USING EMBEDDED PROCESSORS IN HARDWARE MODELS OF ARTIFICIAL NEURAL NETWORKS

DENIS F. WOLF, ROSELI A. F. ROMERO, EDUARDO MARQUES

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Av. Trabalhador São-carlense, 400
13560-970 - SÃO CARLOS – SP
BRASIL

E-mails: {denis, rafrance, emarques}@icmc.sc.usp.br

Abstract: Artificial Neural Networks are applied for solving a wide variety of problems in several areas such as: robotics, image processing, and pattern recognition. Many applications demand a high computing power and the traditional software implementation are not sufficient. Hardware implementations of neural network algorithms are very interesting due their high performance. In this paper, an implementation that joins the software flexibility with the excellent hardware performance has been performed through the use of reconfigurable computing and embedded processors technologies.

Keywords— Neural Networks, MLP, FPGA, Reconfigurable Computing, Embedded Processors

1 Introduction

The Multilayer Perceptron (MLP) is a neural network model that is being widely applied in the solving of diverse problems. A supervised training is necessary before the use of the neural network. A highly popular learning algorithm called back-propagation is used to train this neural network model (Haykin, 1999). Once trained, the MLP can be used to solve classification problems. This process involves a large number of complex arithmetical operations but sometimes, the software implementations of the neural networks do not have the performance desired.

An interesting method to increase the performance of the model is by using hardware implementations. The hardware can do the arithmetical operations much faster than software. Nowadays, the reconfigurable computing has been used as a very interesting technique to project and prototype hardware because it also allows a very fast design and prototyping. But, pure hardware implementations have some disadvantages such as synchronization in complex implementations. Hardware implementations lack some flexibility and many tasks such as training the neural network are very difficult to be implemented in hardware. Most hardware implementations does not allow on-chip learning. The presented implementation does not do it too.

To explore the software flexibility and hardware performance, an implementation using embedded processors is presented in this paper. Some experiences have been done using the Altera Excalibur (Altera, 2000) development kit, which includes the Altera Nios (Altera, 2000b) softcore processor and the APEX20k FPGA (Field Programmable Gate Array) device. Very interesting results have been obtained as it is shown on the Section 4.

The rest of this paper is organized as it follows. In Section 2, a brief description about the FPGA technology and Reconfigurable Computing is presented. A description of the MLP model that has been implemented is presented in Section 3. In Section 4, the hardware implementation is discussed with details. Finally, in Section 5, some conclusions and future works are presented.

2 Reconfigurable Hardware and FPGA Technology

The main architecture of a FPGA device consists on an array of logic blocks (configurable cells), enclosed within a single chip. Each one of these cells has a computational capability to implement logic functions and much of these operations can occur at the same time. The communications among the cells are done by interconnection resources (Rose et al., 1993) (Brown & Rose, 1996).

The FPGA technology development has been allowing great performances, high-density levels of integration, and low cost prices. This fact makes shorter the distance between the FPGA and the chips implemented directly on silicon, allowing this technology to be used in the construction of more complex computer architectures (Donachy, 1996).

The utilization of FPGA to realize computing lead to a new general class of computer organization called Reconfigurable Computing Architecture (Dehon, 1999). This class of architecture provides a highly custom-built machine that can attend to the instantaneous needs of an application. Thus, it is possible to have the application running over a spe-

cially developed architecture, bringing more efficiency than general-purpose processors. In other words, to achieve the best performance of an algorithm, it has to be executed in a specific hardware.

With this inherent speed and adaptability, the reconfigurable computing can be specially exploited on applications that need high performance like parallel architectures, image processing and real-time applications.

2.1 Altera's Excalibur Development Kit

The Altera Excalibur development kit comprehends all the hardware and the software necessaries to compose a complete SOC (System On a Chip) devices development. The SOC paradigm consists in the development of complete systems. All the components of a system such as: processor, memories, timers, and interfaces have been put in just one chip. Almost all of the hardware developers have been using this paradigm.

The Excalibur kit includes the Nios softcore (a complete 32-bit RISC processor), the GNUPro compiler (a C compiler for the Nios processor), the Quartus development tool (used to configure the APEX FPGA device), and the development board, which includes the APEX20k200E FPGA device.

