Hardware Implementation of Programmable Neural Networks for CHEMFET Signals Analysis in SEWING Project

A. Klepaczko*, A. Napieralski*, R. Kielbik*, J. M. Moreno**

* Department of Microelectronics & Computer Science, Technical University of Lodz
Al. Politechniki 11, PL-93590 Lodz, {kleppack, napier, rkielbik}@dmcs.p.lodz.pl
** Department of Electronic Engineering, Technical University of Catalunya
C4, Campus Nord c/Gran Capita s/n, 08034 Barcelona, Spain, moreno@eel.upc.es

ABSTRACT

This paper focuses on problems of hardware implementation of neural networks in the re–programmable structures. The project presented here is an attempt to realize this task in a System–on–Chip device. Great concern has been put on the design's flexibility, so that it would be applicable in a number of not predetermined jobs, independently from the level of their complexity. Programmable features that have been introduced to the design enable users to suit the system to their personalized demands.

Much attention has been also devoted to the practical application of the neural network in the System for European Water Monitoring. The constructed system improves CHEMFET sensors' behavior, whose reliability is still not satisfactory enough.

Keywords: neural network, system–on–chip, hardware/software co–design, SEWING, CHEMFET sensors.

1. INTRODUCTION

The goal of this project is to construct programmable multi–layer perceptron (MLP) network and implement it in the re–programmable structure. It must be noted that only few devices based on the neural networks' (NN) principle have found their commercial realization in the form of integrated circuits (IC). Fully hardware implementation meets numerous barriers which limit networks' capabilities and constitute a remarkable challenge for designers. A need for dealing with the problems of limited logic resources appears particularly when using digital equipment.

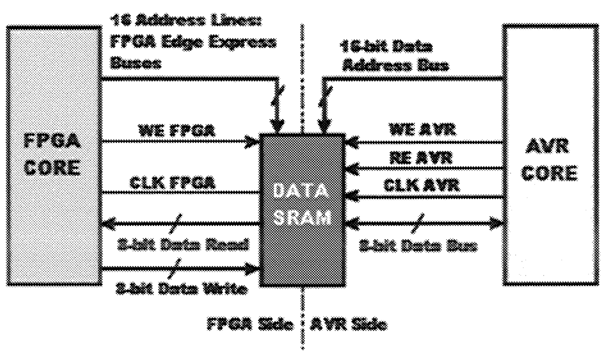

Figure 1: Basic blocks of the AT94K [1]

This project explores possibilities of neural network implementation in the Atmel's AT94K Field Programmable System Level Integrated Circuit (FPSLIC, see figure 1), that belongs to the group of devices which share the common name *System–on–chip* (SoC). This new class of electronic tools, which integrate in one silicon wafer entire microprocessor systems, facilitates NN construction and their application. The cooperation of the microcontroller unit (MCU) and field programmable gate array (FPGA) helps to overcome space– and interconnection–limitations.

2. SPECIFICATION

Figure 2 presents possible architecture of the *multi–layer perceptron* (MLP). The structure depicted in the figure comprises of two layers of neurons (the output and the hidden layer) and the set of input nodes. Both the input and the output vectors of the hidden layer are extended on the constant value, what enables neuron's threshold modification.

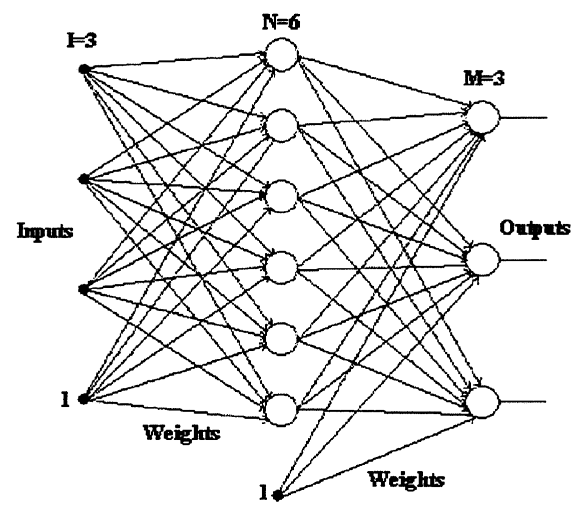

Figure 2: Example of the MLP structure

Structure in this or similar form is able to solve many non–linearly separable complex problems (should more burdensome jobs be attained larger networks have to be used). The hidden layer transforms input vectors into the space where they are linearly separable. Signals are then propagated through the output set of neurons that execute final classification [2], [3]. The formal description of a neuron's behaviour may be expressed in the equation (1):

