Approach to an FPGA embedded, autonomous object recognition system: run-time learning and adaptation

Rubén Salvador, Carlos Terleira, Félix Moreno and Teresa Riesgo
Centro de Electrónica Industrial, http://www.cei.upm.es
Departamento de Automática, Ingeniería Electrónica e Informática Industrial
Universidad Politécnica de Madrid.
felix.moreno@upm.es

ABSTRACT

Neural networks, widely used in pattern recognition, security applications and robot control have been chosen for the task of object recognition within this system. One of the main drawbacks of the implementation of traditional neural networks in reconfigurable hardware is the huge resource consuming demand. This is due not only to their intrinsic parallelism, but also to the traditional big networks designed. However, modern FPGA architectures are perfectly suited for this kind of massive parallel computational needs. Therefore, our proposal is the implementation of Tiny Neural Networks, TNN –self-coined term–, in reconfigurable architectures. One of most important features of TNNs is their learning ability. Therefore, what we show here is the attempt to rise the autonomy features of the system, triggering a new learning phase, at run-time, when necessary. In this way, autonomous adaptation of the system is achieved. The system performs shape identification by the interpretation of object singularities. This is achieved by interconnecting several specialized TNN that work cooperatively. In order to validate the research, the system has been implemented and configured as a perceptron-like TNN with backpropagation learning and applied to the recognition of shapes. Simulation results show that this architecture has significant performance benefits.

Keywords: hardware embedded intelligence, FPGA embedded system, neural networks, pattern recognition, autonomous system.

1. INTRODUCTION

One of the major problems in computer vision is to build systems with the ability to identify shapes in real world scenarios [1], [2], [3], [4], [5], [6], [7]. The target application of our work is the correct identification of road traffic signs in images taken by a car mounted camera, [8], [9]. This paper shows the on-going work towards low cost FPGA-based object detection and recognition systems. It deals with the run-time learning and adaptation capabilities implemented. It is part of a broader line of research investigating methods of scaling high level, intelligent, cognitive architectures, into limited resources embedded systems. An adapted Blackboard architecture, BB1/AIS [10], [11], is the architectural underlying framework on top of which the system is built.

The network chosen is a two layer perceptron-like TNN, with 30 and 3 neurons in each layer, and has been implemented on an Altera Cyclone II device. The system, based on a low-cost PAL standard video camera, has the required video preprocessing stages to adapt the pixel stream to the format required by the network. In order to reduce the size of the network to comply with the TNN definition, it is necessary to select a Region of Interest, RoI, within the whole image, to reduce the computational load necessary for the hardware implementation, since not all the information in the image is pertinent for the reference application. Besides, signal recognition is based on the identification of singularities – characteristics elements of shape.

The main contribution of this paper is the attempt to rise the autonomy features of the system, this is, making it able to adapt to unforeseen events or situations. For this reason, a mechanism for triggering a learning (re-training) phase if the system detects either a false positive or false negative has been implemented. The implementation of the learning algorithm –variable learning rate backpropagation–, has been done in the PowerPC processor available in a Xilinx Virtex-II Pro device. The reason to use a microprocessor instead of a custom hardware module is, besides the flexibility and reduced development times offered by a software approach, testing the capability, in terms of computational power,

of the processor to implement the learning task. As stated in the article, the backpropagation algorithm is not as fast as desired but the control-intensive part of the system is likely to be definitively implemented in an embedded processor.

When the learning phase is triggered, the data of the initial training of the network, done with 150 lists of three patterns, is retrieved and used as a seed for the new training process along with the input data (training pattern) that caused the false positive/negative. Therefore, since the weights and biases values of the network are already close to the solution, the training time is reduced. When this finishes, the new network parameters, as well as the training data, are stored to be used as initial seed for future learning processes, and the neural network reconfigured to reflect the new acquired knowledge.

There are several levels of parallelism in the implementation of the neural network recognition system that we are proposing: Parallelism among networks, among the layers of a network, among neurons and among connections. All of them are shown on the General Architecture of the system (Figure1).

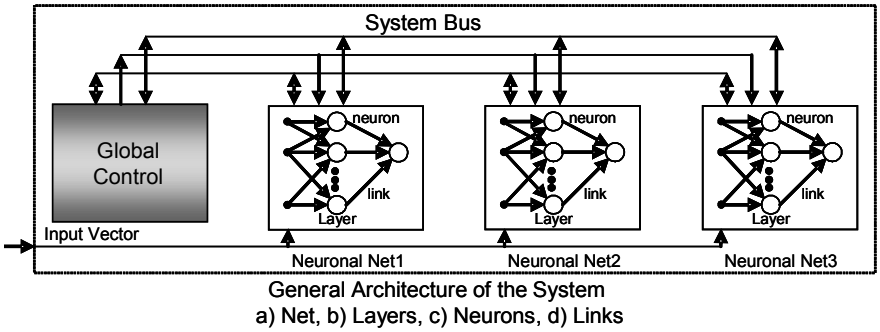


Figure 1: General architecture of the system.

