A NEURAL NETWORK IMPLEMENTATION ON FPGA FOR ECOLOGICAL MONITORING

Diaf YOUSSOUF Kaddeche MOHAMED
Laboratory of Study and Research on Instrumentation and Communication Annaba (LERICA), University Badji Mokhtar Annaba, P.O. BOX 12, 23000 Annaba Algeria.
youcef.diaf@univ-annaba.org, mohamed.kaddeche@univ-annaba.org.

Abstract: In this paper, we present an implementation of an artificial neural network (ANN) for predicting blooms (coloured waters) of a species of toxic phytoplankton called Dinophysis acumunata that prevalent in coastal of Havre (France). A Multi-layer Perceptron (MLP) neural network is designed to predict the rate of algal cells concentration in a short time series. The processed data are extracted from analysis of average numerical values of physical and chemical parameters. The implementation of the designed neural network on FPGA (FPGA: Field Programmable Gate Array), a programmable circuit, help us to set up an automatic monitoring lagoon system.

Keywords: *Automated monitoring, Toxic algae, Eutrophication, Phytoplankton, Neural network, FPGA.*

I. INTRODUCTION

The phytoplankton's or algae's are microscopic organisms of few microns size. They have a great importance in the chain food and environmental water balance. The presence of some species of toxic algae (Dinophysis Alexendriuem) is the origin of blooms called also coloured waters.

The bloom or the eutrophication is the increase and acceleration of the organic algae production at a rate where they colonize their environment. We speak here of the biological pollution of coastal waters by phytoplankton. This type of pollution makes water unsuitable and the consumption of shellfish very dangerous. In other terms it's a danger for human health and can cause negative effects on the environment. Therefore, it is necessary to ensure monitoring of the quality of water in these areas and promote all efforts to alert the population and the authorities in case of pollution rate exceeds tolerable threshold.

However, difficulties persist in the accuracy of the classification and detection procedures. The process of appearance and development of these microorganisms remains unclear, but it seems that certain environmental conditions influence the process of appearance of these organisms. The main objective to design a monitoring system is primarily to overcome the old methods that include a taxonomic analysis that take several days before to achieve a definitive conclusion and to reduce the cost for operators and tools.

The Multi-layer Perceptron (MLP) neural network type has proven its power for the approximation and the prediction of nonlinear and random process by exploiting the physical and chemical environmental data [1;2;3;8;9].

The goal of this paper is to implement the MLP neural network model on FPGA to be installed permanently in a site. The steps of data acquisition and conversion from sensors are made out of FPGA and are not addressed in this work. The back propagation algorithm used for training the MLP neural network is performed off line with a personal computer.

II. METHODS

II.1. Phytoplankton eutrophication model

Eutrophication is defined as the enrichment of water with nutrient elements useful for plants or other primary producer's growth. The eutrophication of lakes and rivers is a growing problem across the world. It affects more and more communities. Models of phytoplankton cell growth are limited and reduced. This is due to a large number of biological and physiological parameters that no database can reaches; or because of the detailed description of the cell processes that require variables which cannot be obtained experimentally [4], [5], [6].

Our database consists of 37 samples of coastal seawater from "Le Havre France"; for an average of three months: July, August and September of the year 1985. These data show the concentration of cells evolution of the toxic algae Dinophysis ACUMINATA, depending on physical and chemical parameters. The considered parameters are shown in Table 1.

Among these environment variables, some affects directly the Algae cells growth like temperature, pigments solar, nutrients, organic carbon; others are the results of cells photosynthetic process such the chlorophyll and the oxygen.

II.2. Optimization of the neural network inputs

The neural network has the ability to determine critical inputs, but the use of too many inputs can affect the speed of the model; increase the size of the model implementation and cause redundancy problems between different variables [9]. The physical and chemical parameters that affect the blooms remain poorly unknown [10]. Principal component analysis (PCA) is used to select the pertinent inputs of the neural network. To reduce the number of attributes, we divided these variables into several groups based on PCA

Manuscript received April 25, 2013; revised September 14, 2013

result and works already done on the species Dinophysis acuminata. The retained groups G1, G2 and G3 giving good results in the neural network training are shown in Table 2.

