Volume 4, Issue 2, February 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Implementation of FPGA based Multilayer Perceptron using VHDL

Prashant D.Deotale, Lalit Dole

G.H. Raisoni College of Engineering

Nagpur, India

Abstract—This paper represents the development and implementation of a generalized back propagation multilayer perceptron (MLP) architecture described in VLSI hardware description language (VHDL. A Field Programmable Gate Array based hardware design of multilayer perceptron (MLP) as a realization of artificial neural network is presented on FPGA hardware platform. Such hardware implementation solutions are easy to implement by using training of neurons in MLP. MLP work describes a platform that offers a high degree of parameterization, while maintaining generalized network design with performance comparable to other hardware-based MLP implementations. So the main purpose of this paper lies in studying the effect of changing both the no of hidden layers of multilayer perceptron (MLPs) and the no of processing elements that exist in the hidden layers .After formation of such a Multilayer perceptron(MLP) and changing the number of hidden layers, we found that with increasing the number of processing elements in the hidden layers. We reach to an optional output results with respect to the experimental one.

Keywords—VHDL,MLP,Processing elements, neurons.

I. INTRODUCTION

Artificial Neural Network present an unconventional computational model characterized by densely interconnected simple adaptive nodes. From this model stem, several desirable traits uncommon in traditional computational models; most notably, an MLP ability to learn and generalize upon being provided examples. Given these traits, an MPL is well suited for a range of problems that are challenging for other computational models like pattern recognition, prediction, or optimization [1]–[4]. An MLP ability to learn and solve problems relies in part on the structural characteristics of that network. Those characteristics include the number of layers in a network, the number of neurons per layer, and the activation functions of those neurons etc. There remains a lack of a reliable means for determining the optimal set of network characteristics for a given application. Numerous implementations of MlP already exist [5]–[8], but most of them being in software on sequential processors [2]. Software implementations can be quickly constructed, adapted, and tested for a wide range of applications. In this work, a multilayer perceptron (MLP) is introduced, which fulfills the following requirements. The hardware solution should be application independent, which makes it usable for several problems during runtime. This can be achieved by high flexibility. High flexibility can be guaranteed by a highly parameterizable design. The neurons (also called perceptrons in this case) transfer function can be freely chosen, which makes it adaptable to different scenarios. Additionally, an easy integration of the MLP[9]-[12] . The training of the MLP is done directly in software using Xilinx 13.4. A concept is defined and realized to comfortably interact with the MLP. Additionally, the resulting design is investigated in terms of the size and performance as well as compared with the trend of available hardware resources of different FPGAs over time. Below, the following main contributions are briefly described.

- Brief description of the basic concept is given and required parameters of an ANN are investigated [5].
- Design of a parameterizable MLP implementation is described [6].
- Concept for configuration and interaction with the MLP is presented [8].
- The hardware utilization and performance of the developed MLP are presented[10].

II. BASICS AND DESIGN CONCEPT

There are many concepts for designing MLPs e.g. fully parallelized, serialized, or even a mixed architecture. However, MLPs are well known, established, and suited for a realization in hardware because of their regular structure. A three layer MLP is depicted in Figure 1. An MLP consists of several artificial neurons. These neurons are represented by perceptions [1].A this layers MLP is realized and each layer contains a set of neurons. The first layer is the input layer and represents the input values. The hidden layer is the second layer and is fully meshed to the previous input layer. The output layer generates the outputs and receives the values from the hidden layer. To train the MLP, weight factors of the connections have to be set depending on a given use case. The determination of the weight factors is called training.

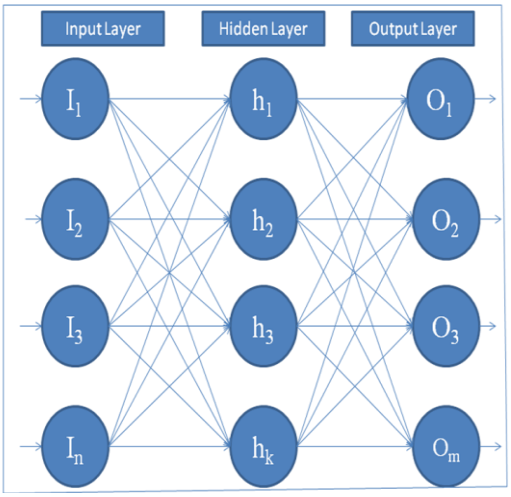

Fig 1. MLP architecture

A. Artificial Neuron

An artificial neuron is the processing unit of the neural network architecture. The neurons' structure includes:

a. An entry set that receive neurons' input signals[17];
b. A synaptic set whose intensity is represented by an associated weight[18];
c. An activation function that compares entries and their synaptic with the function threshold to give the neuron's output [19].