The number of "synapses" and multipliers included in a fully interconnected network is proportional to the squared total number of neurons. The speed slows down due to the increase in the number of multipliers, and the chip size or chip area increases significantly, which becomes one of the critical points in TNN design. In order to solve this problem, the use of hardware multipliers seems to be an option to solve the chip size problem; as well as the design of neural networks without multipliers or reusable ones [13], [14].

Our work explores multiplier re-usability based on an internal bus structure. Taking into account the parallelism of the neural network model, it is possible to map the architecture on array processors, obtaining a linear growth in the number of multipliers. We have an ideal scenario for ANN implementation in embedded systems. Figure 2 shows a network interconnected by mean of an array processor model [15], [16], [17], [18].

Traffic signal and/or pedestrian recognition are two of the most relevant applications. These networks work cooperatively to obtain the classification of the image.

The main restriction comes with the complexity of the information contained in the image data, because it is sensible to changes of the environment. It is then necessary to have a recognition system that allows dynamic reconfiguration [1], [19].

An architecture that allows optimum usage of hardware resources was to be implemented due to the limitations in power and available area. The suggested system is formed by small Perceptron multilevel networks, and was implemented in an Altera Cyclone II FPGA.

2. CHARACTERISTICS OF THE SYSTEM ARCHITECTURE FOR THE FPGA BASED APPROACH

The requirements of recurrent learning processes can be satisfied by the reconfiguration and flexibility of FPGAs, [11], [20], [21]. Weight modification and architecture reconfiguration can be carried out at run time.

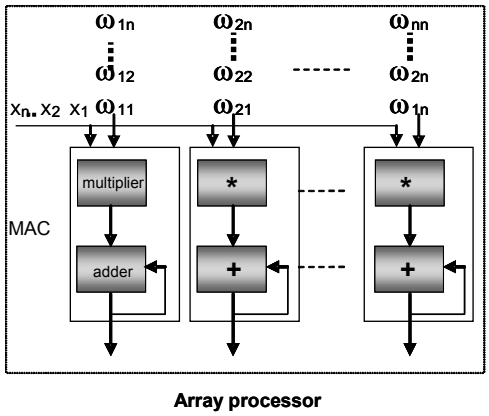


Figure 2: Network example.

When dealing with ANN implementation in hardware, the following considerations should be taken into account: frequency, precision, configuration issues, and parallelism. In order to improve general design characteristics, two units have been conceived: basic and control units.

Basic units (specialized neural networks) are in charge of signal processing and weight and bias data storage, including all the network data processing (multiplication of the weights by the inputs, accumulation and nonlinear function activation). The control units work on the basis of signal transmission including parallel processing and the algorithmic datapath control.

By considering those units, the proposed design (as shown in Figure 3) has an efficient architecture based on specialized neural networks in pattern recognition, implemented in an FPGA.

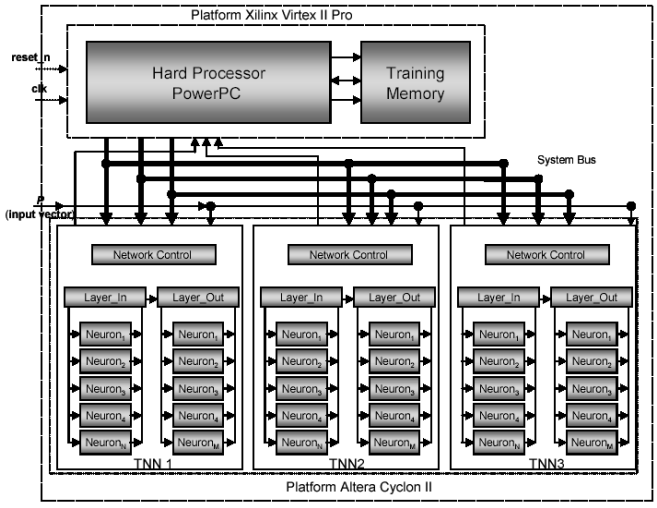


Figure 3: System Architecture

3. UNITS

For achieving the learning operation the algorithm is divided in three phases, known as: feed-forward, back-propagation and up-date [22], [23], [24]. In the feed-forward phase the input signals propagate through the network layer by layer, eventually producing some response at the output of the network. This response is compared with the desired (target) response, generating error signals that are propagated in backward direction through the network. In this backward phase of operation, the free parameters of the network are adjusted so as to minimize the sum of square error. Finally, weights

and biases are updated using the data obtained in the previous phase. The process is repeated as many times as necessary in order to have a trained network. Usually this process is made using general-purpose computers, and is known as off-line training. The three phases of algorithm are shown in Figure 4.