Table1. The considered environmental parameters

Variables	variable description	Measurement unit
T	Temperature	°C
O2	Oxygen	mg/l
COD	Carbon Organic Dissolved	ppcm
COT	Carbon Organic Total	ppcm
PO4	Phosphate	µatg/l
NH4	Ammonium	µatg/l
NO2	Nitrite	µatg/l
NO3	Nitrate	µatg/l
SI	Silica	µatg/l
CHLOR	Chlorophyll	mg/m3
Pigment	Pigments	mg/m3

Table 2.The top three inputs combination groups G1, G2 and G3.

G1	G2	G3
T	T	T
O2	O2	O2
NO3	NO3	NO3
NO2	NO2	NO2
NH4	NH4	NH4
COT	PIgm	PO4
COD	CHLOR	SI

II.3. MLP artificial neuronal network model background.

A neural network model with MLP architecture is performed to predict the evolution of Dinophysis acuminata cells concentration. We have tested different network architectures to find a good compromise between size and error rate. MLP architecture models have proven their power when the process of the cell proliferation is theoretically unknown or has a high nonlinearity. A MLP network can approximate any function that is linear or nonlinear. This property is very helpful for decision and prediction applications.

The MLP network allows only the direct connection from lower layer to upper layer of neuron as shown in figure 1. The optimum MLP network i.e. the optimum number of inputs; the number of hidden layers and the number of neurons in each hidden layer, is obtained with the test and error approach [2] [3]; so several MLP network architectures were tested to determine which network has good compromise between size and reduced error.

All data were normalized in the range 0 to 1 according to the following formula:

$$K_n = \frac{K - K_{\min}}{K_{\max} - K_{\min}} \quad (1)$$

Where K represents any data that is either an input or output; and K_n is the normalized data.

Any layer contains bias entries equal to 1. Each neuron performs the function given by the following formula.

$$x_j = \sum_{i=1}^{n} W_{j,i} \cdot K_i \quad (2)$$

Where W_j presents the weight vector of the j th neuron; and x_j is the sum of multiplication between inputs of neuron and the corresponding weights.

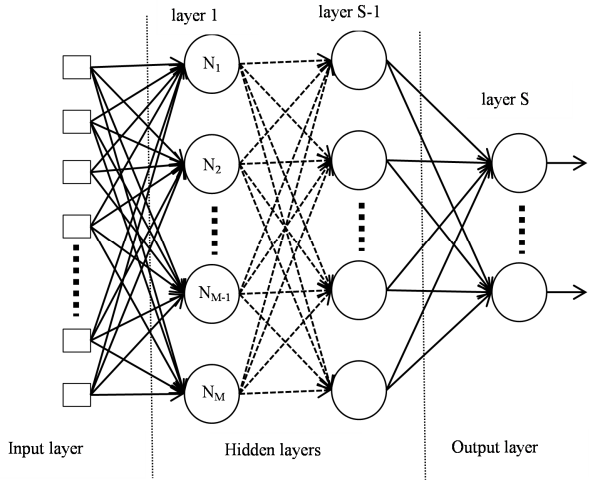

Figure1. MLP network architecture

Hidden layers contain sigmoid function calculated by the following formula:

$$y_j = f(x_j) = \frac{1}{1 + e^{x_j}} \quad (3)$$

Where y_j is the output of neuron j.

A linear transfer function is used in the output layer. The purpose of learning step is to adjust the weight of the network to obtain the desired output. The back-propagation algorithm adjusts the weights of intersection and bias by minimizing a criterion that is the sum of square errors e which is given by the following formula.

$$e = 0.5 \sum_{i=1}^{N} \left(\hat{y}_i - y_{d,i} \right)^2 \quad (4)$$

Where ŷ is the output generated by the network and y_d is the desired output of samples used for the learning step. The network is trained with the back-propagation Levenberg Marquardt algorithm [3]. This type of algorithm is a nonlinear second order least square optimization method based on the Newton method. In this algorithm, the weights of network are modified as follows.

$$\Delta w = -(J^T(w) \cdot J(w) + \eta I)^{-1} \cdot J^T(w) \cdot e(w) \quad (5)$$

Where J is the Jacobean matrix, containing the first derivative of the network error, w are the weights of the network, *e* is the error vector calculated as (4) and η is the constant learning rate.

The learning is good if at each time the value of the error *e* is less or equal to its previous value. The value of the parameter η is increased each time the error *e* is higher and decreased each time the error *e* is lower. The learning algorithm and the validation of the network are performed in the following order:

1. Starting with randomly weights,
2. Training the network with the learning database,
3. Check if the error $e \leq T_1$: if yes continue, otherwise return to step 1,
4. Simulate the network with the test database,
5. Check if the error $e \leq T_2$: if yes continue, otherwise return to step 2,
6. Save the network and simulate the validation database.

