

Fig. 14. Neural Network Placed & Routed on Virtex-4

This architecture has been tested for visual pattern recognition processes using different case of studies. Good results have been registered both in functionality and performance. Using a 12-trained-neuron network with a input feature vector sized 32, one succeeded to classify up to 170064 patterns per second [15] [16]. It must be pointed out that this application test also involves a characteristics extraction phase creating a composite profile, activated sequentially. The classification phase consumes a large portion of operating time. Better results can be obtained optimizing the feature extraction phase and incorporating a pipelining.

4. CONCLUSION

In this work a RBF Neural Network is presented, analyzing multiple parameters to evaluate the cost both in area and performance. A comparison was made between several implementations of this architecture and commercial chips (ZISC78 and CM1K). The results demonstrate that this architecture is well suitable for applications with small number of neurons demand. For example, considering a network composed by 20 neurons and a feature vector size of 16, the proposed architecture duplicates the performance obtained by CM1K device.

On the other hand, the inter-neuron registers allowed an operating frequency of 50 MHz, impossible to achieve otherwise.

Prospective future works will use this generated architecture in other areas of application (e.g. signals) and incorporate embedded learning.

5. ACKNOWLEDGEMENTS

The authors would like to thanks Géry Bioul for his support in the revision of this work.

6. REFERENCES

[1] Wolfram Research, Inc, “Neural Networks”, Wolfram Research, Inc, 2005.

[2] J. Moody and C. Darken, “Learning with localized receptive fields,” Proc. Connectionist Models Summer School, San Mateo, CA, 1988.

[3] I. Park and I. W. Sandberg, “Universal approximation using radial basis function networks,” Neural Computat., vol. 3, pp. 246–257, 1991.

[4] StatSoft Inc, “Neural Networks”, www.statsoftinc.com, 2003.

[5] N. Acosta, M. Tosini, “Custom Architectures for Fuzzy Logic and Neural Networks Controllers”, Journal of Computer Science & Technology, ISBN 1666-6064, pp. 9-20, 2002

[6] Valeriu Beiu, “Digital integrated circuit implementations”, Handbook of Neural Computation release 97/1, Publishing Ltd and Oxford University Press,1997.

[7] Clark S. Lindsey , Bruce Denby , & Thomas Lindblad, “Neural Network Hardware”, http://neuralnets.web.cern.ch/NeuralNets/nnwInHep.html, 1998.

[8] Yihua Liao, “Neural Networks in Hardware: A Survey”, Department of Computer Science, University of California, 2001.

[9] Silicon Recognition, "ZISC: Zero Instruction Set Computer", Version 4.2, Silicon Recognition, Inc., 2002

[10] Z. Uykan, "Clustering-Based Algorithms for Radial Basis Function and Sigmoid Perceptron Networks," Ph.D. dissertation, Control Eng. Lab.,Helsinki Univ. Technol., Helsinki, Finland, 2001.

[11] Cognimem, CogniMem_1K: Neural network chip for high performance pattern recognition, datasheet, Version 1.2.1, www.recognetics.com, 2008.

[12] Xilinx, Inc. Virtex-4 User Guide, UG070 (v2.6). www.xilinx.com, 2008.

[13] Xilinx, Inc. Xilinx Synthesis Technology (XST) User Guide. UG627 (v 11.1.0) www.xilinx.com, 2009.

[14] Xilinx, Inc. ISE 9.1.03i Documentation. www.xilinx.com, 2009.

[15] L. Leiva, "Herramienta para Diseño Automático de Arquitecturas Basadas en Redes Neuronales para Reconocimiento de Patrones Visuales", Graduation thesis, System Engineering, UNCPBA, 2006.

[16] L. Leiva, M. Vázquez, N. Acosta, G. Sutter, "Herramienta de Generación de Arquitecturas Hardware para Reconocimiento de Patrones en Imágenes", JCRA 2007: Jornadas de Computación Reconfigurable y Aplicaciones. September 12-14, 2007, Zaragoza, España.

Received: December 2010. Accepted: February 2011.