A PARAMETERIZABLE HANDEL-C NEURAL NETWORK IMPLEMENTATION FOR FPGA

Cherrad Benbouchama[1], Mohamed Tadjine[2]
[1] E.M.P, Bordj El Bahri, 16111 Alger, Algeria
[2]Ecole Nationale Polytechnique, El Harrach Alger, Algeria

Ahmed Bouridane[3]
[3]School of Electronics, Electrical Engineering and Computer Science, Queen's University

Keywords: Neural networks, FPGA, Parameterizable implementation, Handel-C.

Abstract: This paper shows the design possibility of a parameterizable implementation of neural multi-layer network on FPGA circuits (Field Programmable Gate Array) through the use of Handel-C language. The algorithm used for the training is the back-propagation. The tools of implementation and synthesis are the DK 4 of Celoxica and the ISE 6.3 of Xilinx. The targeted components are XCV2000 on Celoxica RC1000 board and XC2V1000 on RC200. The representation of the real numbers in fixed point was used for the data processing. The realization of the activation function is made with the approximate polynomial. A high level environment was designed in order to specify and introduce architecture parameters.

1 INTRODUCTION

The first ANNs implementation on FPGA was carried out by Cox and al within the framework of the GANGLION project (Cox and Blanz, 1992). It was applied for real time images segmentation. Other work followed since, for the implementation of various types' architectures of ANNs and various real data representations. The algorithm most used for multi-layer ANNs training is the back-propagation one. Eldredge had made a success, in 1994, of the first implementation of this algorithm on the RRANN platform (*Runtime Reconfiguration Artificial Neural Network*) built around the XC3090 FPGA (Eldredge and Hutchings, 1994). Ferrucci and Martin have designed the ACME multi-FPGA platform *(Adaptive Connectionist Model Emulator*) which is composed of fourteen (14) *Xilinx* XC4010 FPGAs (Ferrucci 1994) (Martin, 1994). Ossoing, in 1996, had also implemented on FPGA the back-propagation algorithm. It had implemented the architecture [3,3,1] on four (4) Xilinix XC4013 FPGAs and one Xilinix XC4005 (Ossoinig and al, 1996). Beuchat & al., in 1998, were heavily influenced by Eldredge work. They developed the RENCO platform containing four (4) FPGAs controlled by microprocessor. It was successfully applied to hand-written characters recognition (Beuchat and al, 1998). Pandya, in 2005, had implemented on FPGA the back-propagation algorithm with Handel-C language and had worked out a partially parallel architecture and another completely parallel, which proved, respectively, twice and four times faster than a sequential architecture (Pandya, 2005). However, the necessary time for the ANNs implementation on FPGAs is of about months (Benbouchama and al, 2007). Moreover, the programming of these circuits requires the mastery of specific languages, which reduces considerably their use on a large scale. It would be then interesting to carry out a graphical environment which would be dedicated for users not necessarily initiated to FPGAs programming.

In this paper, we are interested in the design of a parameterizable tool that generates a neural multi-layer network implementation on FPGAs through the use of Handel-C language. This work aims at the realization of a high-level environment making it possible, on the one hand, to ensure the interfacing between the user and an FPGA board, and on the other hand, to generate automatically configurations dedicated to the ANNs. The algorithm used for the training is the back-propagation, and the tools of implementation and synthesis are the DK 4 of Celoxica and the ISE 6.3 of Xilinx.

2 TECHNICAL PRINCIPLE

2.1 Handel-C Language

Handel-C is perhaps the most popular high-level language currently available for hardware specification (Stöcklein and Bhig, 2002). Its syntax is based on the C language making it easily adopted by traditional software engineers. The benefits of rapid hardware development and simplicity of specification often come at a price. In comparison with traditional hardware description languages such as VHDL (Ashenden, 2002), Handel-C and its compiler produce hardware that consumes more FPGA area (Hopf, 2003) and is often slower in performance.

2.2 A Parameterizable Implementation

A parameterizable implementation is that in which a user would have the possibility of changing the application parameters without having to modify the hardware configuration of the FPGA. The implementation will be flexible.

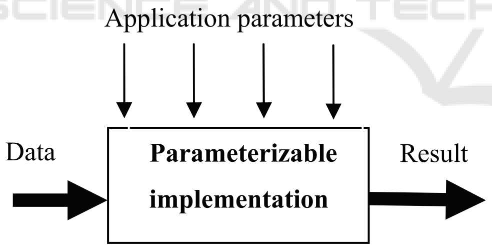


Figure 1: A parameterizable implementation.

In a parameterizable ANNs implementation (figure 2), the inputs are, in addition to the maximum of epochs and the desired error, different data concerning the architecture and the training parameters.