In these learning steps T_1 and T_2 are the learning and the testing maximum errors respectively; those values are set in order to have the best simulation results. This method produces successful prediction because of the constraint imposed by T_2 in the test phase.

II.4. Design and physical implementation of neural model on FPGA

The FPGA (Field Programmable Gate Array) is a logical programmable circuit. It presents a flexible structure by means of its programmable interconnection blocks (see Figure 2). Moreover, the fact that parameter adjustment and changes can be done easily on FPGA ensures reduced production time and cost.

We have used Xilinx ISE tool for the design of the MLP neuronal model on FPGA type Virtex5 XC5VL110T system. This type of FPGA contains 31 million programmable logic gates which are placed on ML501 platform. Figure 3 shows the flow calculation and the various components of the programmable circuits on FPGA.

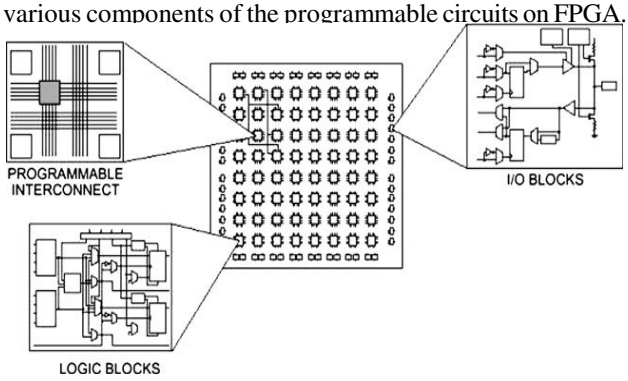

Figure 2.FPGA's basic building structures

II.4.1. Numbers representation

Only fixed point representation is used in the current implementation of the architecture since floating point representations are not attractive for ANN designs [13]. We opted this type of number representation because of the measurements from the sensors have the same order of magnitude after normalization. The number representation is limited to a maximum precision of 16 bits that seems to be sufficient, especially for representation of network weights, and it achieved a good compromise between time and area for arithmetic operator implementation.

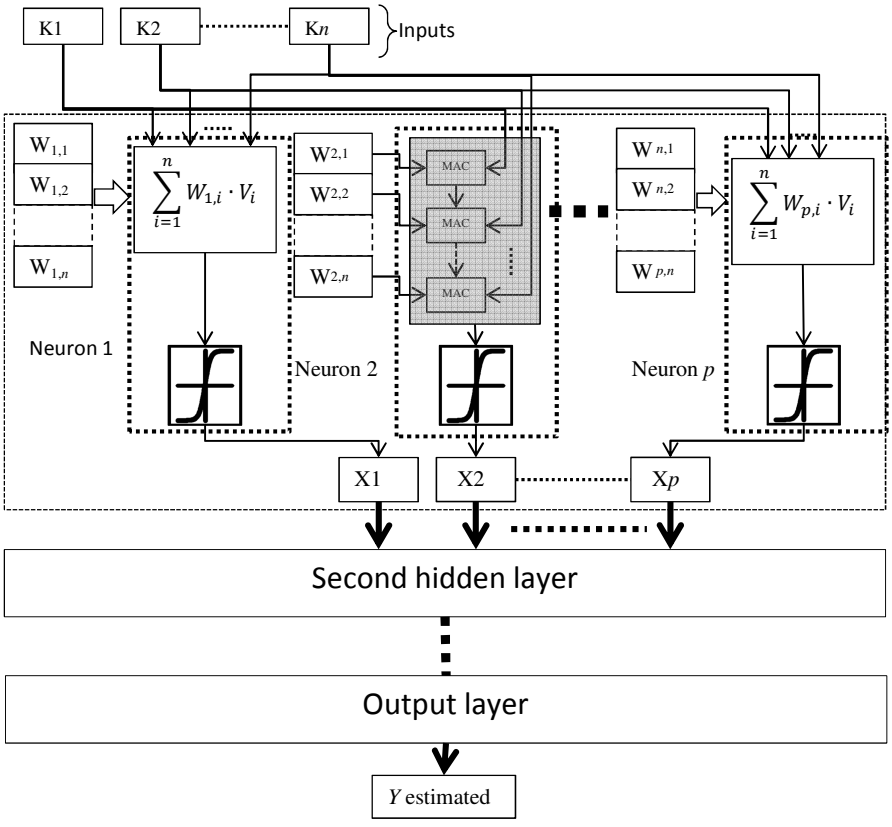

Figure 3.Flow calculation and different circuit components.