In the Fig. 2 each Wi represents the weights associated with each input Xi and Q is the activation function. The result of the synaptic is given by the sum of products of the entries vector by the weight vector and the output by the computation of Q(u). The most used activation functions are:

a) Step function
Q(u) = 1 if u > 0, Q(u) = 0, otherwise
b) Ramp function
Q(u) = max{O.O, min{l.0, u + 0.5}}
c) Sigmoid function
Q(u) = a I{ 1 + exp(-bu) }

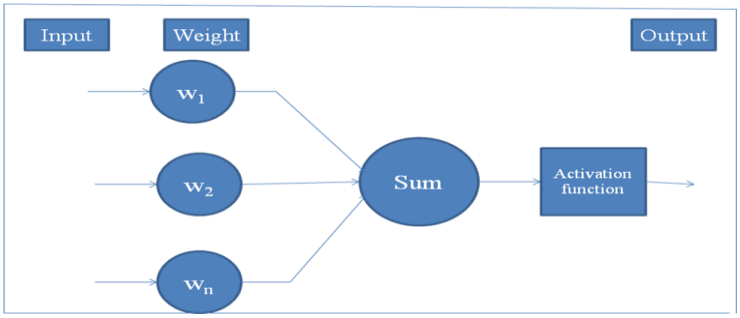

Fig 2. Neuron Architecture

B. Previous work

Many MLP have already been implemented on FPGAs. The vast majority are static implementations for specific offline applications without learning capability [11]-[14]. In these cases, the purpose of using an FPGA is generally to gain performance advantages through dedicated hardware and parallelism. Far fewer are examples of FPGA-based MLP that make use of the reconfigurability of FPGAs. Flexible Adaptable Size Topology (FAST) [15] is an FPGA based MLP that utilizes runtime reconfiguration to dynamically change its size. In this way, FAST is able to skirt the problem of determining a valid network topology for the given application. Runtime reconfiguration is achieved by initially mapping all possible connections and components on the FPGA, then only activating the necessary connections and components once they are needed. FAST is an adaptation of a Kohonen type neural network and has a significantly different architecture than our multilayer perceptron (MLP) network.

C. Platform

We are focusing on development platform like Xilinx navigator 13.5 which is implemented on FPGA [2]-[5]. While our design is not directed exclusively at this platform and is designed to be portable across multiple FPGA platforms, we will mention some of the characteristics of the Xilinx 13.5 important to the design and performance of our system. This model of the Xilinx navigator 13.5 contains 4080 configurable logic blocks (CLBs), the basic logical units in Xilinx FPGAs. Each CLB holds eight logic function generators (in lookup tables), eight storage elements, a number of

multiplexers, and carry logic. Relative to the time in which this paper is written, this is considered a large FPGA[12]-[15]; large enough to test a range of online neural networks of varying size, and likely too large and costly to be considered for most commercial applications.

III. LEARNING OF PROCESSING ELEMENT IN MLP

In reality, during the learning process, the network adjusts its parameters, the synaptic weights, in response to a stimulus input ***so*** that its actual output response converges to the desired output. At this level the synaptic weights of each processing unit are dynamically modified to reach a defined error level according to an optimization criteria called learning algorithm (for example back propagation algorithm) in order to identify the best architecture with a given number of cells for a allow specific problem[13]. When the actual output response is the same as the desired one, the network has completed the learning phase. Then the optimized structure is known [20]-[21].

a. number of inputs
b. number of outputs
c. number of layers
d. synaptic weights
e. transfer function/squashing function