The Nios is a complete processor that adds lots of functionality and versatility to a hardware project. It can be programmed using the C language through the GNUPro compiler. The Nios and dedicated hardware can be used together on the same chip. It has all necessary interfaces to handle memories, dedicated hardware, serial and parallel ports, etc… In this work, it is proposed a dedicated hardware model of a Neural Network that is handled by the Nios processor. Some performance comparisons will be presented later on this paper.

3 The MLP Neural Network Model

A hardware implementation of MLP has been built for solving the classification problem for the iris data set. The iris data set is a very popular data set that has been widely used for the test of learning algorithms.

The iris set contains a database with 150 flowers, classified into 3 different groups. Each sample (or pattern) has 4 attributes. So, the MLP topology that has been considered for the classification of this particular data set is constituted by 3 layers being 4 neurons on the first layer, 4 neurons on the hidden layer, and 3 neurons on the output layer (Figure 1).

The main purpose of the first layer is just to deliver the input signals for all the neurons of the hidden layer. As the signals are not modified by the first layer neurons (the neurons do not have arithmetical operations), the first layer can be represented by a single set of busses in the hardware model.

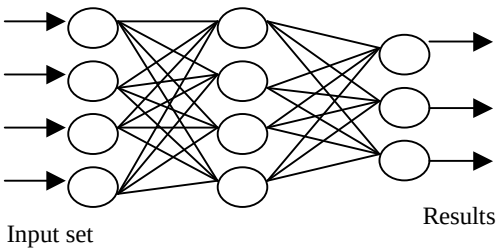

Figure 1: MLP Neural Network

4 Hardware Implementation

The three basic problems to implement neural networks in reconfigurable hardware are: floating-point numbers representation, neuron's transfer functions (non-linear), and device capabilities.

Neural Networks, in general, work with floating-point numbers. Working with floating-point numbers in hardware is a difficult problem because the arithmetic operations are more complex than with integer numbers. Further more, the dedicated circuits for floating-point operations are more complex, slower, and occupy a larger chip area that integer number circuits (Schonauer, 1998) (Moerland, 1997). A solution used to make this project easier and improve its performance has been converting the floating-point numbers to integer numbers. Of course it implies in some loss of precision but in this particular case, good results have been achieved.

Other problem for representing the arithmetic operations using digital hardware is related with the neuron's transfer functions. Some transfer functions like the sigmoid function (frequently used in the MLP model) need some modifications to make easy the design of the hardware. In this case, the sigmoid function has been substituted by a piecewise linear function (Figure 2).