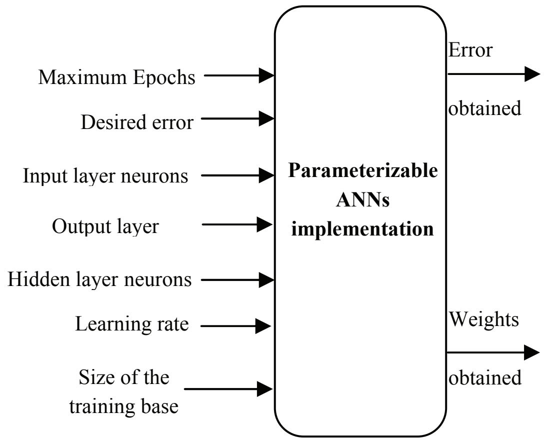


Figure 2: Parameterizable ANNs implementation.

The development of this parameterizable ANNs implementation is made with handel-C language which, contrary to VHDL, makes it possible to design completely parameterizable applications.

2.3 Automatic Generation of Configurations

It consists in developing an automatic generator of configurations starting from the host computer towards the FPGA board (figure 3).

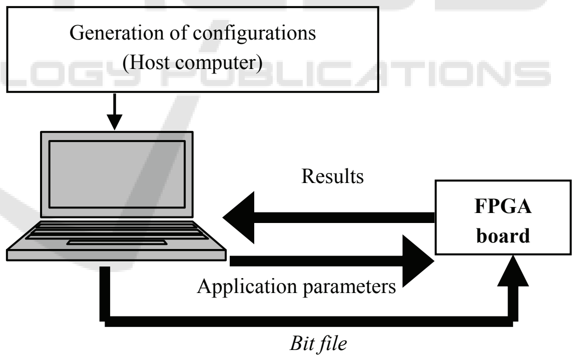


Figure 3: Principle of the software platform.

The generator of configurations ensures the following tasks:

- The synchronization between the host computer and the FPGA board.
- The configuration of the FPGA circuit by the bit file.
- The data transfer between the FPGA board and the host computer.

2.4 Graphical Interface

From these graphical menus, the user is invited to design his application. Thus, he can specify the different parameters of the application. This system can bring many tools for the implementation design and control. The advantage resides in the facility of the approach: there is no language, it requires only the use of graphical menus (control buttons, edition zone…). The graphical environment carried out with *Microsoft visual C++* can be composed of one or several windows.

3 IMPLEMENTATION RESULTS

To demonstrate the findings, the parameterizable implementation was realized using the XCV2000 of the FPGA family VirtexII.

3.1 First Case

Input layer neurons number ≤ 2
Output layer neurons number ≤ 2
Hidden layer neurons number ≤ 15
Hidden layer number ≤ 3

Table 1: First case.

Resources	Used	Available	Utilization
Slices	17019	19200	88 %
LUTs	31320	38400	81 %
IOBs	88	404	21 %
Block RAMs	320	160	**200 %**

3.2 Second Case

Input layer neurons number ≤ 2
Output layer neurons number ≤ 2
Hidden layer neurons number ≤ 10
Hidden layer number ≤ 1

Table 2: Second case.

Resources	Used	Available	Utilization
Slices	13101	19200	68 %
LUTs	24978	38400	65 %
IOBs	88	404	21 %
Block RAMs	180	160	**112 %**

3.3 Third Case

Input layer neurons number = 1
Output layer neurons number = 1
Hidden layer neurons number ≤ 10
Hidden layer number ≤ 1

Table 3: Third case.

Resources	Used	Available	Utilization
Slices	12144	19200	63 %
LUTs	23223	38400	60 %
IOBs	88	404	21 %
Block RAMs	108	160	67 %

From these results, it can be stated that the size of the parameterizable implementation to be realized depends on the targeted FPGA.

4 EXPERIMENTAL EVALUATION: THE POSITION CONTROL OF A DC MOTOR

We test the high level-environment on the implementation of a neural controller which improves an existing linear controller in order to position control a DC motor (Fig. 4). This approach is called "feed-back error learning" and is based on the use of an existing regulator to approximate as an unknown function (Benbouchama and al, 2007).

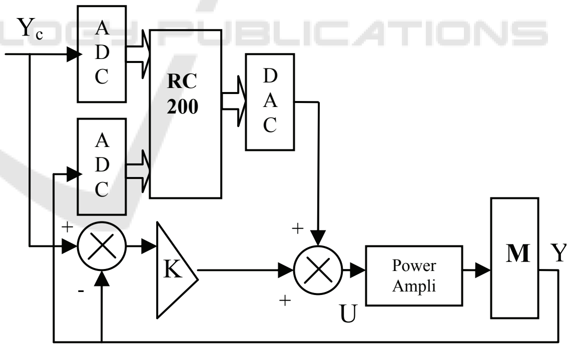


Figure 4: The position control of a DC motor.

