FPGA Implementation of an Image Recognition System based on Tiny Neural Networks and on-line Reconfiguration

Félix Moreno, Jaime Alarcón, Rubén Salvador 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 are widely used in pattern recognition, security applications and robot control. We propose a hardware architecture system; using Tiny Neural Networks (TNN) specialized in image recognition. The generic TNN architecture allows expandability by means of mapping several Basic units (layers) and dynamic reconfiguration; depending on the application specific demands. One of the most important features of Tiny Neural Networks (TNN) is their learning ability. Weight modification and architecture reconfiguration can be carried out at run time. Our system performs shape identification by the interpretation of their singularities. This is achieved by interconnecting several specialized TNN. The results of several tests, in different conditions are reported in the paper. The system detects accurately a test shape in almost all the experiments performed. The paper also contains a detailed description of the system architecture and the processing steps.**

In order to validate the research, the system has been implemented and was configured as a perceptron network with backpropagation learning and applied to the recognition of shapes. Simulation results show that this architecture has significant performance benefits.

I. INTRODUCTION

One of the major problems in computer vision is to build systems with the ability to identify shapes in real world scenario [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]. The basic technique used for this in most applications is to compare each portion of an image with a set of known models (pattern-matching). On the other hand, the approach taken in our work is to use specialized Tiny Neural Networks (TNN), making it possible to use a massively parallel architecture efficiently. A most important feature of Artificial Neural Networks (ANN) is their learning ability. Size and real-time considerations show that on-chip learning is necessary for a large range of applications [3].

There are several levels of parallelism in 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 (figure.1).

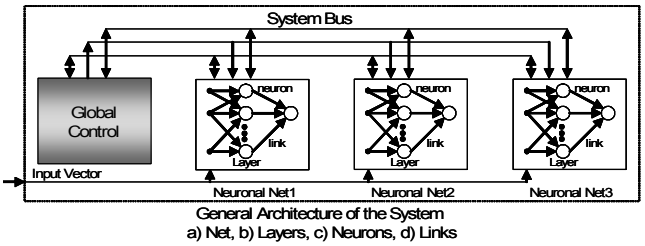


Fig. 1. System Architecture

In contrast to software implementation of ANN, hardware implementation provides a high level of parallelism. This allows us to make several computations concurrently in order to have a higher processing speed [10], [11], [12]. In addition, the hardware implementation is highly portable since it has minimum requirements in area and power consumption, and provides standalone.

We can classify the ANN hardware implementation in two main categories: that based on microprocessors by using Digital Signal Processors (DSP) or general purpose processors, and that using an Application Specific Integrated Circuit (ASIC) or Field Programmable Gate Array (FPGA).

The first one based on microprocessors, is more flexible and relatively easy to implement. However, when the network becomes larger (for example, in a fully connected network), it is not the best option: general purpose computers are required, and the usage and application depend on power and area characteristics available.

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 ANN design. In order to solve this problem, the use of hardware multipliers seems to be an option to

978-1-4244-1766-7/08/$25.00 ©2008 IEEE

resolve 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].

The main objective of this research is the design of a reconfigurable, efficient, low cost architecture for shape recognition. Robust methods for the analysis of images, and the implementation of a system based on specialized TNN have been developed for shape recognition by means of the analysis of some characteristics of the image (singularities). Traffic signal recognition 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 they are sensible to changes of the environment. It is then necessary to have a recognition system that allows dynamic reconfiguration [1], [19].

It is necessary to develop an architecture that allows optimum usage of hardware resources, 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.

