A Pipeline Hardware Implementation for an Artificial Neural Network

Denis F. Wolf, Gedson Faria, Roseli A. F. Romero, Eduardo Marques, Marco A. Teixeira, Alexandre A. L. Ribeiro, Leandro C. Fernandes, Jean M. Scatena, Rovilson Mezencio

Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo
Av. Trabalhador São-carlense, 400 13560-970 - São Carlos – SP - Brasil
{denis, gedson, rafrance, emarques}@icmc.sc.usp.br

Abstract: Artificial Neural Networks are computational devices inspired by the human brain for solving problems. Currently, they are being widely applied for solving problems in several areas such as: robotics, image processing, pattern recognition, etc.... The neural network model, Multilayer Perceptrons, is one of the most used due to its simple learning algorithm. However, its convergence is very slow. To take advantage of the massive parallelism inherent to this model, a hardware parallel implementation should be performed. There are different hardware parallel implementations for this particular model. This paper presents a reconfigurable hardware parallel implementation for Multilayer Perceptrons by using pipelines. Tests realized showed that the use of pipelines speeded up the execution time of the hardware parallel implementation.

1. Introduction

The most general-purpose computers are based on the von-Neuman architecture, which is sequential in nature, on the other hand, artificial neural networks profit from massively parallel processing [Schonauer et al. 1998]. The Multilayer Perceptrons (MLPs) model is being widely applied successfully to solve difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm [Haykin 1999]. An interesting method to explore this parallelism is using hardware implementations. The good performance and the natural parallel operation of the hardware devices, turn it as an attractive option to implement neural network algorithms. Today, the reconfigurable computing has been used as a very interesting technique to project and prototype hardware. Reconfigurable computing joins the hardware performance with the software flexibility [Gonçalves et al. 2000].

The hardware implementation of neural network has been already discussed in many articles, such as [Pérez-Uribe and Sanchez 1996], [Demian et al. 1996], and [Molz et al. 2000]. Some principles and perspectives of the digital neurohardware are discussed by Schonauer et al. [Schonauer et al. 1998] with some implementation examples. Suggestions of adaptations to hardware implementation of neural nets are presented by Moerland et al. [Moerland and Fiesler 1997]. According to Moreno [Moreno et al. 1999a][Moreno et al. 1999b], a reconfigurable device was proposed, which can be used to realize the most of arithmetic operations necessary to implement neural networks in hardware.
As it is known, two basic hardware models can be considered to hardware projects: the analog hardware and the digital hardware. The analog hardware can reproduce better the non-linear functions, which are frequently present in the transfer functions of the neurons belonging to a MLP. The digital hardware is not so accurate, but provides the projects with simplicity and good performance.

In this work, a pipeline hardware implementation for a MLP by using the digital reconfigurable hardware is proposed. For this, some modifications in the transfer functions of neurons have been realized and the float pointing numbers have been converted to integer numbers.

This paper is organized as it follows. In Section 2, a brief description about the FPGA technology and Reconfigurable Computing is presented. Some modifications in the neuron model necessary for improving the performance and adjust the implementation are presented in Section 3. In Section 4, the hardware implementation is discussed based on a simple example and the obtained results are also showed. Finally, in Section 5, a conclusion and future work are presented.

2. Reconfigurable Computing and FPGA technology

A Field Programmable Gate Array (FPGA) is 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 to do a route allowing communication among the cells, all this operations can occur at the same time on entire array of cells [Rose et al. 1993][Brown and Rose 1996].

In the FPGA architecture, the interconnection between the configurable logic blocks takes place through electrically programmable commuter like in the traditional Programmable Logic Devices (PLD). The FPGA matches the gate array's versatility and the PLD's programming capability [Donachy 1996].

The FPGA technology has evolved towards better performance, higher density levels and lower cost price. This fact makes shorter the distance between the FPGA and
the chips implemented directly on silicon, allowing that this technology be used to construct more and more complex architecture.

The utilization of FPGA to realize computing lead to a new general class of computer organization called Reconfigurable Computing Architecture [DeHon and Wawrzynek 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 specially 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.

3. Neuron Adaptations

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

![Figure 2: MLP Topology](image2.png)

The iris set contains a database with 150 flowers, classified into 3 different groups. Each sample (or pattern) has 4 attributes. So, a MLP topology has been considered for the classification of this particular data set 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 2).

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 (Figure 3).

