Proceedings of ICSP '98

FPGA Implementation of a Multilayer Perceptron Neural Network using VHDL

Yamina TARIGHT, Michel HUBIN

Laboratoire Perception Systèmes et Information (PSI-LCIA),

INSA de Rouen, Place E Blondel BP 08

F-76131 Mont Saint Aignan, France

Abstract:

Our aim is how to use the run-time reconfiguration ability of SRAM-based FPGA circuits in order to implement artificial multilayer neural networks which are needed to process, in real-time, non linear data obtained from a set of microchemical sensors. So we propose, in this paper, an innovative alternative to ASICs and/or multiprocessors based architectures.

1. Introduction:

There are many real time applications, which require a low-cost, miniaturized and automatic system used for classification and recognition. Frequently such a device is organized around several sensors providing complex data that only an artificial neural network will be able to interpret.

In this frame, the development of electronic noses, which operate upon a similar, but simplified, principle as the human olfactory system, represents an alternative to the classical physico-chemical analytical technique. It has many potential applications, as it allows quantitative and qualitative analysis of gaseous mixtures in different domains such as food-processing industry, security and air pollution evaluation. The prototypes of electronic noses seen in the literature [1] [2] are composed of a set of a temperature sensor, a humidity sensor and chemical microsensors (semiconductor ones based upon tin oxide sensing layer). These sensors present some advantages such as low cost, sensitivity and response time but their lack of selectivity [3] is resolved by means of a neural network computing involving two steps: learning and recognition.

The ANN used in these prototypes are always on parallel algorithm software based and simulated on computers.

Because of the real time, miniaturization and power constraints imposed by some specific applications like the supervision of the atmospheric pollution in urban sites [4] we consider a hardware implementation of such ANNs. Very few neural devices are referring to a hardware implementation as parallel neural structures are very expensive to realize in terms of design time and silicon surface used in ASIC technology and they also present an evident economic risk because of the fixed architectural concept of such a type of component. An alternative is possible: the use of SRAM field programmable gate arrays (FPGA) which allows the design of cost effective circuits using the dynamic reconfiguration ability of such components.

In this paper, using our experience in the design of FPGA, we will describe our original approach and the proposed solution for the hardware implementation of a multilayer perceptron (MLP) specifically dedicated to the real time application following of the air pollution [4]

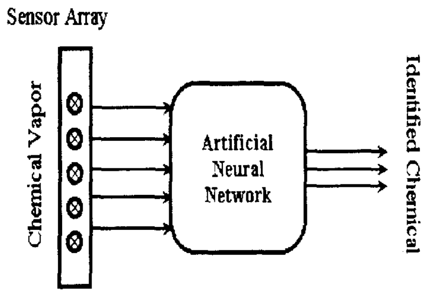

Figure 1. Schematic diagram of an Electronic nose

2. MLP detection and classification

A. algorithm

The MLP consists of various layers: an input and output ones between which lie one or several hidden ones whose outputs are not observable. These layers are based upon some processing unit (neurons) interconnected by means of feed-forward pondered links (figure 2).

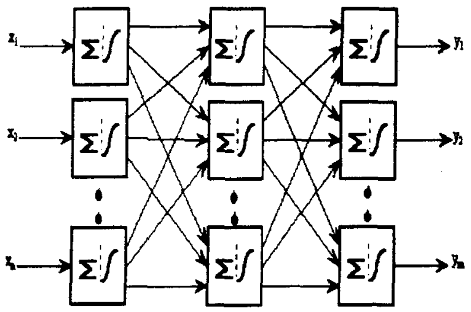

Figure 2. Multilayer perceptron neural network

All these processing units carry out the same operation (figure 3): i.e. the sum of their weighed inputs (equation 1). Then they apply the result to a non-linear function named activation function and generally based upon the sigmoid function (equation 2).

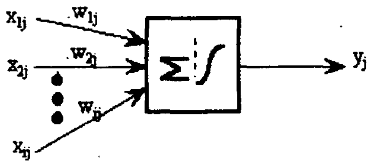

Figure 3. Structure of the neuron

$$y_j = f\left[\left(\sum_i w_{ij} \cdot x_{ij}\right) - b_j\right] \qquad (1)$$

$$f(x) = \frac{1}{1 + e - x} \qquad (2)$$

where y_j is the output of the processing unit j
w_{ij} the synaptic weight coefficient of the i-th input of the processing unit j
b_j is the bias

The MLP processing is based on two sequential steps: learning and recognition which are independent, and consequently could be treated separately.

- **learning**

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 backpropagation algorithm) in order to identify the best architecture with a given number of cells for a allow specific problem. 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, i.e.:

- number of inputs
- number of outputs
- number of layers
- synaptic weights
- transfer function

- **recognition:**

The MLP doesn't give only the desired outputs corresponding to known inputs, but also the main plausible outputs for any set of inputs using the equation (1) with the w_{ij} evaluated during the learning phase.