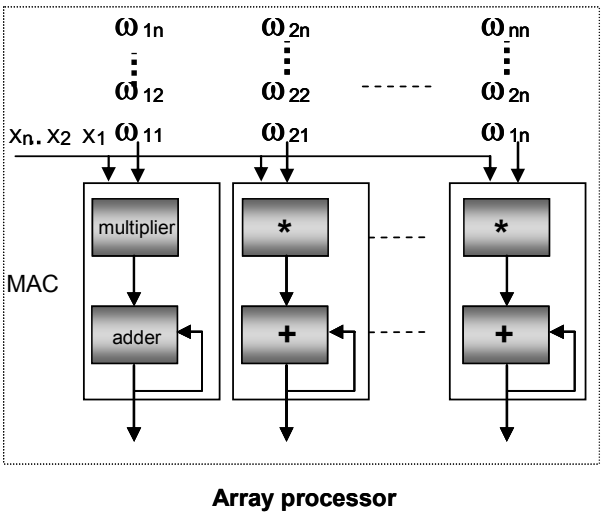


Fig. 2. Network Example

II. System Architecture Characteristics of 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 during run time.

When talking about ANN implementation, the following considerations should be taken into account: frequency, precision, configuration issues, and ANN parallelism. In order to improve general design characteristics, there are two units: basic and control units.

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

By considering those units, the proposed design (as shown in figure 3) has an efficient architecture based on specialized neural networks by recognition, to be implemented in FPGAs.

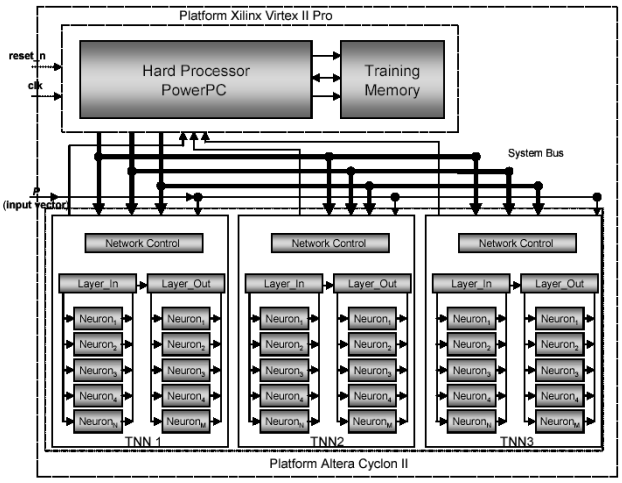


Fig. 3. System Architecture

III. 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.

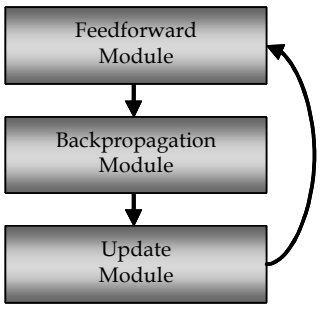


Fig. 4. Sequential Algorithm for Learning Operation

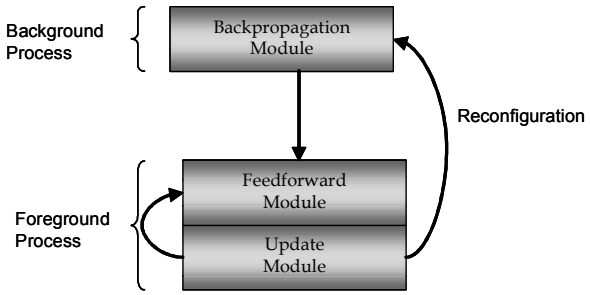


Fig. 5. Algorithms Segmentation for Learning Operation

Since the proposed architecture is auto-reconfigurable during the execution time, separated modules where developed [11], [12], [25].

So that the system carries out an on-line reconfiguration [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 along with the new pattern to be recognized.

When the learning processes finishes, collected data are transmitted to the weights and bias network memories, by means of the control unit and pass through the backpropagation level, which checks the reconfiguration and learning of it.

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.

At that moment, the uncertainty module 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), background process. Each module of the algorithm follows the general architecture of the system proposed showed in figure 3, consisting of a global control and several specialized neural networks.

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 mode, the network works in feed-forward, and when it detects that a new pattern has arrived it changes to the Reconfiguration mode. When this mode finished, the update is carried out in other to begin again with the classification mode. The different modes of operation and the states machine will be explained later in this paper.

IV. 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 configure a neuron or a number of them in order to create one of the layers of the network, obtaining different topologies according to the programming of the internal registers of the system.

