DUAL-PROCESSOR NEURAL NETWORK IMPLEMENTATION IN FPGA

Diego Santos, Henrique Nunes, Fernando Morgado-Dias

Madeira Interactive Technologies Institute and Centro de Competências de Ciências Exactas e da Engenharia, Universidade da Madeira

Campus da Penteada, 9000-039 Funchal, Madeira, Portugal.
Tel:+351 291-705150/1, Fax: +351 291-705199

Abstract: Artificial Neural Networks have become a common solution for many real world problems. Many industrial, commercial and research applications need hardware implementation due to issues regarding stability, speed, price and size. This paper presents the implementation of a feed forward Artificial Neural Network in FPGA using two embedded processors. The processors used are Xilinx hardcore PowerPCs. To verify the implementation developed, a control loop of Direct Inverse Control was simulated using a Personal Computer and a FPGA, thereby implementing a direct and an inverse model of a system, respectively. The results obtained show that the hardware implementation works properly and introduces no additional error. Copyright CONTROLO2012.

Keywords: Artificial Neural Network, FPGA, embedded processor, Xilinx, Direct Inverse Control

1. INTRODUCTION

Nervous cells, or neurons, are excitable cells specialized in receiving and sending electrical and chemical impulses. These cells are constituted by a nucleus, axons and dendrites. The nucleus controls the impulses originating from other neurons and generates a response. Axons are responsible for the propagation of the message to other neurons. Finally, dendrites act like bridges, by connecting the nucleus of one neuron to various axons from other neurons (ADEAR, 2010). The natural neuron is represented in figure 1.

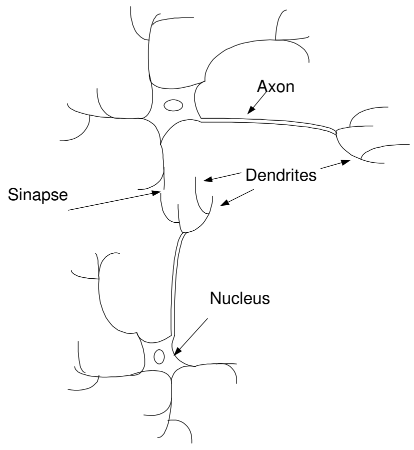


Fig. 1. Representation of a natural neuron (Dias, 2005).

This behavior can be mimicked by a computer, thus originating in a simplified neuron which is known as an artificial neuron. Artificial neurons can be modeled as shown in figure 2, in which inputs are multiplied by their respective weights (w), thereby simulating an inhibition or stimulation behavior. A polarization signal, or bias, is then added to these inputs, which are then fed into an activation function, whether it be linear or not, thus providing the output result.

Artificial Neural Networks (ANN) are composed of these artificial neurons, usually disposed in layers and connected so as to form networks.

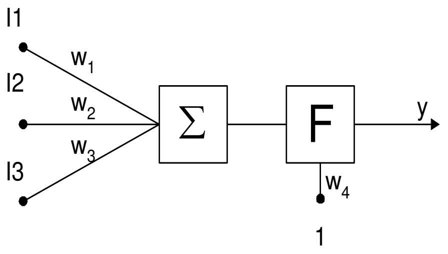


Fig. 2. Neuron structure.

The Perceptron or Feed-forward Neural Network (FNN) was the first neural network to be built. It was a *feed forward* type network. Figure 3 shows one network of this kind, where it is possible to see that there are no lateral or feedback connections (Dias, 2005).

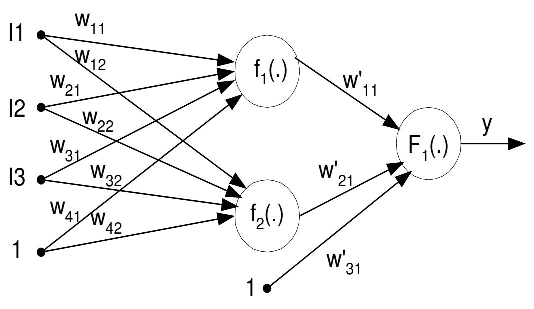


Fig. 3. Example of a FNN.

The neuron implements the general equation:

$$y = F(\sum_{i=1}^{n} Ii.wi) \qquad (1)$$

where usual functions for F are sigmoidal, linear and hard limit.