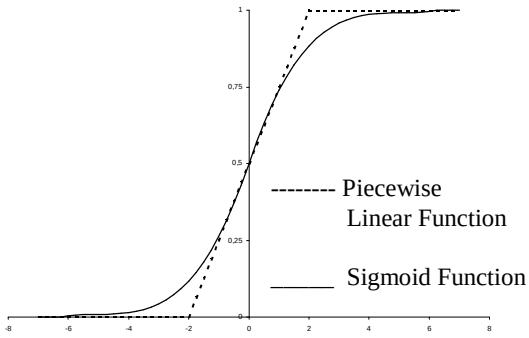

Figure 2: Sigmoid and Piecewise Linear Func-

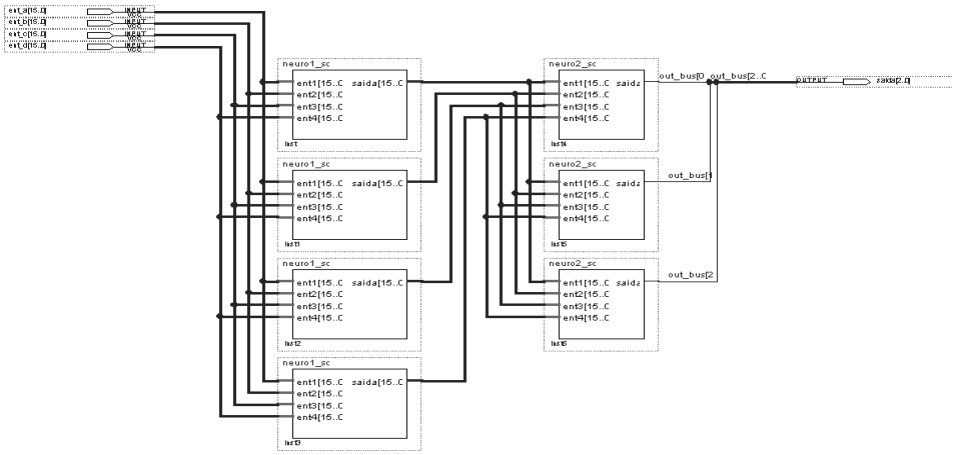

Figure 3: The Complete Neural Network Dedicated Hardware

The capability of the reconfigurable hardware devices is still being a limitation. Although there are reconfigurable chips with up to 10 millions of logic gates, some hardware implementations need more than this.

The use of embedded processors also helps in the efficient use of the FPGA devices. Only the critical parts of the algorithms have been built in dedicated hardware, the less important parts can be implemented in software and run in the embedded processors as it can be seen on the implementation done in this research.

4.1 Implementation

The hardware model of the neural network has been designed with the Altera Quartus Design Tool. To generate a hardware model from software algorithm some simple logic and arithmetic blocks such as: multipliers, adders, divisors and logic gates have been used. Basically, each neuron has four multipliers to multiply each input value by the corresponding weight. The four results of the multipliers are added with the bias and finally, the transfer function delivers the neuron's output.

The complete diagram of the neural network can be seen in Figure 3. The four pins on the left side of the figure are the four input values. They are connected to the set of busses (first layer of the MLP model); these busses distribute the input signal to the next layer (hidden layer). The results are provided to the output layer and finally, the results are showed in the output pin (on the right side of Figure 3). This model needs 32ns to processing the input values and presenting a result.

This is a very fast implementation that can process up to 31.25 billions input sets per second. However, the neural network is only a part of some algorithms and the joining of an extremely efficient dedicated hardware with a flexible general-purpose processor is an interesting alternative for solving a great number of computational problems.

In this case, the neural network's hardware model has been joined to the Nios processor and some interesting results have been obtained. The Figure 5 shows the entire project.

The software part of the algorithm has been implemented using the C language. Stead dozens of source code lines that would process the neural only some I/O operations have been needed. In Figure 4, it is showed an example of how the neural network hardware can be accessed by the software, which is running in Nios processor. Another important advantage of the use of embedded processors is that the neural network can be trained on-chip (it is part of the future plans). The learning algorithm can be implemented in the high level language that runs on the processor and it adjusts the neuron's weights. The learning algorithms of the MLP model are very difficult to be implemented directly in hardware and they are used few times in comparison to the executions of the classify algorithm. Consequently, the overall system performance is not affected.

```
out (#port 1, #value 1);
out (#port 2, #value 2);
out (#port 3, #value 3);
out (#port 4, #value 4);
result = in (#port5);
```

Figure 4: Software Algorithm that makes access to the hardware of the neural network

4.2 Results

The Nios processor takes 4 clock cycles to execute an "out" instruction and 8 clock cycles to execute an "in" instruction.