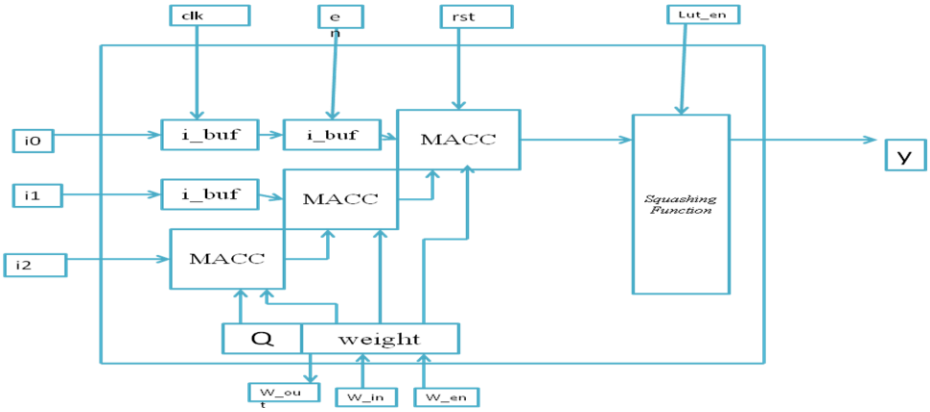

Fig. Processing element in MLP

VI. FPGA IMPLEMENTATION

A. *Synthesis Tool*

Actually, the synthesis tools allow the use of FPGA resources with schematics entrance as well as an algorithmic one[9]-[12]. The algorithmic design is written with VHDL (Very High-speed Integrated Circuits Hardware Description Language)[5]. This language permits the design of complex circuits with a structural description (data flow type) or a behavioral one. The synthesis software (Viewsynthesis of ViewLogic Company) leads from a VHDL program, and after Compilation, to an XNF (Xilinx Netlist Format) file used by the Xilinx tool, and the corresponding schematics by means of primitives and X-Bloc components with a logic optimization. The functional simulation insures of the functionality of the design before the routing phase of the FPGA (fig 3)

B. *Design Architecture*

Our design approach is based on the separation of simple modular functional components and more complex intelligent control oriented components. The functional units consist of signal processing operations (e.g. multipliers, adders, squashing function realizations, etc.) and storage components (e.g. RAM containing weights values, input buffers, etc.). Control components consist of state machines [16] generated to match the needs of the network as configured. During design elaboration, functional components matching the provided parameters are automatically generated and connected, and the state machines of control components are tuned to match the given architecture [18].

Fig 3.The design flow of the synthesis

IV. RESULT AND CONCLUSION

The use of run-time reconfiguration of the FPGA of the Xilinx company will allow us to develop and implement some different algorithms or developed application on the same circuit but not useful at the same time. Moreover the implementation of the different blocks shown above permits the evaluation and the validation of the architecture proposed for a specific problem.

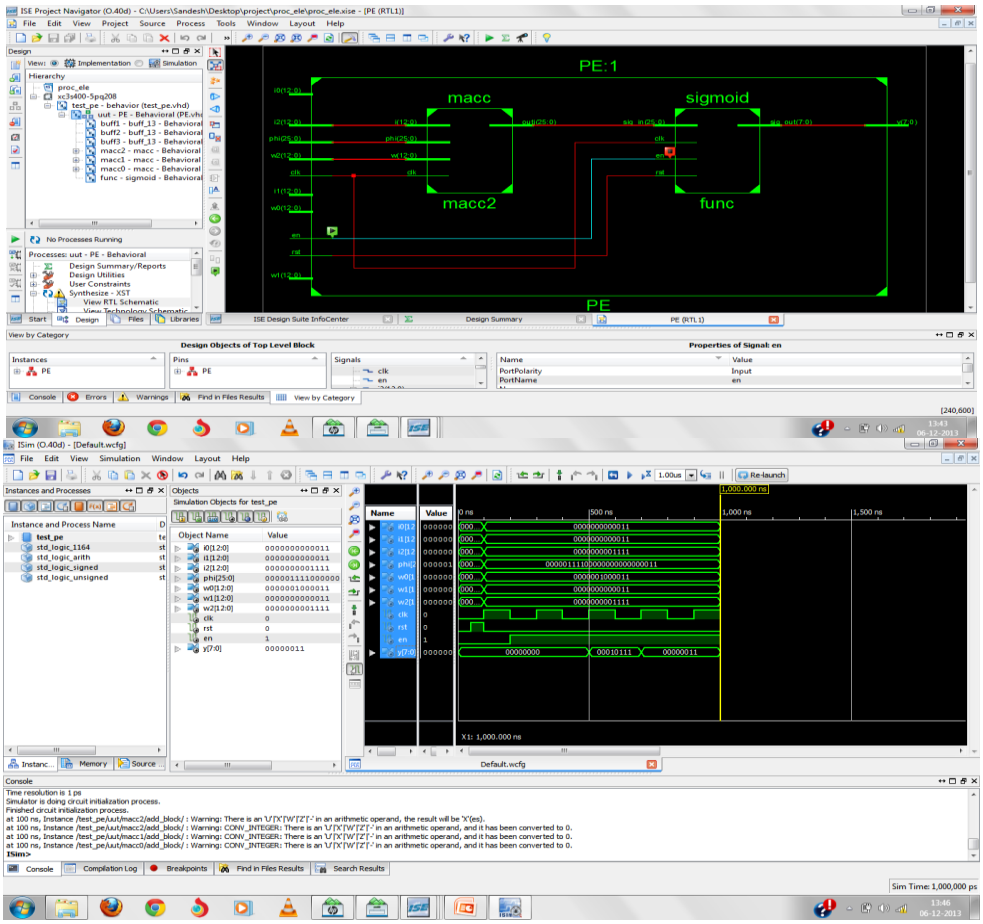

Fig 4.Simulation output of PEs

V. ACKNOWLEDGMENT

I seize this opportunity to thank all the people who directly or indirectly helping me in completion of my project. I take immense pride in paying gratitude to my project guide **Prof.*Lalit Dole*** who always appreciating me for innovative ideas and giving me a chance to implement them.