$$z_m = f\left(\sum_{j=1}^{N+1} w_{mj} y_j\right). \qquad (1)$$

The constructed system enables achieving a two–layer network of perceptrons activated in the hidden layer by the bipolar sigmoid function:

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

and by the linear function in the output layer:

$$f(x) = a \cdot x. \qquad (3)$$

It is expected from the user to declare the numbers of neurons of the hidden (N) and the output (M) layer and specify the length of the input vector (I). Though the numbers I, N and M are optional, the project in the realized form does not allow exceeding the maximum values: I, N = 9, M = 10. This restriction is, above all, the matter of the micro–controller's program structure and may be easily extended on higher ranges.

Network's training is not the part of the project as far as its implementation in hardware is concerned. It is presumed that new weights, appropriate to the current application, are calculated for the desired MLP architecture, according to the BP algorithm on the PC–platform and then downloaded directly to the project together with the design's properties — I, N, M.

3. IMPLEMENTATION

3.1. System Partitioning

Systems based on the conjunction of microprocessor and logic arrays require a specific programming approach. Just after formulating project's specification, designer needs to divide the tasks to be realised between the processor and the blocks integrated with it. Such an approach is called *hardware/software co-design*. The partitioning of this project — shown in figure 3 — emerges directly from the specification presented above.

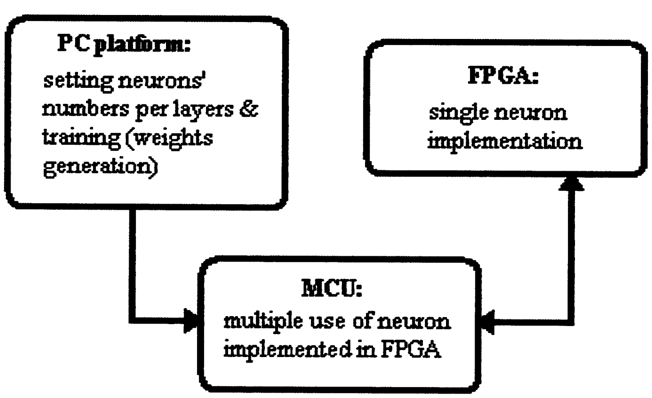

Figure 3: Project's partitioning

Programmability of the MLP structure implies that in case of any new application it needs to be reconstructed according to the current numbers: I, N and M. The system should thus ensure neuron's model flexibility, so that it could accept input vectors of dynamically altered length (which depends on I and N) and multiply them by the pertinent weight vector. Moreover, number of neurons in both layers required for the ultimate MLP network's instantiation is also application-dependent (parameters: N and M). This suggests employing some basic neural cell which could be used as a standard component for building any larger network.

3.2. Hardware Part

The experiments carried out on AT94K proved that fully parallel neuron's model (i.e. a model where multiplications of all input vector's compounds by the corresponding weights are performed simultaneously in the separate multipliers) yields danger of reaching resources limits. Thus, sequential model has been developed to deal with this problem. Its basic blocks are depicted in figure 4.