Since the proposed architecture is auto-reconfigurable during the execution time, separated modules where developed [12], [25].

For the system to carry out an on-line adaptation [3], [16], the same learning rules should be applied concurrently over a new pattern. When the network is reconfigured, the control unit executes the learning process concurrently, using the training patterns stored in the previos learning phase along with the new pattern to be recognized. This pattern is the one that triggered this learning phase..

When the learning process finishes, collected data is transmitted to the weights and bias network memories. This is carried out by the control unit, which makes them flow through the back-propagation level, directing this learning phase and making the FPGA get adapted.

Figure 5 shows the implementation of the different levels of the learning algorithm. The feed-forward and update modules corresponding to the basic unit were implemented directly in the same FPGA (Altera Cyclone II), and they are executed concurrently in a foreground Process.

To accomplish with the adaptation, an uncertainty computation module that is triggered after the feed-forward operation, determines if there is a new pattern and sends a request to the control unit to reconfigure the network. The backpropagation module corresponding to on line learning was implemented in the PowerPC processor XILINX (Virtex II), as a background process.

By means of a state machine, three modes of operation of the system were defined. In the Initialization mode, the system loads the initial values of the weights and biases, and begins the Classification mode. In this state, the network works in feed-forward. When, as explained above, it detects a new pattern to apply to a new learning phase, it changes to the reconfiguration mode.. When this mode is over, the update is carried out in other to begin go back again into the classification mode. The different modes of operation and the states machine will be explained later in this paper.

4. SPECIALIZED TINY NEURAL NETWORKS

Considering the problems of size and scalability, we propose a design based on the mathematical model of the neural networks, similar to the model shown in Figure 6(a).

As explained above, the synapse number is limited (network size) by the size of the internal memory of FPGA [26]. In addition, the network architecture (number of neurons and number of layers) is also limited by the hardware resources [27].

In order to avoid these difficulties, a Basic Processing Unit is suggested as the central component of the network. This unit is called the Knowledge Unit (KWU) and can be modified to feature one or several neurons. Each layer of the network is composed by several of these KWUs, obtaining different topologies according to the configuration of the internal registers of the system.

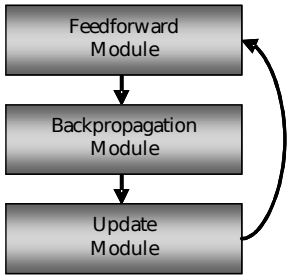


Figure 4: Sequential Algorithm for Learning Operation

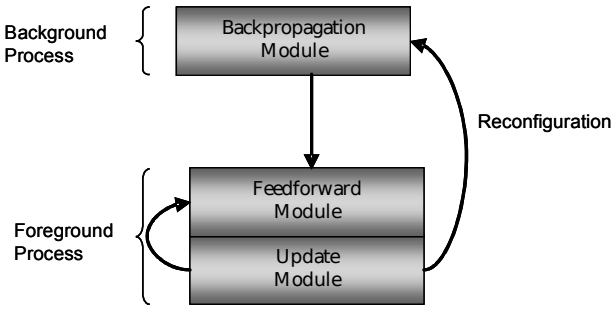


Figure 5: Segmentation of the algorithms for Learning Operation

Figure 6b shows the model of a Basic Learning Unit. The hardware architecture is obtained by directly mapping the high level, functional model of the perceptron neural network into its equivalent hardware representation. A data input vector (coming from the acquisition and pre-processing levels) and the external buses system (address, data and control) is used to interconnect the knowledge units with the general control of the system [8].

All hardware neurons (Basic Units), are formed by a MAC Unit (multiplier and accumulator), a Serial Unit (multiplexer), and the Non-linear Functions calculator, all of them interconnected by a parallel system bus as shown on Figure 6b.

The weight and bias memories have been implemented in the RAM modules embedded in the FPGA. These modules allow being accessed independently, so faster memory accesses are achieved thanks to this distributed memory scheme.

Having the Learning Unit, it is easy to have a neural network interconnecting two or more Basic Units; depending on the number of layers those neurons have (Modular and Scalable features of the Architecture). The interconnection is performed by an internal bus that transfers the data vectors of the previous stage. The data flow is controlled by a control unit through a protocol which indicates to the next layer on the network, the beginning and end of the information vector.

With the described architecture for the Basic and Control units, it is easy to build up a neural network by simply interconnecting two or more couples of these units, depending on the number of layers required. The inter-layer connection is performed by sticking together data and control buses from each layer. The data flow is controlled by the control unit explained above. This hardware architecture of the network layers confers the system modularity and scalability features. This may be helpful for future and more powerful versions, implemented on bigger FPGAs..

