Experimental Platform for Accelerate the Training of ANNs with Genetic Algorithm and Embedded System on FPGA

Jorge Fe, R.J. Aliaga, and R. Gadea

Universidad Politécnica de Valencia, Spain
jorfe@posgrado.upv.es
http:www.upv.es

Abstract. When implementing an artificial neural networks (ANNs) will need to know the topology and initial weights of each synaptic connection. The calculation of both variables is much more expensive computationally. This paper presents a scalable experimental platform to accelerate the training of ANN, using genetic algorithms and embedded systems with hardware accelerators implemented in FPGA (Field Programmable Gate Array). Getting a 3x-4x acceleration compared with Intel Xeon Quad-Core 2.83 Ghz and 6x-7x compared to AMD Optetron Quad-Core 2354 2.2Ghz.

1 Introduction

The structure most known and used in artificial neural networks ANN[1] is the multilayer perceptron (MLP) and backPropagation algorithm (BP) is used for training, then, one of the problems when implementing an ANN training with MP structure and BP as training algorithm is to determine the optimal topology. For the determination of the optimal topology requires a long time of experimentation. In [2] different methods are detailed topology optencion as, trial and error, empirical or statistical methods, hybrid methods, constructive and or pruning algorithms y evolutionary strategie. Another problem that has a high computational cost is the optimal determination of the initial weights of ANN training, in [3] shown that a proper selection of the initial weights reduces training times.

Available literature known developments in ANN applied to specific problems in FPGA devices as [4] where it develops a coprocessor for convolutional neural networks, made a comparison of CPU acceleration. Of [5]provides a training platform for reconfigurable topology. The drawback to this application is that it has to synthesize each time you change the topology and then implement it in FPGA. Another application implements a fixed-topology ANN and Backpropagation algorithm for training [6]. In [7] where proposed BP algorithm implementation in FPGA, this is reconfigurable by software. In [8] have implemented in FPGA accelerator for online training.

According to the previous review proposes a systematic search of hidden neurons number and value of initial weights in ANN. The search for both variables, initial weights and number of neurons, is done with genetic algorithms (GA) [9]. The platform consists of eight embedded systems (IS) development boards implemented. The board has a Cyclone IV EP4CE115F29CN FPGA Altera [10]. Each of the ES using the coprocessor [11] called Neural Network Processor (NNP) having a training system in FPGA based algorithm Resilient Backpropagation (RBP). ES each communicates with a host PC via Ethernet, this PC with Matlab [12], Global Optimization Toolbox and Parallel Computing Toolbox, manages training with GA and parallel tasks executed in each of the ES. Getting a 6x-7x acceleration compared with Quad-Core AMD Optetron 2354 and 3x-4x Porcesador Intel Xeon Quad-Core E5540. The paper is organized as follows: Section 2, describes the platform. Section 3, the implemented software. Section 4, details the embedded system. Section 5, presents experimental results. Section 6, conclusions

2 Platform Training

The experimental platform is presented in Figure 1, composed of a host PC multi-core, and eight ES, connected to the network via Ethernet. Use this mode of communication as it allows the scalability, and transfer data effectively, and allows to quickly add ES. As the number of variables for the determination of the

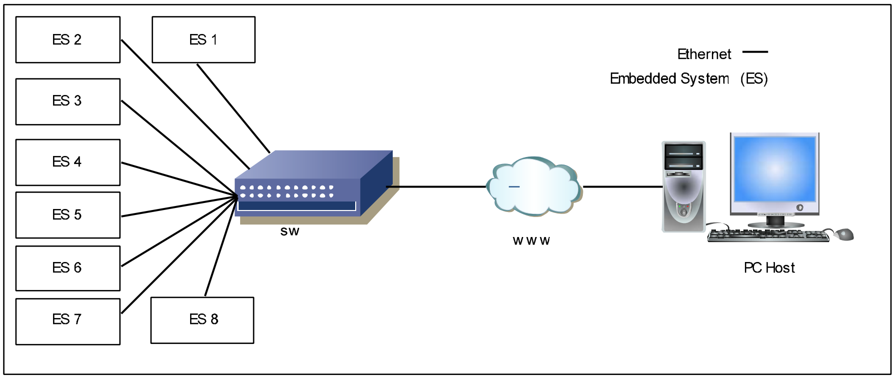

Fig. 1. ANN Platform training

topology or the initial weights vary, when used GA to the systematic search for both variables, generate an initial population of individuals n, the calculation of each individual is performed in each ES, allowing the calculation of eight individuals simultaneously.