An optimal fixed point representation is one of 16 bits total width with a balance between integer and fractional resolution. A 1-4-11 representation is used throughout this work, with a range of ±16. The values are also saturated if they exceed the dynamic field of the process.

II.4.2. Neuron architecture

The number of inputs determines the width of the weighted sum module implementing Eq. (1), as well as the size and width of the memory that holds associated neural weights and the transfer function module implementing the Eq. (2).

The sum product calculation performed by the neuron is provided with a systolic architecture. It is a structure composed of several units or arithmetic element processors (EP) connected locally to increase the calculation flow. The creation of an EP allows generation of the complete circuit; we grown only the systolic array while keeping the same topology. The matrix vector product is often performed from a linear systolic architecture as proposed in [7] and as shown in Figure 3. In the gray area, each EP is composed of several MAC (Multiplier accumulator) which must perform the following operation:

$$C = C_{t-1} + w \cdot v \tag{6}$$

Where C_{t-1} is the value sent by the previous MAC, so, the value of C must be reset to zero when a new matrix-vector product should be done.

II.4.2. Sigmoid activation function

The most difficult component in the implementation of our neural network is the sigmoid activation function. Several possibilities exist to approximate the sigmoid function [11], [12]. In our case, we have used a piecewise linear approximation proposed by [11], where the sigmoid function is divided into eight segments with X comprised in the interval [-15, 15] and Y in the interval [0, 1].

The sigmoid function and its approximation are symmetrical in [Y = 0.5, X = 0], which means that the calculation is performed only for the positive values of X in order to determine the corresponding value output Y. We give in (7) the approximation of the sigmoid function:

$$\begin{cases} Y = Y_1 = 0.25|X| + 0.5 & 0 \le |X| < 1 \\ Y = Y_2 = 0.125|X| + 0.625 & 1 \le |X| < 2.375 \\ Y = Y_3 = 0.03125|X| + 0.84375 & 2.375 \le |X| < 5 \\ Y = Y_4 = 1 & |X| \ge 5 \\ Y = Y' = 1 - Y & X < 0 \end{cases} \tag{7}$$

Figure 4 shows the obtained function with MATLAB7.6 and the function obtained after synthesis in VHDL with ISE9.2.

III. Results and comments

The database is divided into three time series. First 20 samples collected in July are used for training the designed neural network; the 7 other samples collected in the month of August represent the test database and the last 10 samples collected in September are used to validate the short term prediction behavior of the network. Several values in the test and validation steps are not present in the learning database. The simulation of the neural model is performed with MATLAB R2008a 7.6 and the synthesis of the implementation is realized with XILINX ISE12.2.

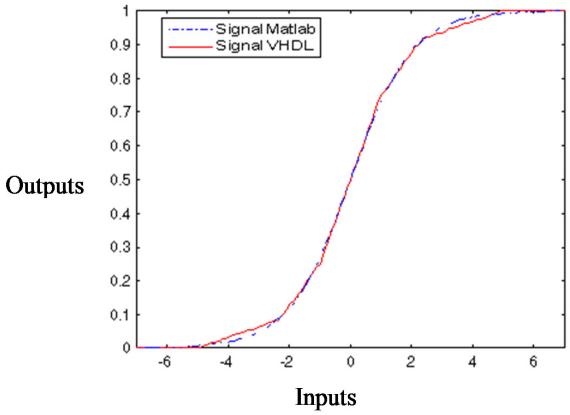

Figure 4. Activation function in MATLAB7.6 and VHDL.

III.1. Search for the optimal model

We executed simulations with several neural models: with one hidden layer and two hidden layers. We tested those models with several variants of neurons in the hidden layer (10, 9, 8, 7, 6). The mean square error e given by the formula (4) between the target and the output of the networks for the different combinations of inputs groups G1, G2 and G3 in function of the variation of neuron numbers in the hidden layer (represented as 6n,7n…10n. for 6,7…10 neurons and 6-10n for 6 neurons in the first layer and 10 neurons in the second layer). The obtained results are represented in Figure 5.a, Figure 5.b and Figure 5.c for the case of learning, the test case and the validation case respectively.

The group G1 (T, O2, NO3, NO2, NH4, TOC, COD) is the best group in the simulation results for the three phases; the error e vary for learning phase between 0.0016 and 0.0084 for the test between 0.0123 and 0.0181.