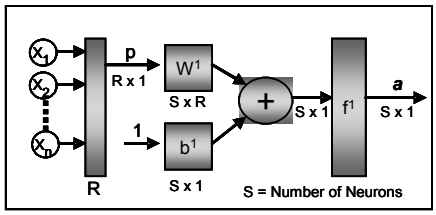
The modular design of the Control unit of the network layers avoids the need for global control and a completely modular and scalable system is obtained. The main activities carried out by the Control Unit are signal transmission and learning algorithm execution.

The state machine of the Control Unit has been carefully designed to improve the system performance. Figure 7 shows the three different branches that implement the functional behavior previously described.

The design of the system architecture allows for several networks to work (classify) in parallel. Besides, the training process could be executed in another TNN concurrently.

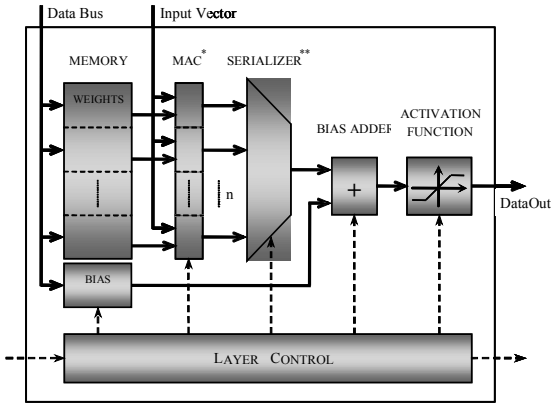
In order to maintain the processing speed of a TNN in hardware and its versatility in simulations, the reconfiguration of the neural network on its different hardware levels has to be possible (Reconfiguration features). Different researches have revealed that general purpose processors can be used in order to reprogram the neural network. We could also use FPGAs to modify the bus structure and the Basic Unit processing by means of the change in the configuration registers [21].

The proposed system implements an heterogeneous architecture, combining the features of general purpose processors (programming easiness and flexibility) with FPGAs (parallel processing and, therefore, performance).



a = f (Wp +b)

a) Mathematic Model
Layer of S Neurons, abbreviated notation



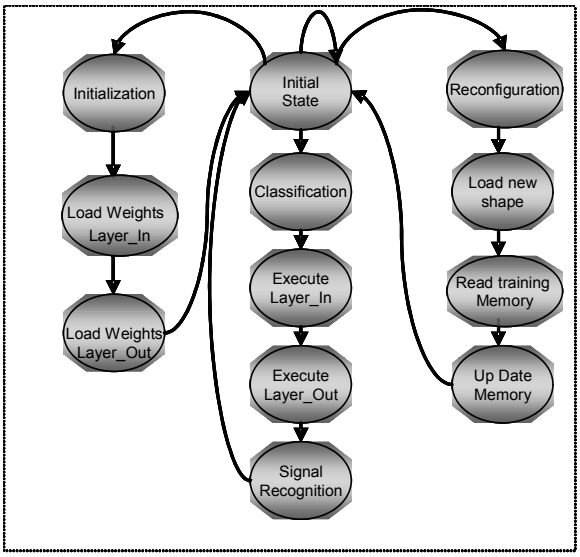
b) Electronic Model
Basic Processing Unit of ANN

Figure 6: TNN model

The specialized network design has an Uncertainty computation stage. Its main function is checking that the output data of the networks is somehow related to the output obtained with the training patterns, validating the recognition process. This happens when the uncertainty module reports a probability higher than 75%. On the other hand, if it is in a range between 50% and 75%, a reconfiguration request is asserted [28]. This probability levels have been empirically set. This way, the system is aware of what network was supposed to have identified the object, and therefore, train that network again with the new pattern (input data just analyzed). The stages of this uncertainty module are shown on Figure 8.

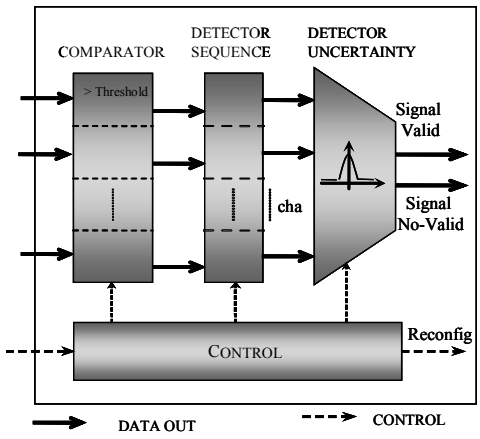
When this request is asserted, input data is acquired and attached to the appropriate training memory (initially filled with the patterns of the initial on-line training) as a new pattern for the following training process. Since the networks are trained to identify singularities, the data stored in memory are the pixels related to these singularities, not the entire image.