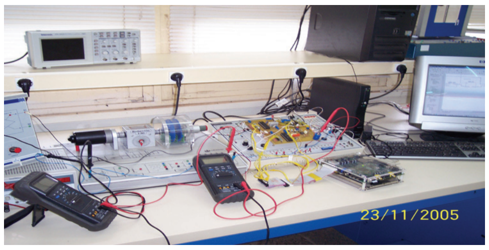
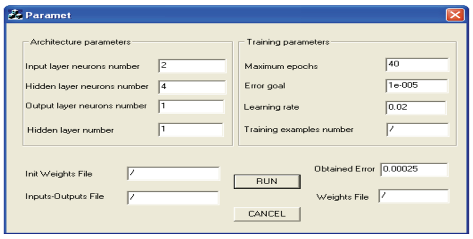


Figure 5: Hardware view.

In this experiment an on-chip learning type of the neural controller was used.
The graphical interface used to specify the different parameters of the neural controller is as follows:


Paramet

Architecture parameters
Input layer neurons number: 2
Hidden layer neurons number: 4
Output layer neurons number: 1
Hidden layer number: 1

Training parameters
Maximum epochs: 40
Error goal: 1e-005
Learning rate: 0.02
Training examples number: /

Init Weights File: /
Inputs-Outputs File: /
Obtained Error: 0.00025
Weights File: /

Figure 6: The graphical interface.

The implementation results of the parameterizable ANNs implementation for this experimental evaluation are as follows:

Table 4: Implementation results.

Resources	Used	Available	Utilization
Slices	2748	5120	53 %
LUTs	4926	10240	48 %
IOBs	20	324	6 %

Once the learning phase was completed we have tested the behaviour of the neural controller with a square input signal: 0° - 45°.

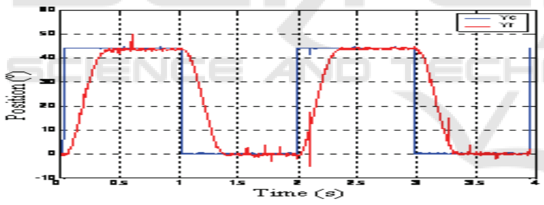


Figure 7: The behaviour of the neural controller.

The results indicate the validity of the high-level environment used for the ANNs implementation on FPGAs.

5 CONCLUSIONS

This paper outlines a means that was created to facilitate and accelerate the ANNs implementation on FPGAs. A parameterizable tool was designed to generate a neural multi-layer network implementation through the use of Handel-C language. This tool was destined to be used by the high level environment, which is presented, at the user, as a graphical interface with menus. The advantage which it offers resides in the facility of the approach: there is no language and it requires only the use of the graphical menus.
To be able to implement significant neural networks architectures with this high level environment, we must use a board with an FPGA circuit which is not limited in resources.
Finally, experimental evaluation setup has been developed to demonstrate the validity of the high-level environment for the ANNs implementation on FPGAs.

REFERENCES

Cox, C.E. and E. Blanz, GangLion, “ a fast field - programmable gate array implementation of a connectionist classifier ”, *IEEE Journal of Solid-State Circuits, 1992. 28(3): p. 288-299.*

J. G. Eldredge and B. L. Hutchings, “ RRANN: A Hardware Implementation of the Backpropagation Algorithm Using Reconfigurable FPGAs ”, *In IEEE World Conference on Computational Intelligence, Orlando, FL, 1994.*

A. T. Ferrucci, “ A Field-Programmable Gate Array Implementation of Self-Adapting and Scalable Connectionist Network ”, *Mars 1994.*

M. Martin, *“A reconfigurable hardware accelerator for back-propagation connectionist classifiers”*, Masters thesis, University of California, Santa Cruz, 1994.

H. Ossoinig, E. Reisinger, C. Steger, and R. Weiss, “Design and FPGA implementation of a neural network ”, *In Proc. 7th Int. Conf. on Signal Processing Applications and Technology, pp. 939943, Orlando, Florida, 1996.*

J. Beuchat, J. Haenni, and E.Sanchez, *“ Hardware Reconfigurable Neural Networks, In Parallel and Distributed Processing ”*, IPPS/SPDP, pp. 91 98, Springer-Verlag, 1998.

V. Pandya, *“ A Handel-C implementation of thebackpropagation algorithm on field programmable gate arrays ”*, Master thesis. Faculty of Graduate Studies of the University of Guelph, Canada, December 2005.

C. Benbouchama and al, *“ The FPGA Neural Networks Implementation for a Real Time Control ”*, Archives of Control Sciences (A.C.S.), Vol. 17(LIII), n. 1, pp. 527, 2007.

T. Stöcklein and J. Bhig, *“ Handel-C an effective method for designing FPGAs (and ASICs) ”*, Georg Simon-Ohm Fachhochschule, NÜRNBERG 2002.

P. J. Ashenden, *“ The Designer's Guide To VHDL ”*, second edition, San Diego: Morgan Kanfmann, 2002.

J. Hopf, “ Comparing the Bitstreams of Applications Specified in Hardware Join Java and HandelC ”, *2nd IEEE International Conference on Field Programmable Technology, Tokyo, Japan, 2003.*

Celoxica: www.Celoxica.com