The best results in prediction phase are obtained with two-layer architecture with 10 neurons in the first layer and 6 neurons in the second layer. This architecture has allowed us to achieve very good results in the three phases; the error e is 0.0032 for learning, 0.0169 for the test and 0.0458 for prediction. Figure 6 shows the real number of cells and the two-layer architecture estimated.

The use of this network to monitor and predict blooms caused by Dinophysis Acuminata is very promising.

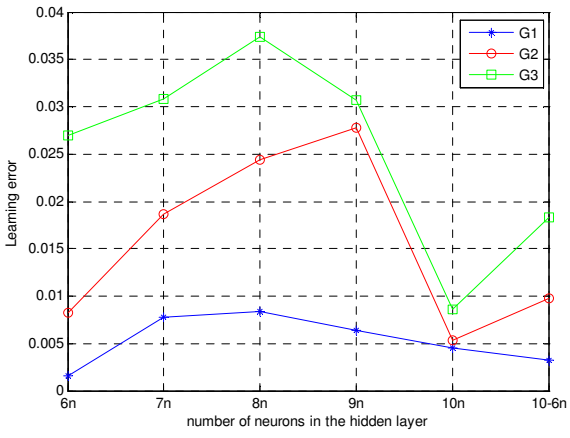

Figure 5.a.Learning error for group G1, G2 and G3.

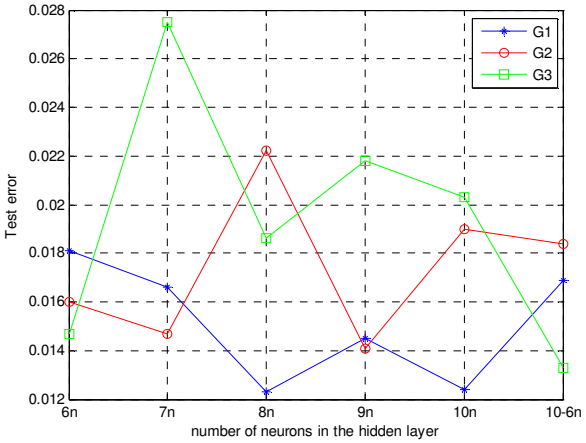

Figure 5.b.Test error for groups G1, G2 and G3.

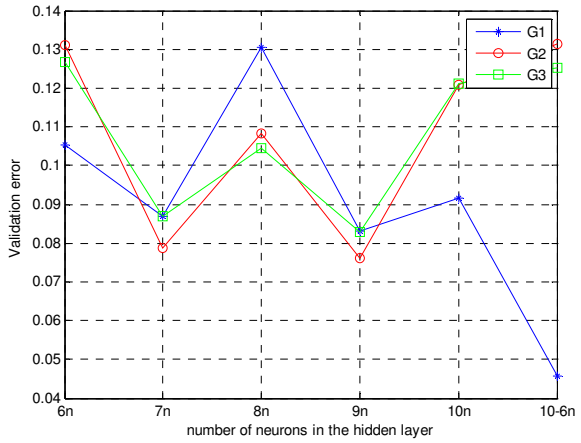

Figure 5.c.Validation error for groups G1, G2 and G3.

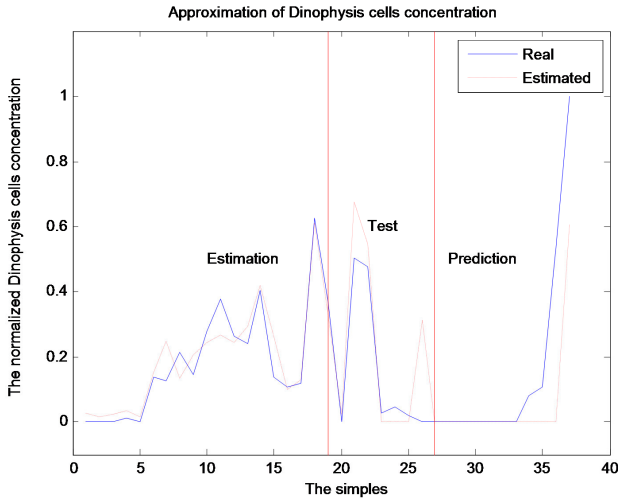

Figure 6. Result obtained with MATLAB7.6 for the architecture 10-6 neurons in hidden layers

III.2. Implementation results

Once the simulation and the network selection are made with MATLAB7.6; we generate the complete architecture i.e. the creation of all different network components in VHDL language. The back-propagation algorithm is not included in the FPGA circuit and performed off line.