B. Looked-in Process:

For our system, the learning phase is obtained by means of an algorithmic simulation made on a computer in order to identify the synaptic weights and the best architecture of the network. Then the hardware implementation may be considered.
We must take into account some hardware constraints. The optimized software solution leads generally to build extremely complex circuits depending on the size of the data. In order to develop realistic devices with a convenient accuracy and a reasonable hardware size, one must redefine the resolution of the coefficients and of the calculus. So the optimization of the architecture and the reduction of the number of configurable logic blocks (CLB) used in the FPGA circuit implicate the best accuracy obtained during the learning phase.
On account of all these considerations, we determine the arithmetic type used (floating point or not) and the size of the processed data (in bits).
A good compromise between accuracy and complexity of the processing units allows to obtain a low cost effective device (in terms of CLB in FPGA circuits, and consequently in term of silicon area in derivative ASICs.

3. FPGA Implementation:

A. *synthesis tool:*

Actually, the synthesis tools allow the use of FPGA resources with schematics entrance as well as an algorithmic one. 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 (figure 4).

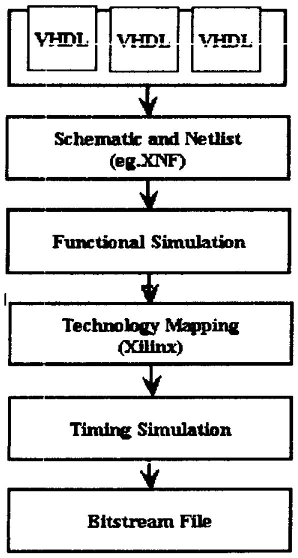

Figure 4. The design-flow of the synthesis

B. Architecture:

The MLP architecture, as shown on figure 2, is iterative and mainly connected and requires, a priori, (i*j) connections between 2 layers of i and j neurons. A null synaptic weight is applied to the non existent link. This leads us to imagine the building of a complete network but only with one layer at a time and to use the property of dynamic reconfiguration of the FPGA alone process which insures the possibility to implement the design on a single FPGA circuit.

♦ **Neuron Architecture:**

The structural description of a neuron with VHDL allows during the compiling step to specify generically some characteristics like the input numbers and the data size, and even to modify the type of some arithmetic operators like the adder or the multiplier and eventually the activation function unit if one wants to modify the activation function.

The neuron computes the product of its inputs, which are saved in RAM, with the corresponding synaptic weights, which are memorized in an EEPROM, and then the results are added. The result is presented to a comparison unit designed to represent an appropriate sigmoid function. Then the output of this comparison unit is used as a controller for a multiplexing unit, which delivers the y output of the neuron (figure 5)

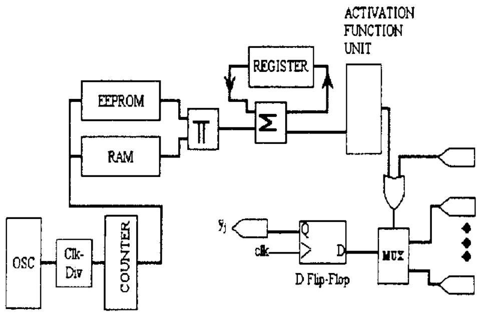

Figure 5. Neural processor architecture

♦ **Layer Architecture**

In the same manner the structural VHDL description of a layer allows during the compilation phase to determine the number of neurons on this layer, which gives the possibility to synthesize layers with any number of neurons. (Figure 6)

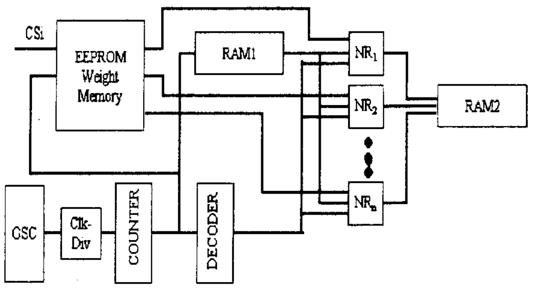

Figure 6. Layer architecture

- **Network Architecture:**

The ability to exploit the run-time reconfiguration of the FPGA circuit leads us to take advantage of 2 RAMs (1) and (2) in order to process the output data of the neuron layer (i-1) by the following layer (i).

4. Conclusion:

The use of run-time reconfiguration of the FPGA of the Xilinx company will allow us to develop and implement some different algorithms 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.

5. References:

[1] J.W.Gardner, H.V.Shurmer and T.T.Tan, Application of an electronic nose to the discrimination of coffees, *Sensors and Actuators B*, 6 (1992) 71-75.

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

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

[4] M.Hubin, Y.Taright, S.Lee, Multisensor Smart Microsystem for the Supervision of Accidental Atmospheric Pollution, *8th International Congress for Sensor , Transducers & systems, Nurnberg,* (1997) Vol 2. 129-133.

[5] J.R.Armstrong, F.Gail Gray, Structured Logic Design with VHDL *,PTR Prentice Hall.*

[6] XILINX. *The programmable Gate Array Data Book.* 1995.