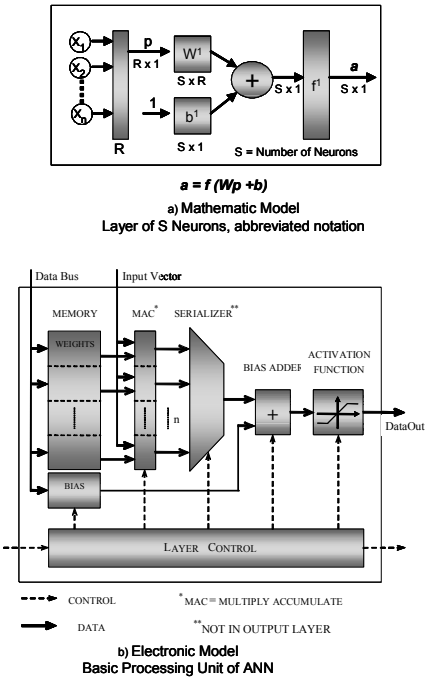


Fig. 6. TNN Model

Figure 6b shows the model of a Basic Learning Unit. The hardware architecture is obtained by mapping the high level algorithmic model of the Perceptron neural network into the

equivalent module hardware. 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].

According to our research it is necessary to know the degree of parallelism (taking into account the hardware resources) of the algorithm, in order to trade off the development and hardware resources consumption when implemented. With the suggested model, the Basic Learning Unit architecture has an almost complete parallel functionality, providing the best development with the minimum resources.

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

MAC Units are connected through the internal data bus to their weight memories and to the series of input data (input vector). Let us suppose that we have an input layer of N neurons. By means of this architecture it is possible to carry out N operations in parallel with serial input data because of the simultaneous access of the memories, through the internal structure of bus. Therefore, 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.

The design of the Basic Unit should include a level in which output data are obtained (output vector) in order to balance cost and development. This internal output contains the results of the first layer neurons and it is used as an input vector on the network's hidden or the output layers.

All of the MAC units makes parallel calculations ending up into an architecture with a high hardware resources consumption, so resources are optimized by an adder and a block which activates the non-lineal function used into the Basic Unit design; this way, the Learning Unit architecture has an input vector and an output vector for the information transfer (feed-forward), through the different network layers, being able to implement several neural networks, Perceptron Multilayer (PM), [18].

As a special case, when talking about a Perceptron Multilayer network and due to the little number of neurons on the exit layer, it is not necessary to use a Serial Unit because cost and development trade off does not have a negative impact on the architecture. There is an adder and an activation function by output neuron; the bias memory has been implemented with registers in the same adder, reducing the RAM cost and achieving resource optimization. We can see the architecture of an exit layer unit on figure 7.

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.

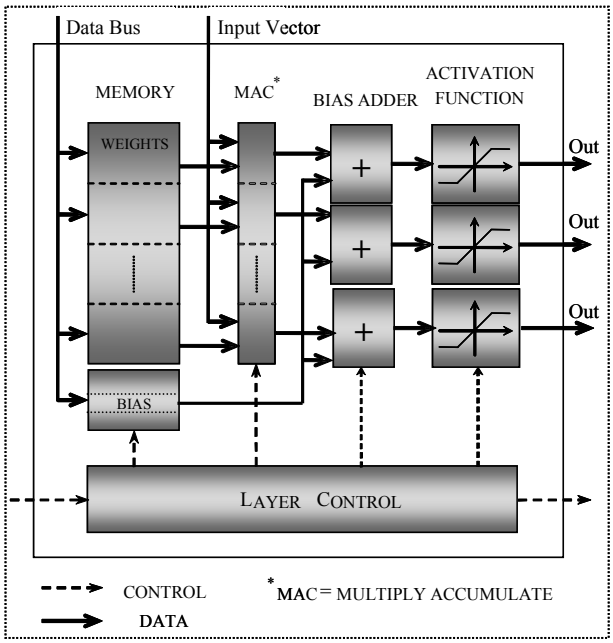


Fig. 7 Basic Unit Output Layer

With the Basic and Control units designed, a new layer of the neural network, which we will call the Learning Generic Unit can be designed. This is the basic module in the network and gives the modular characteristic to the system, which enables the Perceptron networks with several neurons and layers to be implemented.