When this on-line training (actually, this training is an adaptation of the network, because it was initially trained, but is now re-trained to get adapted, taking advantage of the initial training) is finished, the hardware modules have been reconfigured: training memories (content and dimensions), as well as weight and bias memories have been updated, with new values obtained at the end of the process.



State Machine of the System
a) Initialization b) Classification c) Reconfiguration

Figure 7: System operation. State machine.



Uncertainty stage: Comparator,
Detector Sequence, Detector Uncertainty

Figure 8: Uncertainty module.

5. INTERCONNECTION OF THE SPECIALIZED TNN TO THE GLOBAL CONTROL SYSTEM

The designed system is highly parallel so it is able to execute several tasks at the same time. The networks in our system are also cooperative so they are able to solve complex issues through the contribution of each small network. As an example of the system application, the networks can be trained to identify characteristic elements of shape (singularities) such as right-angled corners, round segments and acute-angled corners. These singularities are used for the recognition of rectangular, circular and triangular shapes. Autonomous robots or intelligent systems for cars may use this kind of system [29].

The decision to have the communication of the global control system through a bus structure was taken after consideration of the efficiency level that we wanted to achieve. In this way the memory blocks share the same space on the system and can be accessed with a logic address, having as a result a distributed memory system on the networks with a centralized control. The addressing mode was considered to be the optimum model because it does not require a redundant memory for the networks, and only during the reconfiguration process may exist redundancy in the network memories that have to be reconfigured, achieving a faster convergence of the algorithm (Figure 9).

The learning memory (shown on Figure 3) is non volatile and has all the required training patterns for the training of each of the specialized networks.

The reconfiguration takes place right when a new image must be recognized. Therefore, the architecture has to be modified, and the new training patterns and targets added to the memory. When the training process ends, the memories are updated and the network connections have been already reconfigured so a new recognition process may begin.

According to the research, there are different ways of reconfiguration on a neural network. During the execution time, the number of neurons on the input layer can be modified or enough knowledge can be given to the network by changing the training memory content. Both of these methods explained lead to the recognition of the image.

6. RESULTS

The initial weight and bias data stored in the memory modules were obtained by an off-line learning process, using a backpropagation algorithm. The results obtained in the networks simulation are shown on Figures 10, 11 and 12.

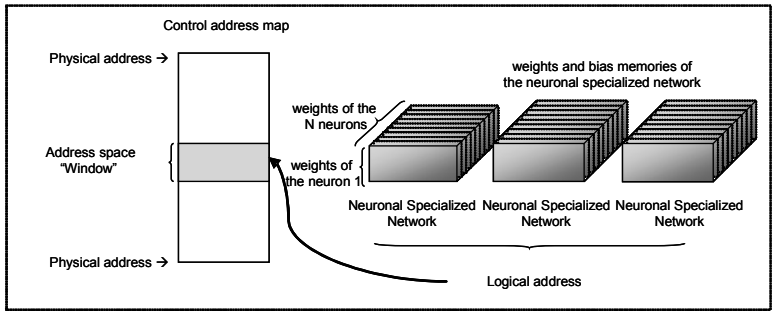


Figure 9: System memory map.