A FNN is composed of an input layer, one or more hidden layers, with one or more neurons, and an output layer, where the neurons are frequently linear.

The Multi Input Single Output FNN in figure 3 implements the following general equation:

$$y = F_1(\sum_{j=1}^{nh} w'_{j1} f_j(\sum_{l=1}^{nI} w_{lj} I_l)) \qquad (2)$$

2. HARDWARE IMPLEMENTATION

ANN have become a common solution for many real world problems. Many industrial, commercial and research applications need hardware implementation due to issues regarding stability, speed, price and size. Connecting a Personal Computer to each application is not a solution in many situations outside the academic environment: the operating system is frequently not stable enough for industrial application; the sequential processing does not provide the speed and the parallelism which are required by some implementations; the price is still high for some implementations; and the size can be too large for reduced scale applications.

In order to choose a platform for hardware implementation in one specific application, one needs to consider many factors such as the price, performance and accuracy.

Developing an Application Specific Integrated Circuit (ASIC) is the best solution as the user can specify all the characteristics, meeting the needs of the application. This solution is, nevertheless, too expensive to be considered in most applications. An acceptable compromise between price and performance can be obtained using a Field Programmable Gate Array (FPGA).

FPGAs also offer the advantage of being reprogrammable, thereby eliminating or reducing the costs of prototypes.

In the review of the literature it is possible to find many implementations of ANN using FPGAs as platforms, but only a few use the embedded processors as the main element in the implementation (Hoelzle, 2009)(Tabari, 2006).

The embedded processor presents the advantage of being easy to program, thereby allowing a fast implementation. This is an advantage over the traditional hardware configuration possibility of FPGAs.

In this paper, an implementation was developed using the embedded processors available in the FPGA. This research is part of a larger project which includes testing other solutions for the hardware implementation of ANN.

3. MICROBLAZE VS. POWERPC

The Xilinx company provides three embedded microprocessors: PowerPc, MicroBlaze, and PicoBlaze. The last alternative was not tested, since it is the one which provides the lowest performance. The characteristics of the first two alternatives are summarized in the following paragraphs.

The Xilinx MicroBlaze processor, on the one hand, is a 32 bit Reduced Instruction Set Computer (RISC) architecture, with a maximum processing frequency of 100 MHz, which originates a processing performance of 280 DMIPS (Dhrystone Million Instructions per Second). Furthermore, it has Harvard architecture with a 32 bit bus and cache memory for faster access to data and instructions. It has a level 3 or 5 pipelining architecture, depending on the MicroBlaze's architecture. This means that it is capable of executing 3 or 5 instructions on a single clock cycle (*Xilinx*, 2010a).

On the other hand, the Apple/IBM/Motorola PowerPC is a third generation 32 bit RISC processor with a level 7 pipelining architecture. It has a processing frequency of 400 MHz, resulting in a processing performance of 700 DMIPS, which is almost three times the performance of the MicroBlaze (*Xilinx*. 2010b).

These characteristics result from the fact that MicroBlaze is a softcore processor, i.e., a processor which is built on the assemblage of the logical blocks composing the FPGA, while PowerPC is a hardcore processor drawn in the wafer which constitutes the FPGA.

4. SYSTEM AND HARDWARE SPECIFICATION

Using the development software provided by Xilinx, namely the Xilinx Platform Studio (XPS) for hardware configuration of the blocks, such as the processor and peripherals, and the Software Development Kit (SDK) for creation/compilation of C/C++ programs, the system which will be described in the following section was created on a Virtex-5 ML510 XC5VFX130T.

Regarding the hardware, using XPS, several configurations were tested. The configuration with the largest set of resources used two PowerPCs with a clock frequency of 300 MHz, bus frequency of 100 MHz, JTAG interface and a Floating Point Unit (FPU) with double precision. These processors had as peripherals a Block Random-Access Memory (BRAM) with 128 kB, a timer to verify the time needed for the calculations and a serial interface (RS232) with baudrate of 230400 bits per second, using 8 bit words with no parity and one stop bit. In addition to this, both processors shared the DIMM memory, and a mutex.