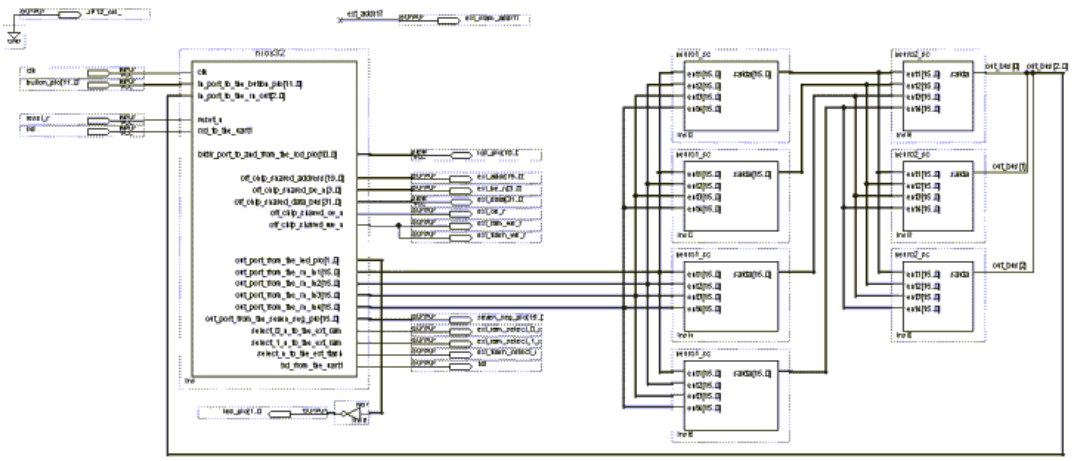

Figure 5: Nios processor and the neural network dedicated hardware

Type of Execution	Frequency	Execution Time
Dedicated Hardware	-	32 ns
Nios + Dedicated Hardware	40MHz	600 ns
Nios (only software)	40MHz	144 µs
Intel Pentium III	550MHz	23 µs

Table 1: Performance Comparisons

Consequently, a complete access to the neural network hardware needs 24 clock cycles to be concluded. The processor has been tested with a 40MHz clock frequency. The Table 1 shows the performance comparisons, where it can be noted the excellent performance of the hardware implementations. The pure software implementations have been compiled using the GNUPro (Nios) and Borland C++ 5.0 (Pentium III).

The neural network dedicated hardware has used approximately 50% of an APEX20k200E FPGA device and the 32-bit Nios processor has used approximately others 25% of the chip. This device has 8320 logic elements (526.000 maximum system gates).

5 Conclusions

The Neural Networks and the Reconfigurable Computing are two important and promising technologies to scientific researches. There are many applications for these two technologies, such as: the robotics area, imaging processing and pattern recognizing.

With the performance of the reconfigurable hardware, the use of the neural networks can be improved and more powerful applications can be obtained. For example, more precise images can be processed in a shorter period of time, a faster pattern recognition problem can be realized and a faster response time robot can be obtained. And through the use of the embedded processors, much more flexibility can be added to the project. The use of high-level languages allows that complex algorithms to be implemented and the very high performance of the dedicated hardware can speed up the critical parts of these algorithms.

As a future work, other models of neural networks will be implemented and the sequential processing will be explored in order to fit larger neural network topologies in just one chip.

Acknowledgements

This work has been partially supported by FAPESP and CNPq under the grant numbers: 2000/02959-3 and 133739/2000-7, respectively.

References

Altera Co., *"Excalibur Backgrounder"*, in URL http://www.altera.com, 2000.

Altera Co., *"Nios Embedded Processor"*, in URL http://www.altera.com, 2000b.

Brown, S.; Rose, J. "*Architecture of FPGAs and CPLDs*", A Tutorial, IEEE Design and Test of Computers, vol. 13, no. 2, pp. 42-57, June 1996.

Dehon, A.; Wawrzynek, J., "*Reconfigurable Computing: What, Why, and Design Automation Requirements*", In Proceedings of the 1999 Design Automation Conference, pp. 610-615, June 1999.

Donachy, P., "Design and Implementation of a High Level Image Processing Machine using Reconfigurable Hardware", *Ph.D. Thesis, Dept. of Computer Science*, The Queen's University of Belfast, 1996.

Moerland P., Fiesler E., "*Neural Network Adaptations to Hardware Implementations"*, Handbook of Neural Computation E1.2: 1-13 Institute of Physics Publishing and Oxford University Publishing, New York, 1997.

Haykin S., *Neural Networks A comprehensive Foundation,* 2ª edition, Prentice Hall, 1999.

Rose, J.; Gamal A. E.; Vincentelli, A. S., "*Architecture of Field-Programmable Gate Arrays*", Proceedings of the IEEE, vol. 81, no. 7, p.1013-28, 1993.

Schonauer T., Jahnke A., Roth U. Klar H., "*Digital Neurohardware: Principles and Perspectives*", In: Proc. Neuronale Netze in der Anwendung - NN'98, Magdeburg, invited paper, pp.101-106, February 1998.