3 Software

This section presents the execution flow of the software to achieve optimal topology and determination of the initial weights of an ANN. The software responsible for managing the training tasks are Matlab with Parallel Computing Toolbox (PCT), this allows you to manage the tasks running on each processor core and the execution of the tasks of higher computational cost run on SE. Besides using Global Optimization (GO), this provides different methods for finding solutions to different problems, one of them is with genetic algorithms are algorithms that are based on natural selection and the laws of genetics.

3.1 GA Tasks

In GA the Most Important parameters to configure are the number of individuals, generation number and the fitness function. These parameters determine the execution time of the GA in this aplication. The computation time depends of the individuals quantity to be computed in parallel, this parameter is configurable in the algorithm implemented by combining the PCT with GO, can be run from 1 to 8 individuals in parallel. Figure 2 shows a flowchart of the algorithm. Here is a brief description of each:

- Initialize the number of cores used in the host PC for ANN training.
- Transmission set of training vectors to SDRAM memory in ES.
- Configure the genetic algorithm (number of generations, numbers of individuals, use PCT, etc).
- Called the fitness function.
- The fitness function determines if there is a free core, then, generates an identifier and this identifier is assigned an IP address.
- The fitness function communicates with each of the SE through the IP address.
- So on until the end of the number of individuals and generations set in the GA is completed.
- The stopping method is by the number of generations.

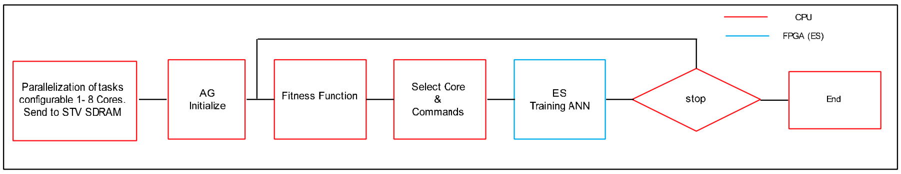

Fig. 2. Software flow diagram

4 Embedded System

Using Terasic development board DE2-115 to implement the embedded system, the ES is mainly composed of a real-time operating system RTOS MicroC/OS-II, processor NIOS II/f and NNP coprocessor, The NNP is a soft-core Single Instruction, Multiple Data Path (SIMD) in Figure 3 shows the hardware implemented.

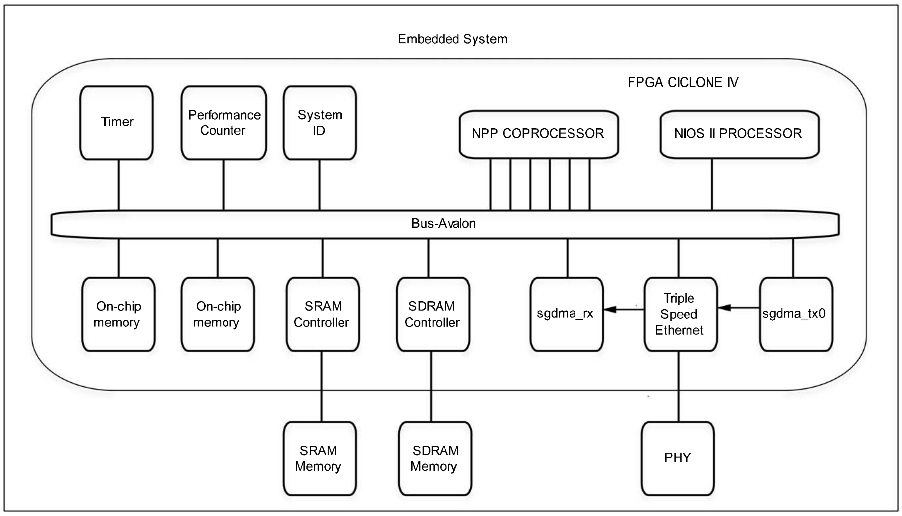

Fig. 3. Embedded System

4.1 Description

The Nios embedded processor is the master and controls the other system components, running from the program memory from External 2MB SSRAM to FPGA. The training vectors are stored in the external SDRAM. The remaining components are internal to the FPGA, besides the coprocessor are two DMAs that control the training vector transfer and instruction to the coprocessor, a memory of 16 KB which stores coprocessor instructions, and a memory of 4 KB (double the size of the memories of weight or gradients) for the variables needed for the algorithm RBP, these correspond to local increases and gradients of previous epoch. Figure 4 shows the connection diagram and internal architecture of the NNP. The processor NIOS II / f receives instructions through the Ethernet connection to the generation of a training. These instructions to configure the topology, if the generation of the initial weights is undertaken by the NIOS II, or should start with weights sent to memory via the Ethernet connection, the number of epoch, next, with all data received NIOS is responsible for generating the instructions for each new topology received, then, known [11] that the coprocessor is capable of processing a certain number n of parallel training vectors