The implementation was performed with XILINX ISE12.2 and its report with Virtex-5 XC5VL110T is presented in Table 3. The minimum period is 30.700ns (Maximum Frequency: 32.574MHz) and the maximum time for output generation after th e first clock cycle is 3.259ns. These results are obtained with a neural network containing 7 entries and architecture of 10 neurons in the first hidden layer and 6 neurons in the second hidden layer, and by using the component configuration and the arithmetic representation presented in Part II . The simulation was performed with XILINX ISESIM12.2. We set the clock period to100 nanosecond; the output is generated by the circuit in 2.3 µs for each input vector. The network initialization is 1 µs; the propagation time in the first layer is 0.4 µs and 0.9 µs for the second layer. The error between the estimated output obtained with the synthesis of architecture with ISESIM12.2 and the MATLAB7.6 software is 0.351 for the 37 studied samples.

Table 3. The XILINX ISE12.2 implementation report with Virtex-5 XC5VL110T circuit.

Slice LogicUtilization	Used	Available	Utilization
Number of Slice Registers	4,968	69,120	7%
Number of Slice LUTs	28,102	69,120	40%
Number of occupied Slices	8,893	17,280	51%
Number of bondedIOBs	130	640	20%
Number of DSP48Es	64	64	100%

VI. CONCLUSIONS

In this work we have simulated a bloom model of toxic algal species, Dinophysis acumunata, from measurable data.

The difficulty in this type of modeling is the strong nonlinearity and the a priori knowledge of those processes. We have used for modeling the process variables that are easy to measure with specific probe sensors. With those variables we have built several architectures of MLP neural networks. The architecture of two hidden layers gives good results in short term prediction. The implementation of this MLP network is done with the VHDL language. The systolic architecture of the network helps in the realization of complete network by building a few elementary blocks. The simulation has shown that the response of the model implemented on FPGA was almost identical to that given by the MATLAB7.6 software and it's much recommended to install it in a site.

REFERENCES

[1] C.Chabbi, M. Taibi and B. Khier, "Neural And Hybrid Neural Modeling of a Yeast Fermentation Process", International Journal of Computational Cognition, Vol. 6 (3), September 2008.

[2] M. N. Karim and S. L. Rivera. "ANN in bioprocess state estimation", Advances in Biochemical Engineering Biotechnology, Vol.46:1–33, 1992.
[3] F. B. Marand. "Modélisation et Identification des Systèmes Non-Linéaires par Réseaux de Neurones a Temps Continu", PhD Thesis, Université de Poitiers, France, 2007.
[4] Geider R.J., MacIntyre H.L., Kana T.M., "A dynamic regulatory model of phytoplanktonic acclimatation to light, nutrients and temperature", Limnol. Oceanogr., Vol.43 (4): 679-694. 1998.
[5] Dunaliellatertiolecta, "limited simultaneously by light and nitrate", Limnol. Oceanogr., Vol.42(6): 1325-1339.
[6] Shuter, B., "A model of physiological adaptation in unicellular algae", J. Theor.Biol., 78: 519-552. 1979.
[7] P. Quinton et Y. Robert, "Algorithmes et architectures systoliques", Masson, Paris,1989.
[8] Julian D. Olden "An artificial neural network approach for studying phytoplankton succession", Hydrobiologia 436: 131-143, 2000
[9] Maier, H. G., and G. C. Dandy. "Neural networks for the prediction and forecasting of water resources variables", a review of modelling issues and applications. Environmental Modelling and Software 15:101–124. 2000.
[10] Maier, H. G., and G. C. Dandy. "Neural network based modelling of environmental variables", a systematic approach. Mathematical and Computer Modelling 33:669–682. 2001.
[11] PJC. Clare & al., "Design and tuning of FPGA implementations of neural networks", SPIE Proc., Vol. 3069:129-136, 1997.
[12] H. Amin & al., "Piecewise linear approximation applied to nonlinear function of neural network", IEE Proc- Circuits Devices Syst., Vol. 144(6), pp 313-317. 1997.
[13] Mousa M, Areibi S, Nichols K, "On the arithmetic precision for implementing back-propagation networks on FPGA: a case study", In: Omondi AR, Rajapakse JC (eds) FPGA implementations of neural networks", Springer, US, pp 37–6. 2006