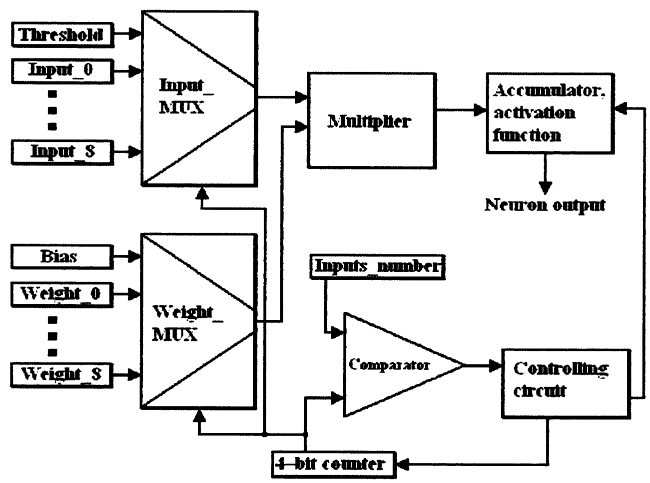

Figure 4: Single perceptron model

It comprises of the set of 10 registers containing input vector compounds (input registers *Threshold*—*Input_8*) and 10 registers for storing weights (weight registers *Bias* — *Weight_8*). The number of 10 is an arbitrary value and may be extended on higher values. These registers are programmed by the MCU. Moreover, the micro–controller accesses the register that contains the number of inputs in a current neuron (*Inputs_number*). This register prevents from presenting to the multiplier the excessive compounds of input and weight vector. The operation is initiated by the MCU which sets a special bit in the controlling circuit. This forces 4-bit counter to start incrementing its contents. The counter selects through the multiplexors the subsequent pairs of inputs and weights which are multiplied — their products are aggregated in the accumulator. When the counter reaches the number of inputs in the current neuron, the comparator issues HIGH state on its output, causing the controlling circuit to terminate counter's operation and reset it. The contents of the accumulator is the address for the ROM-type memory, where the values of the activation function are stored. The data output of this memory is the response of the current neuron, which is ready to be captured.

3.3. Software Part

As it has already been mentioned the software part is responsible for creating a multi–layer structure basing on one neuron instance. Perceptron model implemented in the FPGA is used number of times required to achieve the desired NN architecture. The main algorithm realized by the microprocessor is parameterized by the values of I, N and M, declared at beginning of the program. These three numbers define neurons sizes and their amounts in both layers. The first stages of the algorithm allow defining (by the appropriate code modification) network's parameters and weight matrices in both layers. Next steps are executed after system's initiation by some external device (figure 5).

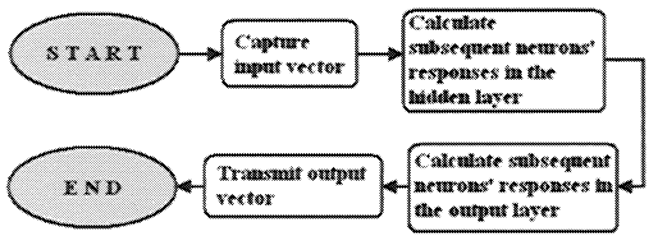

Figure 5: Overall way of operation

When such a START condition is detected, MCU receives input vector from external environment. After this process is finished, calculations in the hidden layer are performed. With respect to the actual values of I and N, micro–controller loads appropriate FPGA–implemented registers (*Bias* — *Weight_8*) with the weight vector compounds declared in the program and stored in data RAM. Consecutive neurons responses are captured from FPGA and stored in micro–controller's data memory. When the last neuron in the hidden layer is reached, system continues with the computations in the output layer. MCU modifies the contents of the input registers (by loading them with the output vector of the hidden layer) and programs weight registers with the weight vectors for the subsequent output neurons. If the response of the last neuron in the output layer is ready, the output vector is transmitted back to the external environment and the operation terminates.

4. APPLICATION

An inspiration for this project and creating a neural network as a hardware structure has been the SEWING project (System for European Water Monitoring). European–wide researches aim to create a cheap and generally accessible system for monitoring and early warning of water pollution [4]. The CHEMFET sensors that will be used in the project and which are supposed to be responsible for detecting ions of pollution are not selective enough yet. Sensors react not only to one (main) ion but they are also sensitive to some other (disturbing) ions.