I am indebted to our respected **Principal Dr.P.R.Bajaj, G.H.Raisoni College of Engineering** , for his time to time guidance and motivation for better compliance of the project.

I am grateful to **Dr. (Mrs.) L.G.Malik, HOD, Dept. of Computer Science and Engineering** and **Dr.D.V.Padole, M. Tech, Coordinator**, who aide us with well equipped Computer Laboratory facilities.

I shall be failing in my attempt if the entire G.H.R.C.E. Family is not remembering for providing all the required resource and support.

REFERENCES

[1] M. Paliwal and U. A. Kumar, "Neural networks and statistical techniques: A review of applications," *Expert Systems With Applications*, vol. 36, pp. 2–17, 2009.

[2] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. V. Campenhout, "Compact hardware liquid state machines on FPGA for real-time speech recognition," *Neural Networks*, vol. 21, no. 2–3, pp. 511–523, 2008.

[3] A. Ukil, *Intelligent Systems and Signal Processing in Power Engineering*, 1st ed. New York: Springer, 2007.

[4] *"MATLAB Neural Network Toolbox User Guide,"* ver. 5.1, The Math Works Inc., Natick, MA, 2006.

[5] J. Zhu and P. Sutton, "FPGA implementations of neural networks—A survey of a decade of progress," *Lecture Notes in Computer Science*, vol. 2778/2003, pp. 1062–1066, 2003.

[6] I. A. Basheer and M. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application," *J. Microbio. Methods*, vol. 43, pp. 3–31, Dec. 2000.

[7] A. R. Ormondi and J. Rajapakse, "Neural networks in FPGAs," in *Proc. Int. Conf. Neural Inform. Process.* 2002, vol. 2, pp. 954–959.

[8] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: Applications in industry, business and science," *Commun. ACM*, vol. 37, no. 3, pp. 93–105, 1994.

[9] J.R.Armstrong. F.Gai1 Gray. Structured Logic Design lithe VHDL .*PTR Prentice* Hall.

[10] P. Ienne, T. Cornu, and G. Kuhn, "Special-purpose digital hardware for neural networks: An architectural survey," *J. VLSI Signal Process.*, vol. 13, no. 1, pp. 5–25, 1996.

[11] B. J. A. Kroese and P. van der Smagt, *An Introduction to Neural Networks*, 4th ed. Amsterdam, the Netherlands: The University of Amsterdam, Sep. 1991.

[12] XILINX.*TheprogrammableGate.IrrgvData*Book.1995

[13] C. Mead and M. Mahowald, "A silicon model of early visual pro- cessing," *Neural Networks*, vol. 1, pp. 91–97, 1988.

[14] J. B. Theeten, M. Duranton, N. Mauduit, and J. A. Sirat, "The LNeuro chip: A digital VLSI with on-chip learning mechanism," in *Proc. Int. Conf. Neural Networks*, 1990, vol. 1, pp. 593–596.

[15] J. Liu and D. Liang, "A survey of FPGA-based hardware implementation of ANNs," in *Proc. Int. Conf. Neural Networks Brain*, 2005, vol. 2, pp. 915–918.

[16] A. Rosado-Munoz, E. Soria-Olivas, L. Gomez-Chova, and J. V. Frances, "An IP core and GUI for implementing multilayer perceptron with a fuzzy activation function on configurable logic devices," *J.*

[17] E. Sanchez, "FPGA implementation of an adaptable-size neural. Work," in *Proc. Int. Conf. ANN*, 1996, vol. 1112, pp. 383–388.

[18] *"Virtex-5 FPGA User Guide,"* Xilinx, 2007. 1 J.W.Gardncr. H.V.Shurmer and T.T.Tan. Application of an electronic nose *to* the discrimination of coffees. *Sen.cor.y and ..Icluatnr.s U.* 6 (1992) 7 1-75. 1314.

[19] J .Mizsei. V .Lantto, Air pollution monitoring with a semiconductor gas sensor array system, *Sensors and Actuators B.* 6 (1992) 223-227.

[20] Niebling. Identification of gases with classical pattern-recognition methods and artificial neural networks, *Sensors and .Ictuators* B, 18-19 (1994) 259- 263.

[21] M.Hubin. Y.Taright. S.Lee, Multisensor Smart Microsystems for the Supervision of Accidental Atmospheric Pollution, *8th International Congress for Sensor , Trnnshicers C% systems, .\'urnberg,* (1997) Vol 2. 129-133.