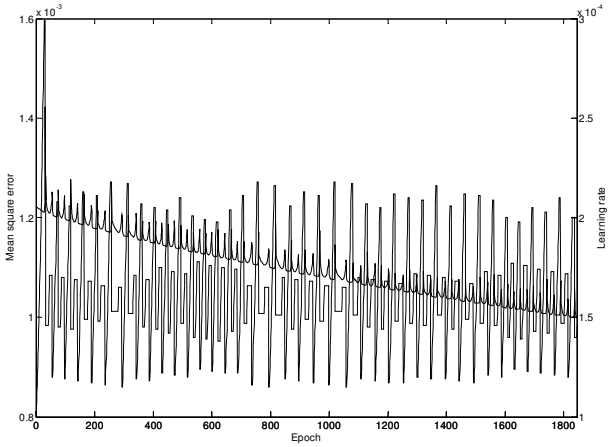


Figure 10: On-line reconfiguration process

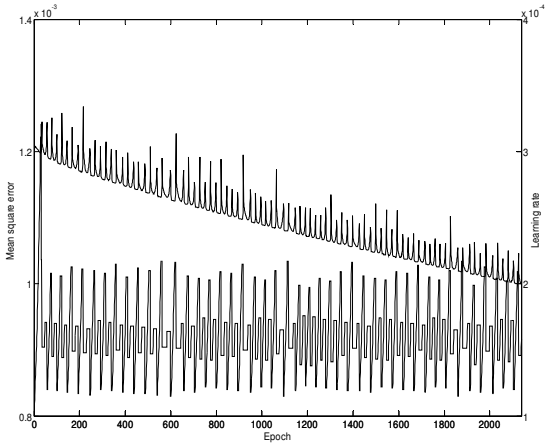
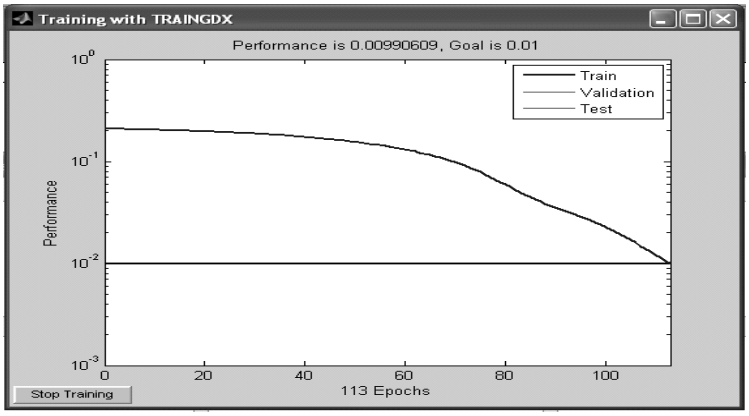
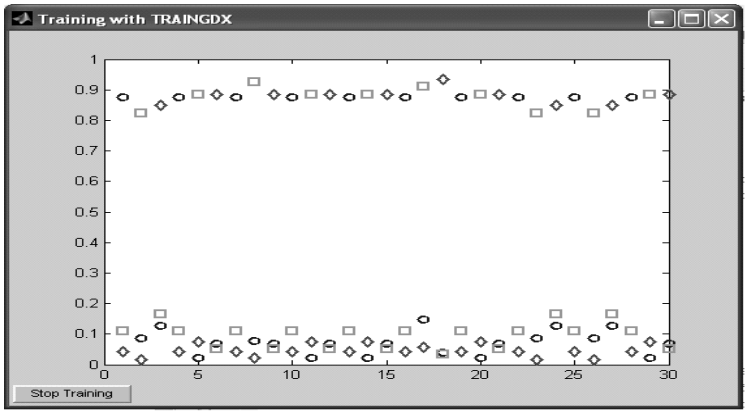


Figure 11: Another example of the on-line reconfiguration process



(a) Training



(b) Simulation

Figure 12: Training results.

In order to obtain the data of the memories of weights, 450 patterns of training with different characteristics have been used, taking into account the fact that all correspond to an image of the same class (rectangular, circular and triangular shapes)

The training method works in batch mode, which means that once all the entries were presented, the learning stage updates the weights and bias according to the decreasing moment of the gradient and an adaptive learning scale [23].

Some Maltab results are shown on figure 10, where the graph shows the stages used for the algorithm to converge with the targets of the parameters [30].

Whenever a signal for reconfiguration is produced, coming from the Uncertainty module, the on-line reconfiguration process takes place. Figure 11 and figure 12 show two examples of the on-line reconfiguration process (means square error and learning rate versus epoch). Both show the number of necessary iterations of re-training for one of the networks specialized.

As a design premise we have always had in mind a design for reuse methodology. Therefore, a big effort has been made to specify as many generic hardware modules as possible. For this reason, the architecture and VHDL description of the TNN has been improved so that later versions, apart from the basic functionality mentioned above, make possible their building N-layer, m-output perceptrons in the easiest and most-automated way possible. These features have been incorporated so that we shall be able, in the future, to test the system architecture on larger FPGAs.

Preliminary synthesis (no synthesis effort or optimizations directed to the synthesizer) results for Altera CYCLONE EP1C20F400C6 and CYCLONE-II EP2C35F672C6 devices have been obtained with the Altera Quartus II (v. 6.0) software package. The proposed architecture (Q8.16) fits in one CYCLONE device, but remaining resources, mainly memory, are a bit scarce. Therefore, the system has also been implemented in the CYCLONE-II device. Functional and post-fitting simulations with Mentor Graphics ModelSim simulation environment show how the real-time restrictions imposed on the system and the functional specifications are met.

Fitting results are shown on Figure 13(a) for the whole Recognition System (including the TNN, Figure 13(b), and an image preprocessing stage). The implemented TNN has 30 neurons in the input layer, and 3 in the output layer. Fitting details for the most important blocks of the architecture are also shown.

Fig. 13 shows the available resources for the Learning Algorithm implementation on a Xilinx Virtex-II PowerPC. It was coded in C language (400 code lines). Some interesting data is:

- Learning Rate α = 0.1: Reconf. Time = 0.3sec., with 12 iterations.
- Learning Rate α = 0.01. Reconf. Time = 0.8 sec., with 58 iterations.