The behavior of the CHEMFET sensors may be described by the semi–empirical Nikolsky equation (4) [5]:

$$E_M = E_0 + 2.303 \cdot \frac{RT}{z_k F} \log\left(c_i + \sum k_{ij} c_j\right) \tag{4}$$

where:

E_0 — constant reference potential,
R — universal gas constant,
F — Faraday's constant,
T — absolute temperature,
z_k — electrovalence of ions (the same for all ions in the interfacial equilibrium) ,
c_i — concentration of the main ion detected,
c_j — concentration of the disturbing ion,
k_{ij} — selectivity coefficient.

Figure 6 presents the influence of one disturbing ion — of the constant concentrations $c_j^1 = 10^{-2}$ and $c_j^2 = 10^{-3}$, and the selectivity coefficient is in both cases $k_{ij} = 0.01$ — on the sensor's indications.

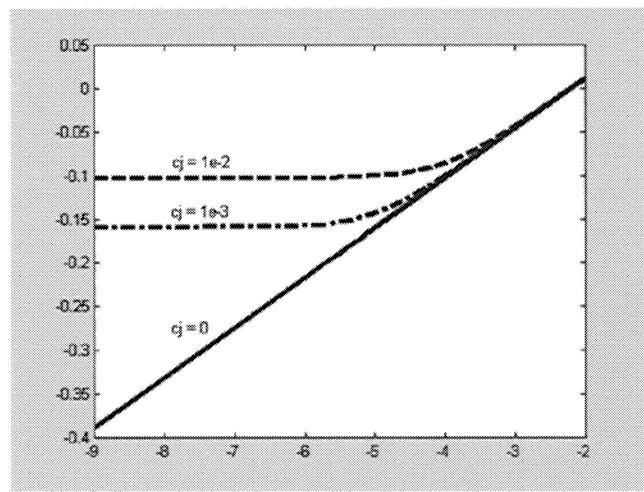

Figure 6: Ideal and corrupted CHEMFET sensors responses

As it can be seen, the difference between the ideal sensor responses and the corrupted ones emerges mainly in the lower range of the main ion concentrations. The higher concentration of the disturbing ion is, the larger is the deviation from the ideal flow. For the higher main ion concentration values, where the component responsible for taking the interfering ion into account is significantly lower, no perturbations are observed.

Since it is difficult to estimate particular ion concentration, relying only on the response of one sensor, information from the other is needed. It cannot be excluded that sensor's indication is violated by one or more disturbing ions. Different than expected voltage on a sensor output may be caused by the enriched concentration of one disturbing ion or by the existence of two disturbing ions but in smaller proportions. Correct interpretation of the sensors' outputs by the traditional processing units appears quite complicated, whereas an attempt to use a neural network may occur successful.

Let us suppose that there are three sensors, sensitive to ions X (concentration c_X), Y (c_Y) and Z (c_Z), responding with the values *v(X)*, *v(Y)*, *v(Z)*, and the expected values relative to the real concentrations are *e(X)*, *e(Y)*, *e(Z)*. If the coefficients k_{XY} , k_{XZ}, k_{YX}, k_{YZ}, k_{ZX}, k_{ZY} define mutual relations between sensors indications and ions concentrations, following equations may be formulated (after evaluating constant expressions in the equation 4):

$$
\begin{aligned}
v(X) &= 0.125 + 0.057 \cdot \log\left(c_X + k_{XY} c_Y + k_{XZ} c_Z\right), \\
v(Y) &= 0.125 + 0.057 \cdot \log\left(c_Y + k_{YX} c_X + k_{YZ} c_Z\right), \\
v(Z) &= 0.125 + 0.057 \cdot \log\left(c_Z + k_{ZX} c_X + k_{ZY} c_Y\right).
\end{aligned} \tag{5}
$$

Considering the above, expected responses *e(X)*, *e(Y)*, *e(Z)* can be derived from the following, non-linear equations:

$$
\begin{aligned}
e(X) &= f_1\left(v(X), v(Y), v(Z)\right) = \\
&\quad f_1\left(c_X, c_Y, c_Z, k_{XY}, k_{XZ}, k_{YX}, k_{YZ}, k_{ZX}, k_{ZY}\right), \\
e(Y) &= f_2\left(v(X), v(Y), v(Z)\right) = \\
&\quad f_2\left(c_X, c_Y, c_Z, k_{XY}, k_{XZ}, k_{YX}, k_{YZ}, k_{ZX}, k_{ZY}\right), \\
e(Z) &= f_3\left(v(X), v(Y), v(Z)\right) = \\
&\quad f_3\left(c_X, c_Y, c_Z, k_{XY}, k_{XZ}, k_{YX}, k_{YZ}, k_{ZX}, k_{ZY}\right).
\end{aligned} \tag{6}
$$