Concerning the SDK, after exporting the hardware from XPS, it was necessary to create a software platform for each processor in which the xilkernel, which is a simple operating system that provides several functionalities (i.e. threading, semaphores, timers, etc.), was activated, since the use of a timer was found to be necessary. Then, in order to avoid data being corrupted in the BRAM, the heap and stack sizes were increased to 0x2000 bytes each.

5. PROCESSING SPEED COMPARISON BETWEEN MICROBLAZE AND POWERPC

As stated in the previous section, the processing speeds of both processors in the test are quite different. Nevertheless, it is interesting to run the same code in both processors and analyze the results obtained in terms of speed.

For this analysis, it was necessary to create a C program which would be tested in the FPGA with a single processor (MicroBlaze or PowerPC), in order to register the network's processing time for these two processors. To ensure that both processors would be on even terms, they were both configured with a processor frequency of 125 MHz (this is the most important parameter concerning the network processing time).

The single processor C program is exemplified by the flowchart present in figure 4.

The activation function used was the hyperbolic tangent in its simplified form, expressed as follows:

$$f(x) = 1 - \frac{2}{1+e^{2x}} \tag{3}$$

Furthermore, the values are expressed in double floating point, which is 64bits of precision.

For testing purposes, two neural network models were used. The first, FORG6700, consisted of 5 inputs, 3 hidden neurons and 1 output neuron, a 5-3-1 configuration; and the second, FORGA001, was composed of 5 inputs, 8 hidden neurons and 1 output neuron, a 5-8-1 configuration.

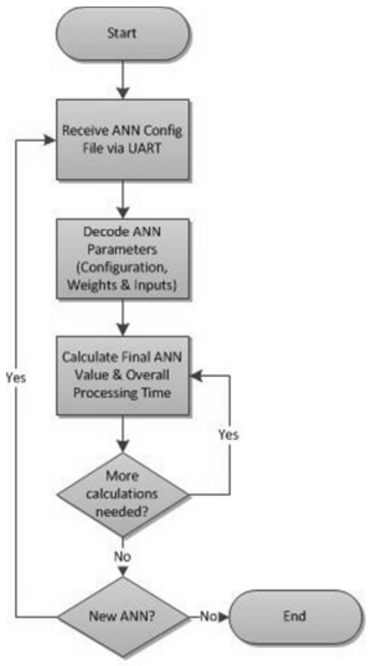

Start
Receive ANN Config File via UART
Decode ANN Parameters (Configuration, Weights & Inputs)
Calculate Final ANN Value & Overall Processing Time
More calculations needed?
Yes
No
New ANN?
Yes
No
End

Fig. 4. Flowchart of the single processor's program.

Figure 5 accounts for the difference in processing time of both neural networks for the two processors analyzed in the initial test. Note that the values expressed only represent the time necessary to calculate the ANN's output. Network configuration is not taken into account.

Taking into consideration figure 5, it is visible that the PowerPC can perform the same calculations as the MicroBlaze in approximately seven hundred times less time. Note that since PowerPC processing times are much shorter, they are almost unnoticeable in figure 5.

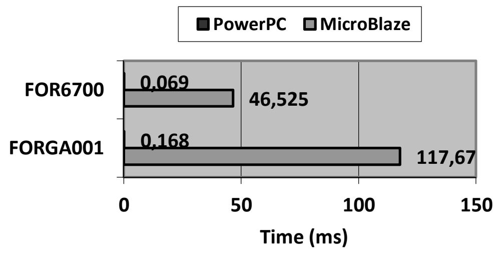


Fig. 5. Comparison between MicroBlaze and PowerPC

This result demonstrates a much bigger difference between MicroBlaze and PowerPC than would be expected from the DMIPS information. Probably due to the complexity of the instructions used in the calculation of the ANN's output, PowerPC can make better use of its higher level pipeline when compared to MicroBlaze.