These units are also the basic modules on the huge architectures on FPGAs platforms, having a growing system on internal networks and the whole one. A growing and modular system is achieved when the modular control unit is designed.

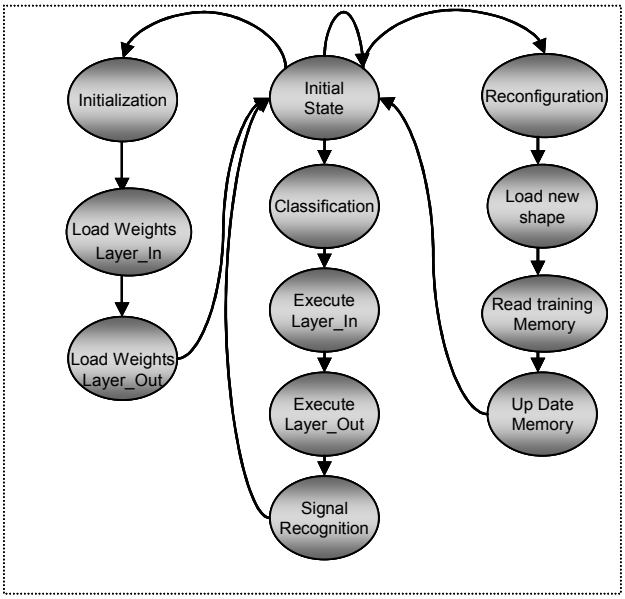
The modular design of the Control unit of the network layers avoids the need for global control and a completely modular and scalable system is obtained. The main activities carried out by the Control Unit are signal transmission and learning algorithm execution.

The state machine of the Control Unit has been designed to improve the system performance. This state machine has been carefully created and has three different ways of implementing the algorithm (figure 8) as explained previously.

The characteristics of the design allow that several networks execute the classification process in parallel, and meanwhile concurrently the training process could be executed in another one TNN.

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 possessing by means of the change in the configuration registers [21].



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

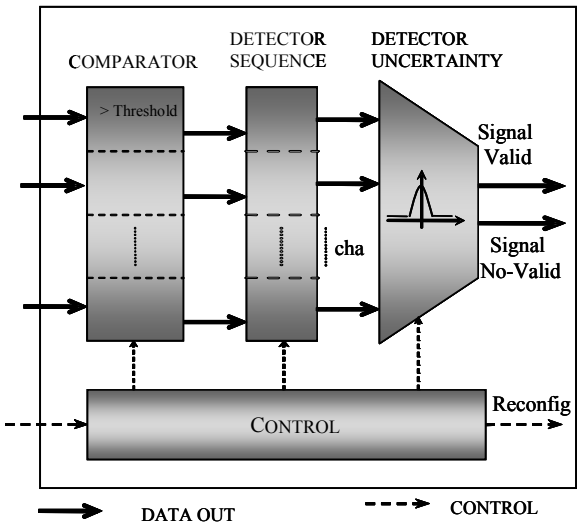
Fig. 8 System Operation

The characteristics of the design allow that several networks execute the classification process in parallel, and meanwhile concurrently the training process could be executed in another one TNN.

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.

The way in which general purpose processors are used is better although it is not completely successful with regard a the processing speed and required areas; in any case, the reconfigurable hardware networks are limited when programming but they have more processing speed, they use a smaller area and they can be included in an integrated circuit (system on-chip). Due to the characteristics of the suggested system, it can be considerate to be a heterogeneous architecture, combining the activities from the general purpose processors (programming availability) and those of the FPGAs (parallel processing and execution speed).

The specialized network design has an Uncertainty stage (module). The basic function of the module is to determine if the output data sequence corresponds to the training pattern or if it is similar to it, generating a valid signal for the recognition. On the other hand, if the Uncertainty module detects similar points on a probability range between 50% and 75%, a signal for reconfiguration is produced [28]. Uncertainty stages are visible on Figure 9.