![Figure 3: The network topology without the first layer](image3.png)
The model MLP, in general, receives floating pointing numbers as input values. Working with floating point numbers in hardware is a problem because the arithmetic operations are more complex than with integer numbers. Further the dedicated circuits for these operations are more complex, slower, and faster (occupying a larger chip area). A good idea to make easy the project and improve the performance is to convert the float pointing numbers to integer numbers. This implies in some loss of precision but in most of cases is sufficient to achieve good results. 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 (Figure 4) 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 as that shown in Figure 4. The precision is decreased again but in this case good results have been obtained.

4. Hardware Implementation

The hardware model of the neural network has been designed and simulated with the Altera Quartus Design Tool [Altera 2000]. 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 5. The four pins on the left side of the picture are the four input values. They are connected to the set of busses (first layer); 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 5).

In this particular implementation, the floating-point numbers (input set, bias, weights, and results) have been converted to 16-bit integers. It has allowed that 100% of precision in the classification of the iris database set. The neural network has been trained in software once this hardware implementation does not allow on-chip training.

The input signals need 32 ns to be processed by the circuit. In Figure 6, the waveform analysis are showed, and the input and output signals can be seen over the time.

A pipeline has been used to improve the performance. In this case, the neural network has been divided in two stages (Figure 7), and each stage can process a different set of input signals at the same time.

The hidden layer receives the set of input values, when it finishes the process the results from the hidden layer are sent to the output layer. At the same time, the hidden layer receives a new input set and both layers operate with different values. When both layers finish their task, the hidden layer sends the results to the output layer again and
receives another input set. At the same time, the output layer shows the results and receives the results from the hidden layer. Through the use of this technique, the neural net can work with two different sets of input values and, a better performance has been obtained. The time analysis can be seen in Figure 8.

The same process that was applied for the neural network can be applied for each neuron. A neuron can be divided in two stages and can process two different sets of values in each stage. The neuron has been divided as it follows: the first stage is constituted by the set of multipliers, and the second stage is constituted by the adders and the circuit of the transfer function, as it can be seen in Figure 9.

Through the use of pipeline neurons in a two stages pipeline network, a four stages pipeline circuit has been obtained. The synchronization of all the circuits with pipeline has been done by the use of clock signals. Using the four stages pipeline network, the circuit can process four different sets of input signals at the same time. The time analysis can be seen in the Figure 10.

In the Table 1, all of the hardware and software implementations are compared. A natural performance difference between the hardware and the software models can be
seen. The modifications, such as: the use of integer numbers and the use of piecewise transfer function, present better results even in software implementation. Joining the neuron modifications with a parallel hardware and a pipeline, a very efficient circuit has been obtained.

5. Conclusion

The Neural Networks and the Reconfigurable Computing are two important and promising technologies to scientific researches. When combined one can obtain very interesting results, as it was showed in this paper. There are many applications for these two technologies, such as: robotics, imaging processing and pattern recognizing.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Transfer Function</th>
<th>Number Representation</th>
<th>Machine</th>
<th>Frequency</th>
<th>Latency</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Sigmoid</td>
<td>Floating Point</td>
<td>Pentium III</td>
<td>550 MHz</td>
<td>---</td>
<td>92 µs</td>
</tr>
<tr>
<td>Software</td>
<td>Piecewise</td>
<td>Floating Point</td>
<td>Pentium III</td>
<td>550 MHz</td>
<td>---</td>
<td>74 µs</td>
</tr>
<tr>
<td>Software</td>
<td>Piecewise</td>
<td>Integer</td>
<td>Pentium III</td>
<td>550 MHz</td>
<td>---</td>
<td>23 µs</td>
</tr>
<tr>
<td>Hardware Without pipeline</td>
<td>Piecewise</td>
<td>Integer</td>
<td>Dedicated Circuit</td>
<td>---</td>
<td>---</td>
<td>32 ns</td>
</tr>
<tr>
<td>Hardware with a 2 stages pipeline</td>
<td>Piecewise</td>
<td>Integer</td>
<td>Dedicated Circuit</td>
<td>100 MHz</td>
<td>42 ns</td>
<td>20 ns</td>
</tr>
<tr>
<td>Hardware with a 4 stages pipeline</td>
<td>Piecewise</td>
<td>Integer</td>
<td>Dedicated Circuit</td>
<td>100 MHz</td>
<td>52 ns</td>
<td>10 ns</td>
</tr>
</tbody>
</table>

Table 1: Test Results
With the performance of the reconfigurable hardware, the use of the neural nets 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.

As a future work, a MLP hardware implementation for using in gesture recognition will be developed by using the same ideas proposed in this work.

References