Recognition System	
Logic Elements (LEs)	4,437 / 33,216 (13 %)
Total Memory Bits	72,416 / 483,840 (15 %)
M4Ks	47 / 105 (45 %)
Frequency	119.32 MHz

(a)

TNN		
	BKU	RoI Extr.+ RoI Buf.
LEs	2924 / 33,216 (9%)	275 / 33,216
Total Mem. Bits	24480/483,840(5%)	2880 /483,840
M4Ks	34 / 105 (33 %)	1 / 105
Frequency	119.32 MHz	

(b)

Figure 13: Recognition system implementation results.

Created by Base System Builder Wizard for Xilinx EDK 8.2 Build EDK_Im.14	
Target Board	Xilinx XUP Virtex-II Pro Development System Rev C
Family	virtex2p
Device	xc2vp30
Package	ff896
Speed Grade	-7
Processor	PPC 405
Processor clock frequency	300.000000 MHz
Bus clock frequency	100.000000 MHz
Debug interface	FPGA JTAG
Data Cache	16 KB
Instruction Cache	16 KB
On Chip Memory	208 KB
Total Off Chip Memory	512 MB
	- DDR_SDRAM_64Mx64 Dual Rank = 256 MB
	- DDR_512MB_64Mx64_rank2_row13_col10_cl2_5 = 256 MB

Figure 14: Learning Algorithm Implementation.

7. CONCLUSIONS

A new hardware architecture system for neural network based on specialized Tiny Neural Networks (TNN) for image recognition has been proposed and designed . One of the most important features of Tiny Neural Networks (TNN) is their on-line learning ability. These TNN are also cooperative in order to solve complex recognition problems. As an example of the system application, TNN can be trained to identify special points on an image. These points are characteristic elements of shape (singularities) such as right-angled corners, round segments and acute-angled corners. The main contribution of this paper is the attempt to rise the autonomy features of the system, this is, making it able to adapt to unforeseen events or situations. For this reason, a mechanism for triggering a learning (re-training) phase if the system detects either a false positive or false negative has been implemented. The implementation of the learning algorithm –variable learning rate backpropagation–, has been done in the PowerPC processor available in a Xilinx Virtex-II Pro device. As stated in the article, the backpropagation algorithm is not as fast as desired but the control-intensive part of the system is likely to be definitively implemented in an embedded processor.

ACKNOWLEDGMENT

This development is carried out in the ASISTENTUR project (Advanced Driver Assistance System for Urban Environments, TRA2004-07441-C03-03/AUT,) with the support of the Spanish Ministry of Science, under the National R&D Plan.

REFERENCES

[1] A. de la Escalera, L. E. Moreno, M. A. Salichs, J. M. Armingol, Road traffic sign detection and classification, IEEE Transactions on Industrial Electronics, 1997, Vol. 44, pp 848-859.

[2] A. de la Escalera, L. Moreno, E. A. Puente, M. A. Salichs, Neural traffic sign recognition for autonomous vehicles, IEEE Int. Conf. Industrial Electronics, Control and Instrumentation, 1994, Vol. 2, pp 841-846.

[3] T. Theocharides, G. Link, N. Vijaykrishnan, M. J. Invin, V. Srikantarn, A generic reconfigurable neural network architecture as a network on chip, Proceedings, IEEE International SOC conference, 2004, pp 191-194

[4] S. Estable, J. Schick, F. Stein, R. Janssen, R. Ott, W. Ritter, Y. J. Zheng, A real time traffic sign recognition system, Proceedings of the Intelligent Vehicles Symposium, 1994, pp 213-218.

[5] J. Torresen, J. W. Bakke, L. Sekanina. Efficient recognition of speed limit signs, Proceedings. IEEE International Conference on Intelligent Transportation Systems, 2004, pp 652-656.

[6] V. Moreno, A. Ledezma, A. Sanchis, A static images based-system for traffic signs detection, Proceedings of the IASTED international conference on artificial intelligence and applications, 2006, pp 445-450.