Uncertainty stage: Comparator,
Detector Sequence, Detector Uncertainty

Fig. 9 Uncertainty Module

When the reconfiguration takes place, the global control system executes several processes concurrently: input vector acquisition, data attachment to training memory, new training vectors (target) and on-line training execution (including the new learning pattern).

When training is finished, the hardware stages have been reconfigured: training memories (content and dimensions), and weight and bias memories have been updated, with new values obtained at the end of the process.

V. Interconnection of the Specialized TNN to the Global Control System

The system designed is highly parallel and able to execute several tasks at the same time. The networks in our system are also cooperative in order to solve complex issues through small networks. As an example of the system application, the networks 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. These singularities are used for the recognition of rectangular, circular and triangular shapes. Autonomous robots or intelligent systems for cars 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 system of the memories 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 can there exists a redundancy in the network memories that have to be reconfigured, achieving a more rapid convergence of the algorithm. See figure 10, for the networks interconnection to the global control.

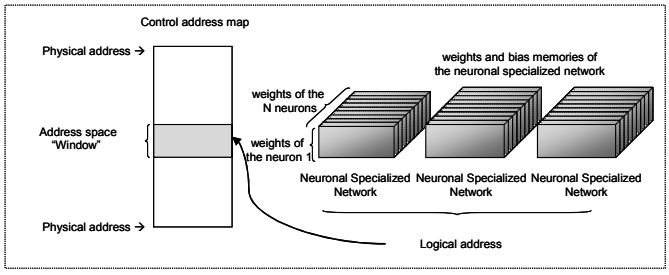


Fig. 10 System Memory Map

Figure 3 shows the general diagram of the system, including the learning memory. That memory is a non volatile one and has all the required training patterns for the specialized networks training.

The reconfiguration takes place at the moment at which a new image must be recognized. Therefore, the architecture has to be modified, and the new training patterns and the targets added to the memory. When the training process ends, the memories are updated and the network has the recognition connections.

According to the research, there are different forms of reconfiguration on a neural network. During the execution time, the number of neurons on the input layer can be modified or we can give enough knowledge to the network by changing the training memory content. The either of both methods leads us to the image recognition.

Depending on available hardware resources and the applications of the system, a dynamic reconfiguration is possible when the image is part of the same group. Specialized cooperative networks, where installed for reconfiguration to be held in the control section, acquire more knowledge as new images are recognized.

VI. RESULTS

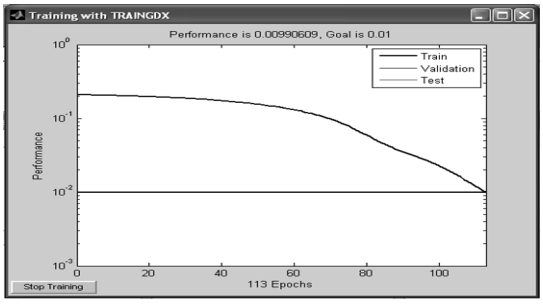
The weight and bias data stored in the memory modules were obtained by the off-line learning, using a backpropagation algorithm. Simulation software was developed using Matlab and Simulink, Neural Network Toolbox.

The results obtained in the specialized networks simulation for images recognition are as follows:

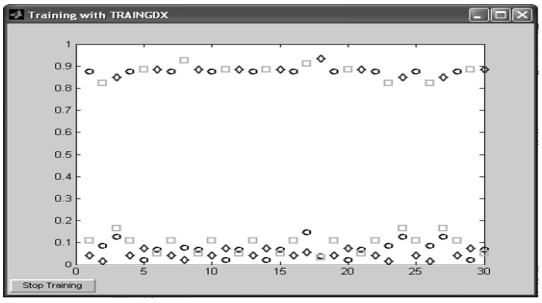
In order to obtain the data of the memories of weights, 450 patterns of training with different characteristics have been used, taking into account the fact that all correspond to an image of the same class (rectangular, circular and triangular shapes)

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 11, where the graph shows the stages used for the algorithm to converge with the targets of the parameters [30].



