Artificial Neural Networks Processor - a Hardware Implementation using a FPGA

Pedro Ferreira, Pedro Ribeiro, Ana Antunes, Fernando Morgado Dias

[1] Escola Superior de Tecnologia de Setúbal do Instituto Politécnico de Setúbal, Departamento de Engenharia Electrotécnica, Campus do IPS, Estefanilha, 2914-508 Setúbal, Portugal Tel: +351 265 790000,
pedromdsf@netcabo.pt ; {pr-apinf, aantunes fmdias}@est.ips.pt

Abstract. Several implementations of Artificial Neural Networks have been reported in scientific papers. Nevertheless, these implementations do not allow the direct use of off-line trained networks because of the much lower precision when compared with the software solutions where they are prepared 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 resolution 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.

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. 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 and reliability.
Considering the possible solutions for a digital implementation the FPGA solution is the most interesting taking into account the balance performance/price.
In the literature it is possible to verify that several solutions have already been tested in the FPGA context. 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 present either a much lower precision when compared with these software solutions [1], [3] or modifications in the activation function that make them unacceptable to use directly the weights previously prepared [6], [7].
In the present work a hardware solution called Artificial Neural Network Processor (ANNP), using a FPGA, designed to fit the above requirements for a specific application is presented.

2 Hardware Implementation

The notation chosen was 32 bits floating point according to the IEEE 754-1985 standard. Although it has been stated in [2] 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 [5] have concluded that "floating point precision is still not feasible in FPGA based ANNs", there were at least two applications reported: [3] which used floating point notation of 17 bits and [4], which used 24 bits.

In the present work, the activation function used was the hyperbolic tangent.

As implementing the full functions is too expensive, the implementation was done using piece-wise linear approximation according to the following steps: Choose the first linear section; Compare this linear approximation with the hyperbolic tangent implemented in MATLAB in order to verify when the maximum allowed error is met; Start a new linear section; Repeat the operation until the region needed was fully represented.

This algorithm led to the representation of the hyperbolic tangent with 256 linear sections and provides a maximum error of 0,0000218 in values that are in the range of [-1,1] (function output). This can be compared with other solutions like the one using the Taylor series used in [3], which obtained an error of 0,51% and piecewise-linear approximation used in [4], which obtained 0,0254 of "standard deviation with respect to the analytic form".

It is worth to verify that to obtain this error with the classical LUT approach 18110 samples of the function were needed. These samples represented in the 32 bits notation used would require more than a single FPGA of the type used, just to represent the activation function.

The hardware platform used is the Cyclone EP1C20F324C7 FPGA from ALTERA, in the kit Cyclone SmartPack from Parallax.

The ANNP was developed using the VHDL language and implements the following components: 32 bits multiplier in floating point notation, 32 bits adder in floating point notation, activation function, memory blocks, 3 finite state machines, serial communication using RS-232 protocol.

The implementation of the activation function uses 3 ROMs of 256x32 bits, that is 24.576 bits, which are implemented in 6 blocks of 256x16 bits.

The complete implementation uses 235.008 bits, which are in fact 285.696 bits if the parity bits (and others that are used for special functions) are taken into account, of a total amount of 294.912 bits available, that means that 96,9% of the bits were used.

3 Test System and Results

The hardware implementation of the ANN was tested using several models of a system, which were previously prepared in MATLAB. This system is a reduced scale prototype kiln, which was working under measurement noise. For further details please see [8]. The comparison of the accuracy obtained in control simulations against the MATLAB software is summarized in table 1.

Table 1. Mean Square Error comparison between the MATLAB and FPGA results

Model	MATLAB Simulation			FPGA Results		
	Ramp	Square	Random	Ramp	Square	Random
M-1	$1{,}2146e^{-4}$	0,0146	$1{,}3343e^{-4}$	$1{,}2151e^{-4}$	0,0146	$1{,}3345e^{-4}$
M-2	$7{,}8682e^{-5}$	0,0040	$1{,}3445e^{-4}$	$7{,}8666e^{-5}$	0,0040	$1{,}3440e^{-4}$
M-3	$8{,}1468e^{-6}$	0,0069	$1{,}5426e^{-5}$	$8{,}1456e^{-6}$	0,0069	$1{,}5426e^{-5}$

4 Conclusions

This work proposes a hardware implementation of an ANN using a FPGA. The FPGA was chosen because of the lower prices for a single implementation.
The goal that was searched was to obtain a hardware solution that allowed the direct use of the weights, usually prepared in a software environment with a much higher resolution than the ones usually obtained in FPGAs implementation.
This objective was successfully accomplished as was shown by the control tests done with models of a real system, where the maximum error obtained between the MATLAB and the hardware solution, using the MSE as a measure was of $5e^{-8}$.
A new algorithm to apply with piece-wise linear approximation was also presented that allowed high resolution in the implementation of the hyperbolic tangent. This algorithm allows the definition of the maximum error that is acceptable and supplies the number and equations of the corresponding linear sections.

References

1. Lysaght, P., J. Stockwood, J. Law and D. Girma, "Artificial Neural Network Implementation on a Fine-Grained FPGA", in R. W. Hartenstein, M. Z. Servit (Eds.) Field-Programmable Logic, Springer Verlag, pp. 421-431, Czech Republic, 1994.
2. Zhu, J. H. and Peter Sutton, “FPGA implementations of neural networks – a survey of a decade of progress”, 13th International Conference on Field Programmable Logic and Applications, Lisbon, 2003.
3. Arroyo Leon, M. A., A. Ruiz Castro and R.R. Leal Ascenccio, "An artificial neural network on a field programmable gate array as a virtual sensor", Proceedings of The Third International Workshop on Design of Mixed-Mode Integrated Circuits and Applications, Puerto Vallarta, Mexico, pp. 114-117, 1999.
4. Ayala, J. L., A. G. Lomeña, M. López-Vallejo and A. Fernández, “Design of a Pipelined Hardware Architecture for Real-Time Neural Network Computations”, IEEE Midwest Symposium on Circuits and Systems, USA, 2002.
5. Nichols, K., M. Moussa and S. Areibi, “Feasibility of Floating-Point Arithmetic in FPGA based Artificial Neural Networks”, CAINE, San Diego California, pp:8-13, 2002.
6. Skrbek, M., “Fast neural network implementation”, Neural Network World, vol.9, nº5, pp.375–391, September 1999.
7. Wolf, D. F., R. A. F. Romero and E. Marques, “Using Embedded Processors in Hardware Models of Artificial Neural Networks”, V Simposio Brasileiro de automação inteligente, Brasil 2001
8. Dias, F. M. and A. M. Mota, ”Comparison between different Control Strategies using Neural Networks”, 9th Mediterranean Conference on Control and Automation, Dubrovnik, Croatia, 2001.