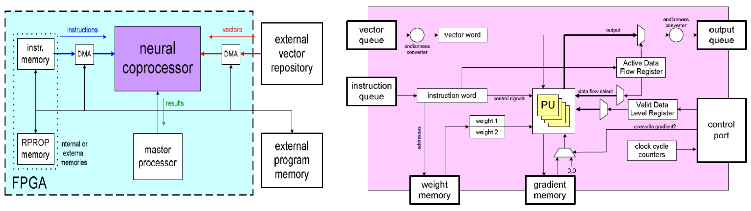

Fig. 4. Neural Network Coprocessor

and the gradients generated automatically accumulate, ultimately appearing in the memory of gradients at the end of the epoch. The reading and processing of each group of n vectors are getting by means of a set of instructions NNP.

5 Experimental Results

To show the validity of the platform as an accelerator expermiental to determine the initial weights of an ANN, has set a topology 6/5/3/1 with a sample set of 243,000 training vectors. Executed tests from one (ES) to eight (ES), also for each test also varied the number of epoch from 5 to 160 epoch. The result are show from figure 5 to figure 8.

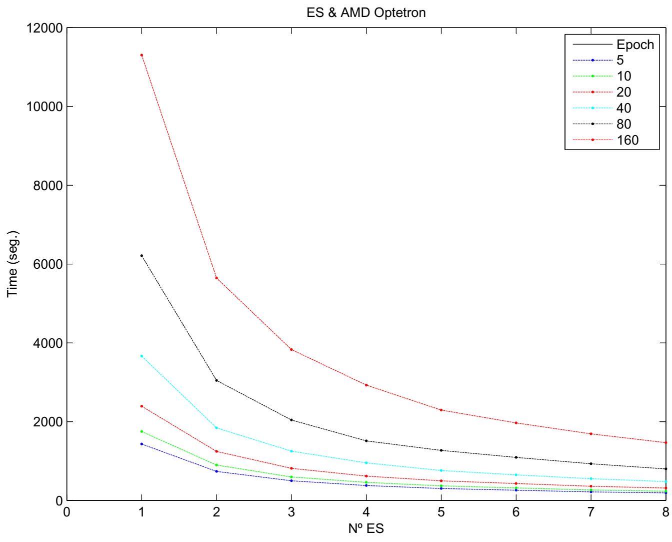

Fig. 5. Training, used from 1 ES to 8 ES, from 5 epoch to 160 epoch

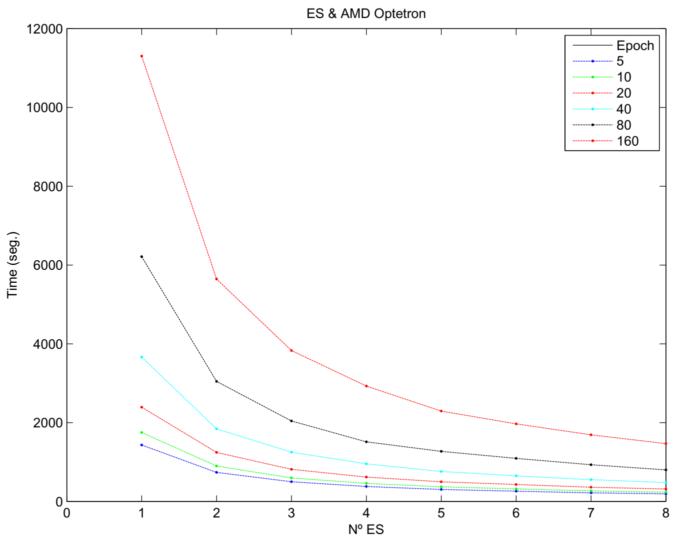

Fig. 6. Training, used from 1 ES to 8 ES, from 5 epoch to 160 epoch

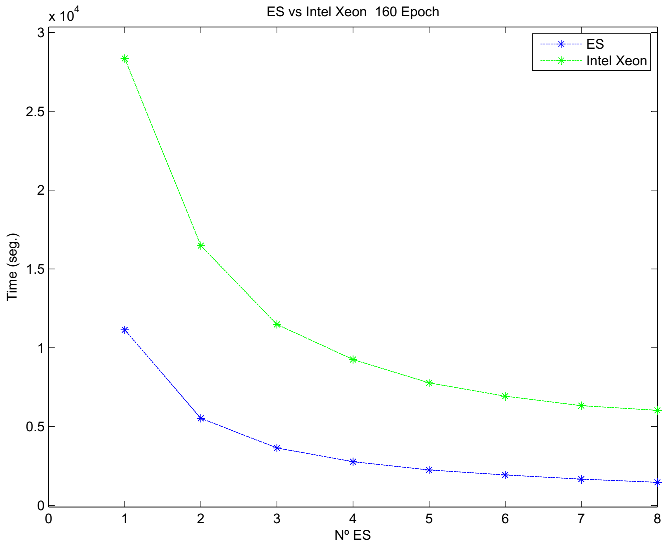

Fig. 7. ES vs CPU to 160 Epoch

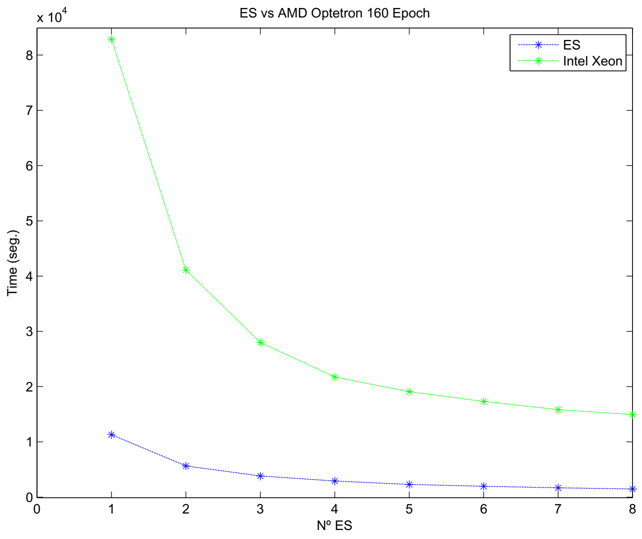

Fig. 8. ES vs CPU to 160 Epoch

6 Conclusions and Future Implementations

A scalable and reconfigurable platform for neural network training was proposed in this paper. It is scalable because it gives the possibility to connect different amounts of hardware accelerators (ES) and software reconfigurable topology. The method has been demostrated using the coprocessor for the computing task that have a high computational cost on ANN. achieving an acceleration of 3x-4x compared to Intel Xeon Quad-Core 2.83 Ghz and 6x-7x compared to AMD Optetron Quad-Core 2354 2.2Ghz.

In future applications will be implemented partial and dynamic reconfiguration. That is, several systems have loaded in Flash memory and to perform hardware reconfiguration. This will allow you to add multiple training algorithms and have more versatility on the platform. Also using the Distributed Computing Server Toolbox will not have the limitation of a PC, this Toolbox lets be scalable to several PC Host.