6. PROCESSING SPEED RESULTS OF THE SINGLE-POWERPC AND DUAL-POWERPC SOLUTIONS

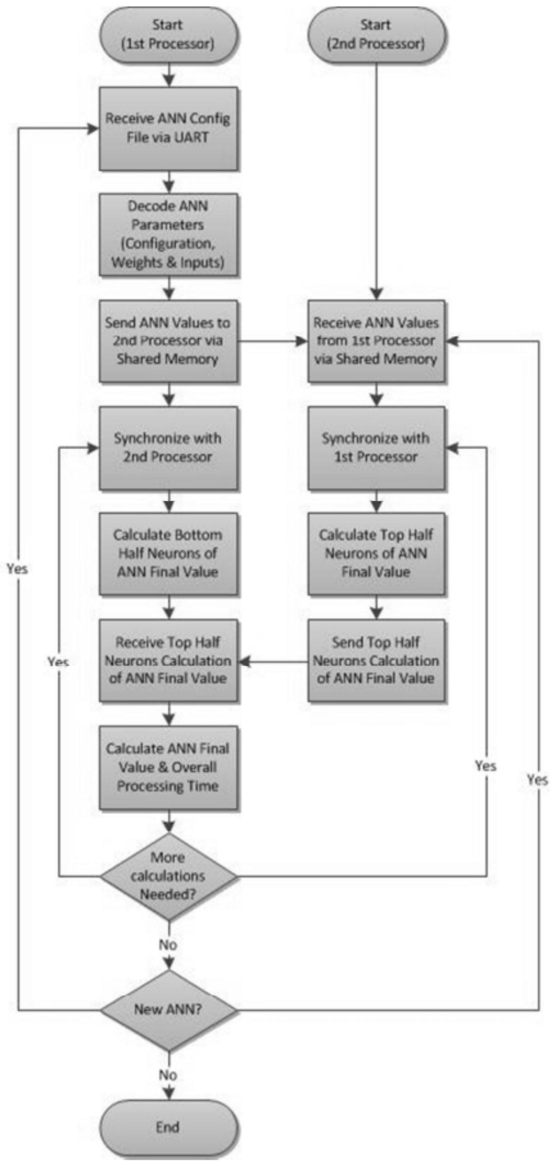
As one of the main objectives of this project was to implement an ANN using two processors, and taking into consideration the results previously obtained, an obvious choice to achieve this objective would be to use two PowerPCs.

In order to perform the tests with both PowerPCs sharing information, another C program had to be developed, although the main characteristics were the same as for the single processor. Since the memory posed some synchronization problems when using the maximum frequency of the PowerPCs, these had to be trimmed down do 300 MHz.

Using two processors, the ANN needs to be divided into two sub-networks. This allows for the splitting of the calculations between the processors, in which one will calculate the top half of the neurons and the other will calculate the bottom half of the neurons of the ANN (consider figure 3 as an example), reducing the overall processing time. Figure 6 shows the flowchart of this new implementation.

Figure 7 is obtained from the Integrated Software Environment (ISE) from Xilinx and illustrates the connection of the two PowerPc processors and their peripherals.

As can be seen in figure 8, it is evident that, although the processing time with two processors does not reach half of the time achieved with one processor at 300 MHz, it does have quite a significant improvement.



Start (1st Processor)
Start (2nd Processor)
Receive ANN Config File via UART
Decode ANN Parameters (Configuration, Weights & Inputs)
Send ANN Values to 2nd Processor via Shared Memory
Receive ANN Values from 1st Processor via Shared Memory
Synchronize with 2nd Processor
Synchronize with 1st Processor
Calculate Bottom Half Neurons of ANN Final Value
Calculate Top Half Neurons of ANN Final Value
Receive Top Half Neurons Calculation of ANN Final Value
Send Top Half Neurons Calculation of ANN Final Value
Calculate ANN Final Value & Overall Processing Time
More calculations Needed?
New ANN?
End
Yes
Yes
Yes
Yes
No
No

Fig. 6. Flowchart of dual processor program.

It is not possible to obtain a reduction of the processing time by half, due to the fact that one of the processors needs to control the communication and sharing of data.

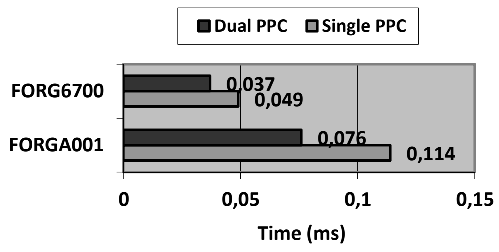


Fig. 8. Comparison between single and dual PowerPc implementation.