(a) Training



(b) Simulation

Fig. 11 Training Results

As an example of the application of the system for the recognition of signals by means of singularities, figure 11a shows the number of necessary iterations of training for one of the networks specialized in recognizing acute-angled corners.

Figure 11b, shows the results of the system (uncertainty module) when they present/display 30 images that contain the vectors that are characteristic in a correct sequence corresponding to the recognition of triangular signals. A region of 6x5 (columns*rows) pixels has been used to detect the singularities. The classification mode has been implemented as a series of regions processing. First, the ROI extractor detects the RoI (Region of Interest), figure 12, sized 60x45 pixels, and stores it in the internal embedded RAM memory. Then, successive vector of characteristic are extracted from the RoI, and sent to the TNN to be processed. So, dividing the RoI in 6x5 sized-regions, results in 10x9 data input vector in one RoI. This is the actual amount of data being processed in each image field, resulting in 90 vectors of

30 pixels each one. This has been accomplished by sweeping the RoI, and by sending each vector of characteristics to the TNN, and by storing the result associated to each region. In this way, probability maps of possible detected singularities are obtained so that the uncertainty stage can decide whether a signal has been detected or not, as shown in figure 11b.

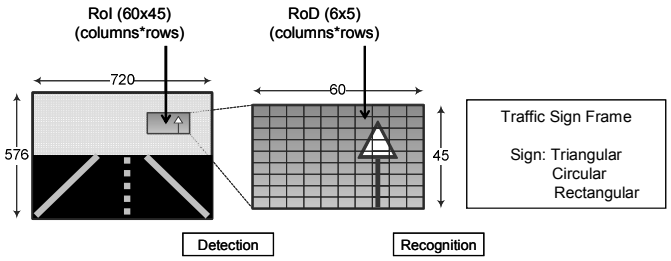


Fig. 12 Traffic Sign Recognition

In addition, further simulations in MATLAB, have established that a Q8.16 format is accurate enough to quantify weights and biases. This reduction in the bit width leads to a reduction in the resources consumed by the network. But more importantly, a great increase in the maximum operating frequency is also achieved. This is a key factor if we want to enhance the system.

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.

	Implementation
Logic Elements (LEs)	6,463 / 20,060 (32 %)
Total Memory Bits	72,416 / 294,912 (25 %)
M4Ks	47 / 64 (73 %)
Frequency	47.21 MHz

	KNU	
	GNN	RoI Extractor + RoI Buffer
Logic Elements (LEs)	4930 / 20,060 (25%)	294 / 20,060
Total Memory Bits	24480 / 294,912 (9%)	2880 / 294,912
M4Ks	34 / 64 (54 %)	1 / 64
Frequency	47.21 MHz	

Fig. 13 TNN Implementation

Fitting results are included for the whole Recognition System (see figure 13). 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.

Figure 14 shows fitting results for the Learning Algorithm execute on-line over the XILINX VIRTEX-II PowerPc, when the Reconfiguration Mode is needed. This Learning Algorithm was coded in C language (400 code lines). Some interesting data are:

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

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

Fig. 14 Learning Algorithm Implementation

VII. CONCLUSIONS

We have proposed and designed a new hardware architecture system for neural network based on specialized Tiny Neural Networks (TNN) for image recognition. One of the most important features of Tiny Neural Networks (TNN) is their on-line learning ability. Those 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. Autonomous robots or intelligent systems for cars use this kind of system.

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] M. Xiaobin, J. Lianwen, S. Dongsheng, Y. Junxun, A mixed parallel neural networks computing unit implemented in fpga, IEEE Int. Conf. Neural Networks & Signal Processing, Nanjing, China December 2003.

[11] M. Avogadro, M. Bera, G. Danese, F. Leporati, A. Spelgatti, The Totem neurochip: an fpga implementation, Proceedings of fourth IEEE International Signal Processing and Information Technology, 2004, pp 461-464.

[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).