Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2007) 71–77
NEUROCOMPUTING
www.elsevier.com/locate/neucom

A high bit resolution FPGA implementation of a FNN with a new algorithm for the activation function

Pedro Ferreira[a,1], Pedro Ribeiro[a], Ana Antunes[a], Fernando Morgado Dias[b,c,*]

[a]*Departamento de Engenharia Electrotécnica, Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal, Campus do IPS, Estefanilha, 2914-508 Setúbal, Portugal*
[b]*Departamento de Matemática e Engenharias, Universidade da Madeira, Campus da Penteada, 9000-390 Funchal, Madeira, Portugal*
[c]*Centro de Ciências Matemáticas—CCM, Universidade da Madeira, Campus Universitário da Penteada, 9000-390 Funchal, Madeira, Portugal*

Available online 20 July 2007

Abstract

Several implementations of Feedforward Neural Networks have been reported in scientific papers. These implementations do not allow the direct use of off-line trained networks. Usually, the problem is the lower precision (compared to the software used for training) or modifications in the activation function. In the present work, a hardware solution called Artificial Neural Network Processor, using a FPGA, fits the requirements for a direct implementation of Feedforward Neural Networks, because of the high precision and accurate activation function that were obtained. The resulting hardware solution is tested with data from a real system to confirm that it can correctly implement the models prepared off-line with MATLAB.
© 2007 Elsevier B.V. All rights reserved.

Keywords: FPGA; Feedforward Neural Networks; Hardware; VHDL

1. Introduction

Artificial Neural Networks (ANN) became a common solution for a wide variety of problems in many fields, such as control and pattern recognition to name but a few. Within these solutions, a large share was developed using one type of ANN that only allows connections in the forward sense (that is, between one layer and the next one closer to the output), called Feedforward Neural Networks (FNN). It is therefore not surprising that some of the solutions have reached an implementation stage where specific hardware is considered to be a better solution than the most common implementation within a personal computer (PC) or workstation.

A number of reasons can be pointed out as the motivation for this:

- need for higher processing speed,
- reduced cost for each implementation,
- reliability.

The PC or the workstation being conventional von-Neumann architectures are not able to provide the high processing speed required by many applications. A hardware solution can also be less expensive and more reliable than a PC.

Also as pointed out by Ref. [7] "The greatest potential of neural networks remains in the high-speed processing that could be provided through massively parallel VLSI implementations".

Within the possible solutions: analog, digital or hybrid, as it is possible to find among the commercial implementations [5], the digital one has the following advantages:

- weight storage in memory,
- facility of integration with other applications,

*Corresponding author. Departamento de Matemática e Engenharias, Universidade da Madeira, Campus da Penteada, 9000-390 Funchal, Madeira, Portugal. Tel.: +351 291 705150/1; fax: +351 291 705199.
E-mail addresses: pedromdsf@netcabo.pt (P. Ferreira), pr-apinf@est.ips.pt (P. Ribeiro), aantunes@est.ips.pt (A. Antunes), morgado@uma.pt (F.M. Dias).
[1]Tel.: +351 265 790000; fax: +351 265 721869.

- facility to implement learning algorithms,
- exact within the number of bits of the operands and accumulators,
- low sensitivity to electric noise and temperature.

Considering the possible solutions for a digital implementation, there are still choices to make: full custom ASICs, Sea of Gates, FPGAs (to name only a few that have already been used in ANN implementation). The real choice is in fact quite reduced since for most of the applications, a specific hardware solution must be designed and therefore it is impossible to use the same design for many applications, which makes it economically unacceptable to use ASICs or Sea of Gates solutions.

This leads, in the case of a specific application, to a FPGA solution because the cost associated with only a few copies of the hardware is acceptable. In the literature, it is possible to verify that several solutions have already been tested in the FPGA context [1–3,6,8,10,11].

Nevertheless, the solutions that were found do not allow the direct use of the neural models that are prepared frequently with software (like MATLAB or specific software for ANN) within PCs or workstations.

All the solutions that the authors were able to verify within FNN present either a much lower precision when compared with these software solutions [1,6,8] or modifications in the activation function that make them unacceptable to use directly the weights previously prepared [6,10,11].

Filling this gap between software and hardware solutions, allowing the direct use of the weights is an important step in the development of the ANN field.

In the present work, a hardware solution called Artificial Neural Network Processor (ANNP), using a FPGA, developed to fit the above requirements for a specific application is presented.

Although the hardware was developed with a specific application in view, the following characteristics can be pointed out:

- Scalability, since the ANNP can be used for different network sizes.
- High precision, since the inputs and internal calculations are done with 32 bits in floating point notation.
- Accurate activation function, since the highest error allowed in the implemented activation function is of 2.18×10^{-5}.

2. Hardware implementation

There are three main problems that must be addressed when a hardware implementation with a FPGA is considered:

- Notation
- Activation function
- Device capabilities.

Notation is a key issue in the hardware implementation. It is almost consensual that floating point notation should be used if a high precision is seeked with a lower number of bits. The problem here is the complexity resulting from the necessary operations (multiplication, division, addition) with this notation.

The activation function is particularly difficult to implement in the case of the sigmoid functions. The direct implementation, for example, of the hyperbolic tangent would require an adder, multiplier and exponential.

The problems pointed out can be solved if enough hardware is available; this is why the device capabilities limit the possible solutions.

2.1. Notation

The notation chosen was 32 bits floating point according to the IEEE 754-1985 standard. Although it has been stated in Ref. [12] that "A few attempts have been made to implement ANNs in FPGA hardware with floating point weights. However, no successful implementation has been reported to date". and Nichols et al. [9] have concluded that "floating point precision is still not feasible in FPGA based ANNs", there were at least two applications reported: Ref. [1] which used floating point notation of 17 bits and Ref. [2], which used 24 bits.

The fixed-point notation would require a larger number of bits to obtain the same precision and maximum number to be represented and for this reason this option was discarded.

Other solutions include pulse stream arithmetic [8] or bitstream [3].

2.2. Activation function

In the present application, the activation function used was the hyperbolic tangent (Eq. (1)):

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

This equation can be re-arranged so that it has only one exponential but it still requires the operations of adding, dividing and the calculation of the exponential itself.

Understanding that the direct implementation is not suitable for the present solution, the hyperbolic tangent was studied in order to simplify its implementation.

As the objective of the present work was defined from the start, a maximum error could be set and a classical approach with a Look-Up-Table (LUT) was tested, but it was easily verified that this solution would be too expensive in hardware.

A new approach was then tested: piece-wise linear approximation. This corresponds to one of the classical solutions for the implementation of the activation function [12], but in this case with an important variation. While most of these solutions are done using only three linear sections to approximate the hyperbolic tangent, the