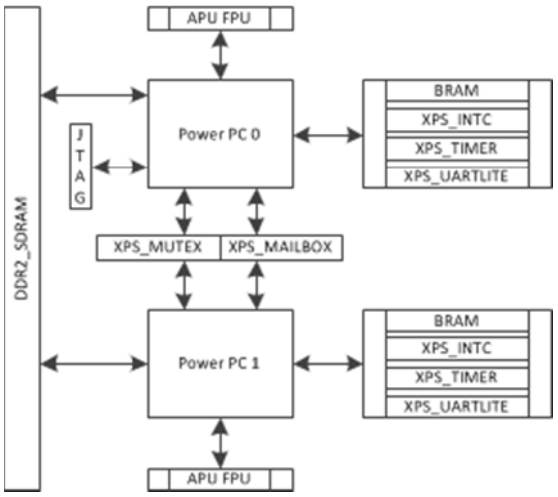


Fig. 7. Block diagram of the two PowerPCs and peripherals.

7. CONTROL LOOP SIMULATION

Once the analysis of the best solution in terms of processors was completed, a testing scenario was developed in which there would be a system to control, an oven and a controller. Further information on this system can be found in (Vieira, 2004).

The controller and the controlled system were implemented in a computer, using Matlab, and the FPGA, respectively. The connection between them was established via a serial interface at a baudrate of 230400 bits per second or 28800 bytes per second.

In order to maintain a centralized supervision over the loop, a Graphic User Interface (GUI) was created in Matlab. This GUI allowed sending the ANN's configuration files to the FPGA, so that it could implement the controller for one particular network. Furthermore, the GUI would receive the values calculated by the controller, feed them to the system which was to be controlled and display a chart which would allow for the comparison of the actual values of the system with the desired reference values (i.e. those which the researchers wanted the system to yield).

In order to make the loop work properly, the models were trained in pairs. This meant that, for every system to be controlled (represented by a direct model), there would be a controller (inverse model). In table 1 it is possible to view the models used and their configurations.

The control results for the three ANN loops can be seen in figure 9.

If a comparison is made between the results of the various networks, it is possible to assert that the loop which uses the model 6700, on the one hand, has values which are very similar to the reference ones, though they have some oscillations in the transitions. The 25F01 model, on the other hand, does not change fast enough to follow the reference values, thus showing that the number of hidden neurons in the direct model has a great influence on the settling time of an ANN. Regarding the 25FMD model, it has the fastest approximation on the transitions in relation to the reference values.