Provided that there are no other polluting ions (i.e. except for X, Y and Z) in the investigated environment, the system obtains data from all sensors, and the number of teaching pairs is big enough throughout the training process, the network should adjust itself to the selectivity coefficients and realize functions defined by the equations (6).

As the CHEMFET sensors are still under development, not many experimentally obtained sensors responses are available. Consequently, for the simulation purposes, teaching pairs had to be generated manually. Table 1 presents three example test vectors (column I), the desired network's output (column II) and the results of calculations in the trained network (columns IV — MATLAB simulation and VI — hardware simulation). To evaluate the quality of network's performance, the mean–square error criterion was used to calculate deviations of:

- input test vectors from the expected results (column III):

$$MSE\_I = \sum_i \left[x(i) - d(i)\right]^2, \qquad i = A, B, C, \tag{7}$$

- output vectors from the expected results (columns V and VII):

$$MSE\_O = \sum_i \left[z(i) - d(i)\right]^2, \qquad i = A, B, C. \tag{8}$$

As it may be judged from the analysis of the table 1, the results obtained in MATLAB calculations approximate quite properly the expected values (corresponding to the real main ions concentrations). In some cases, network may respond with still misleading results. Even then however, the MSE is significantly reduced. Possibly, if more effective training methods were used, or more epochs were launched, network's behavior would be even more reliable.

Table 1: Simulation results

Ion	I	II	III	IV	V	VI	VII
X	0.177	0.141		0.143		0.122	
Y	0.170	0.102	0.006	0.115	0.001	0.113	0.003
Z	0.278	0.278		0.266		0.230	
X	0.412	0.410		0.405		0.452	
Y	0.459	0.459	0.058	0.463	0.002	0.502	0.011
Z	0.356	0.115		0.163		0.201	
X	0.313	0.313		0.302		0.294	
Y	0.212	0.158	0.005	0.185	0.001	0.185	0.003
Z	0.207	0.165		0.164		0.113	

The hardware simulation of the trained MLP structure — when compared to the software one — is slightly worse. Deviation of some of the output compounds from the desired ones exceed reasonable level. The inaccuracies in the hardware computations are caused by the imprecise binary representation and binary arithmetics that evoke necessity of rounding the weight and input values and lead to approximated results. However, most of the output vectors are equivalent to the expected ones and the overall error committed by the network is lower than on the system's input.

5. SUMMARY

The general goal of the project — an attempt to create neural network as a hardware circuit — has been successfully achieved. The structure implemented in the Atmel's AT94K FPSLIC responds on the input vectors according to the calculating algorithm for the multi–layer perceptron networks. Moreover, this project showed that the combination of the hardware and software resources leads to efficient NN implementation. Network's programmability lets an end–user suit the system to custom application. This feature enhances project's utility increasing the range of tasks it may solve.

The simulation results — based on manually generated teaching pairs — are very much promising and do not exclude its possible usage in the System for European Water Monitoring. However, further tests should be carried out, this time based on the real experimentally obtained teaching pairs, in order to confirm or deny project's applicability in SEWING.

REFERENCES

[1] Atmel Corp., AT94K Series Field Programmable System Level Integrated Circuit. Atmel doc1138.pdf.

[2] J. Żurada, "Introduction to Artificial Neural Networks", West Publishing Company, USA 1992.

[3] J. Żurada, M. Barski, W. Jędruch, "Sztuczne Sieci Neuronowe", PWN, Warszawa 1996.

[4] http://www.sewing.mixdes.org

[5] W. Torbicz, "Teoria i własności tranzystorów polowych jako czujników biochemicznych", Ossolineum, Wrocław 1988.