[7] G. Adorni, V. D'Andrea, G. Destri, M. Mordonini, Shape searching in real word images: a cnn based approach, Proceedings Fourth IEEE International Workshop on Cellular neural networks and their applications, 1996, pp 213-218.
[8] J. Alarcón, R. Salvador, F. Moreno, I. López, A new real-time hardware architecture for road line tracking using a particle filter, Proceedings of 32nd annual Conference of the IEEE Industrial Electronics Society, IECON'06, Paris 2006, pp 736-741.
[9] I. López, R. Salvador, J. Alarcón, F. Moreno, Architectural design for a low cost fpga-based traffic signal detection system in vehicles, volume 65900, pages 65900M. SPIE, 2007, Gran Canaria. Spain
[10] BB1 v3.2 Manual. Michael Wolverton, Lee Brownston. Draft, Report No. KSL 94-XX, March 1994. Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford, California 94305.
[11] Hayes-Roth, B.; Pfleger, K.; Lalanda, P.; Morignot, P.; Balabanovic, M. A Domain-Specific Software Architecture for Adaptive Intelligent Systems. IEEE Transactions on Software Engineering, Volume 21, Issue 4, April 1995 Page(s):288 - 301. Digital Object Identifier 10.1109/32.385968.
[12] S. Bridges, M. Figueroa, D. Hsu, C. Diorio, A reconfigurable vlsi learning array, Proceedings of the 31st European Solid-State Circuit Conference, 2005, pp 117-120.
[13] M. A. Figueiredo, C. Gloster, Implementation of a probabilistic neural network for multi-spectral image classification on an fpga based custom computing machine, Proceedings. Vth Brazilian Symposium on Neural Networks, 1998, pp 174-178.
[14] H. Hikawa, Implementation of simplified multilayer neural networks with on-chip learning, Proceedings IEEE International Conferences on Neural Networks, 1995, Vol. 4, pp 1633-1637.
[15] S. B. Yun, Y. J. Kim, S. S. Dong, C. H. Lee, Hardware implementation of neural network with expandible and reconfigurable architecture, Proceedings, IEEE Int. Conf. on neural information, 2002, Vol. 2, pp 970-975.
[16] B. Pino, F. J. Pelayo, J. Ortega, A. Prieto, Design and evaluation of a reconfigurable digital architecture for self-organizing maps, Proceedings, Int. Conf. Microelectronics for neural, fuzzy and bio-inpired system, 1999, pp 395-402.
[17] D. Hammerstrom, A vlsi architecture for high-performance, low cost, on-chip learning, IJCNN International Conference on Neural Network, 1990, Vol. 2, pp 537-544.
[18] S. Vitabile, A. Gentile, G. B. Dammone, F. Sorbello, Multi-layer perceptron mapping on a simd architecture, Proceedings of the 12th IEEE workshop on Neural Networks for signal processing, 2003, pp 667-675.
[19] Y. E. Krasteva, E. de la Torre, T. Riesgo, Partial reconfiguration for core relocation and flexible communications, Proceedings of Reconfigurable Communication-centric SoC, pages 91-97, Montpellier 2006.
[20] J. L. Beuchat, J. O. Haenni, E. Sanchez, Hardware reconfigurable neural networks, IPPS, SPDP worshops, 1998, pp 91-98, url = citeseer.ist.psu.edu/beuchat98hardware.html
[21] J. A. Starzyk, Z. Zhen, L. Tsun-Ho, Self-organizing learning array, IEEE Transactions on Neural Networks, 2005, Vol. 16, pp 355-363.
[22] J. G. Eldredge, B.L. Hutchings, Density enhancement of a neural network using fpgas and run-time reconfiguration, Proceedings, IEEE workshop on fpgas for custom machine, 1994, Vol 10-13, pp 180-188.
[23] Martin. T. Hagan, Howard. B. Demuth, Mark. Beale, Neural Network Design, book: Thomson Learning, United States of America, 1996.
[24] B. Kröse, P. Van der Smagt, An introduction to Neural Networks, eighth edition, The Universidad of Amsterdam, 1996.
[25] M. A. Hannan Bin Azhar, K. R. Dimond, Design of an fpga based adaptive neural controller for intelligent robot navigation, Proceedings, IEEE Euromicro symposium on digital system design, 2002, Vol. 2, pp 283-290.
[26] Y. Taright, M. Hubin, FPGA implementation of a multilayer perceptron neural network using vhdl, Proceedings Fourth International Conference on Signal Processing, 1998, Vol. 2, pp 1311-1314.
[27] J. J. Blake, L. P. Maguire, T. M. McGinnity, L. J. McDaid, Using Xilinx FPGAs to implement neural networks and fuzzy systems, IEE Colloquium on Neural and Fuzzy Systems: Design hardware and applications, Digest No. 1997/133, pp 1/1 – 1/4.
[28] A. Perez-Uribe, E. Sanchez, Implementation of neural constructivism with programmable hardware, Proceedings of the International Symposium on Neuro – Fuzzy systems, 1996, pp 47-54.
[29] L. Priese, R. Lakmann, V. Rehrmann. Ideogram identification in a realtime traffic sign recognition system, Proceedings. IEEE Intelligent Vehicles 1995, Vol. 25-26, pp 310-314.
[30] MATLAB. The Language of Technical Computing. Version 7.4.0.287 (R2007a).