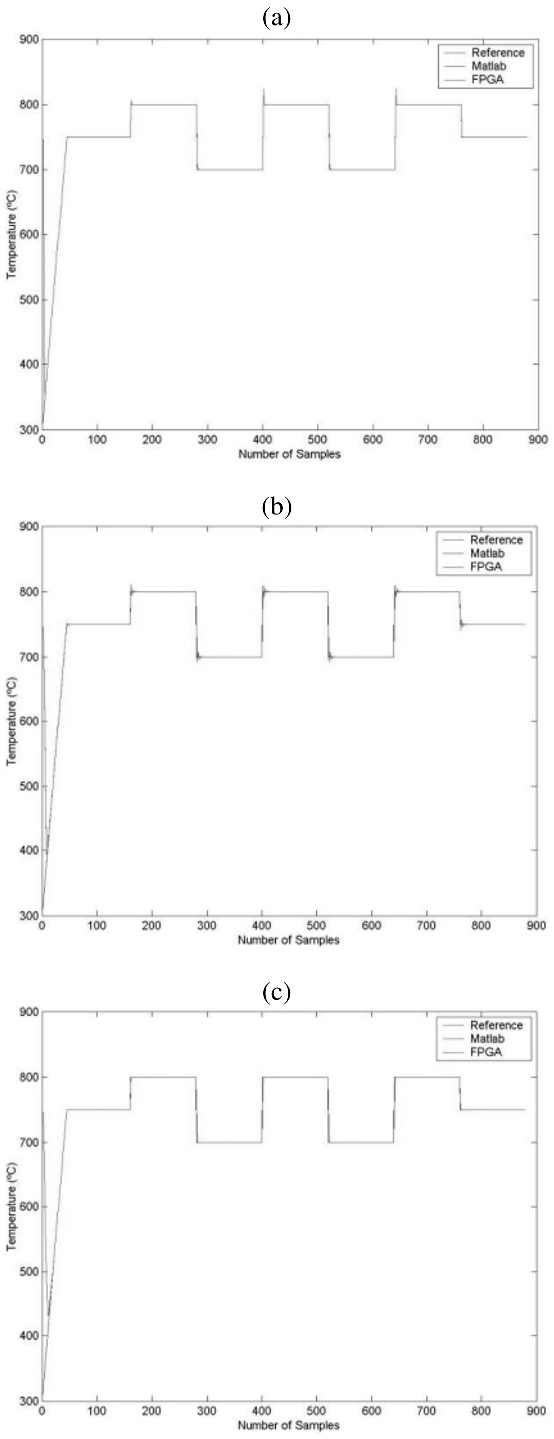


Fig. 9. Control results for a) Model 6700; b) Model 25F01; c) Model 25FMD.

Table 1. Models and direct/inverse ANN configurations

Model	FPGA (Inv)	Matlab (Dir)
6700	5-4-1	5-3-1
25F01	9-4-1	8-2-1
25FMD	5-8-1	5-8-1

The analysis of the previous paragraph is confirmed by table 2 which illustrates the mean square error of the models using the control loop and a simulation performed solely on Matlab (controller + system).

These values display a very small difference of less than 0.1% in both control loops, which thus confirms that the implementation is correct and that the FPGA implementation introduces no errors. In fact, during this FPGA implementation, the exact calculations are used and no simplification is introduced.

Table 2. Mean square error of the models

Models	Matlab	Control loop	Change
6700	17.48×10^{-6}	17.50×10^{-6}	+0.109%
25F01	0.1623	0.1622	-0.062%
25FMD	0.1808	0.1810	+0.111%

8. CONCLUSIONS

This paper describes the implementation of an ANN in a FPGA using two embedded PowerPCs.

The use of embedded processors simplifies the engineer's work. Hence, instead of designing the hardware, it is only necessary to program the processor.

The results show, as expected, that PowerPC is much faster than MicroBlaze and that with two processors a significant improvement in performance can be obtained. Furthermore, the difference between an implementation only in Matlab and another with Matlab and an FPGA is very small, reaching a maximum of 0.11%.

ACKNOWLEDGEMENT

The authors would like to acknowledge the Xilinx University program for their support and the software used in this project.

The authors would like to acknowledge the Portuguese Foundation for Science and Technology for their support for this work through project PEst-OE/EEI/LA0009/2011.

REFERENCES

The Alzheimer's Disease Education and Referral (ADEAR): (2010). *las neuronas y su funcionamiento*. [On-Line]. Available: http://www.nia.nih.gov/Alzheimers/Publications/LaEnfermedaddeAlzheimer/Parte1/neuronas.htm

Mathworks. (2010). Neural Network Toolbox, User's Guide.

Electronica Mexico. (2010). *Redes neuronales artificiales*. [On-Line]. Available: http://electronica.com.mx/neural/informacion/perceptron.html

Xilinx. (2010). *MicroBlaze Processor Reference Guide*. [On-Line]. Available: http://www.*Xilinx*.com/support/documentation/sw_manuals/mb_ref_guide.pdf,

Xilinx. (2010). *Embedded Processing*. [On-Line]. Available: http://www.*Xilinx*.com/technology/embedded.htm

Dias, F. M. (2005). Técnicas de controlo não-linear baseadas em Redes Neuronais: do algoritmo à implementação. PhD thesis. (Universidade de Aveiro)

Hoelzle, G. & Dias, F. M. (2009). Hardware Implementation of an Artificial Neural Network with an Embedded Microprocessor in a FPGA, 8th International Conference and Workshop on Ambient Intelligence and Embedded Systems", Funchal.

Vieira, J. & Dias, F. M. & Mota, A. (2004). Artificial neural networks and neuro-fuzzy systems for modelling and controlling real systems: a comparative study, Engineering Applications of Artificial Intelligence, 17(3), 265-273.

Tabari, K. & Boukadoum, A. & Bensaoula, D. & Starikov, D (2006). Neural Network Processor for a FPGA- based Multiband Fluorometer Device, The Internacional Workshop on Computer Architecture for Machine Perception and Sensing.