References

1. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice Hall (November 2008)
2. Curteanu, S., Cartwright, H.: Neural networks applied in chemistry. i. determination of the optimal topology of multilayer perceptron neural networks. Journal of Chemometrics 25(10), 527–549 (2011)
3. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights

4. Sankaradas, M., Jakkula, V., Cadambi, S., Chakradhar, S., Durdanovic, I., Cosatto, E., Graf, H.: A massively parallel coprocessor for convolutional neural networks. In: 20th IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2009, pp. 53–60 (July 2009)
5. Prado, R., Melo, J., Oliveira, J., Neto, A.: Fpga based implementation of a fuzzy neural network modular architecture for embedded systems. In: The 2012 International Joint Conference on Neural Networks, IJCNN, pp. 1–7 (June 2012)
6. Çavuşlu, M., Karakuzu, C., Şahin, S., Yakut, M.: Neural network training based on fpga with floating point number format and it's performance. Neural Computing and Applications 20, 195–202 (2011)
7. Wu, G.D., Zhu, Z.W., Lin, B.W.: Reconfigurable back propagation based neural network architecture. In: 2011 13th International Symposium on Integrated Circuits, ISIC, pp. 67–70 (December 2011)
8. Pinjare, S.L., Arun Kumar, M.: Article: Implementation of neural network back propagation training algorithm on fpga. International Journal of Computer Applications 52(6), 1–7 (2012)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
10. `http://www.altera.com`
11. Aliaga, R., Gadea, R., Colom, R., Cerda, J., Ferrando, N., Herrero, V.: A mixed hardware-software approach to flexible artificial neural network training on fpga. In: International Symposium on Systems, Architectures, Modeling, and Simulation, SAMOS 2009, pp. 1–8 (July 2009)
12. `http://www.matlab.com`