An Application in Bank Credit Risk Management System employing a BP Neural Network based on *sfloat24* custom math library using a low cost FPGA device

M. C. Miglionico[1], F. Parillo[2]

[1] Department of Culture of the Project – Second University of Napoli
Via S. Lorenzo, Monastero di San Lorenzo, I-81031 Aversa (CE) – ITALY
BENECON Scarl – Member of UNESCO
E-mail:mcristina.miglionico@unina2.it

[2] Department of Electrical Engineering and Information – University of Cassino
Via G. Di Biasio 43, I-03043 Cassino (FR) - ITALY
E-mail:f.parillo@unicas.it

Abstract. Artificial Neural Networks (ANN_S) base their processing capabilities in parallel architectures. This makes them useful to solve pattern recognition, system identification and control problems. In particular, it is extremely important for commercial banks to set up an early bank credit risk warning system. The authors set up early warning indicators for commercial bank credit risk, and carry out the warning for the credit risk in advance with the help of the ANN_S.

A three layer ANN has been implemented, using a custom developed *sfloat24* math library, on a low cost FPGA device.

Keywords: Artificial Neural Network, Field Programmable Gate Array ((FPGA), *sfloat24* math library.

1 Introduction

Artificial Neural Networks (ANN_S) are used with success in pattern recognition problems, function approximation, control, etc. There are different kind of electronic implementation of ANN_S, digital, analog and hybrid [1] and each one has specific advantage and disadvantages depending on the type and configuration of the network, training method and application.

For a digital implementation of ANN_S there are different alternatives, custom design, digital signal processors, programmable logic, etc. Programmable logic offers low cost, powerful software tools and true parallel implementations.

As well known the digital systems, in particular the FPGA devices have the following fundamental properties [2]:

- *Insensitivity to environment.* Digital systems are considered less sensitive to environmental conditions than analog systems. In contrast, digital system's operations do not depend on its environment.
- *Insensitivity to component tolerances.* Analog components are manufactured to particular tolerances. The overall response of an analog system depends on the actual values of all the analog used components.

ANN_s are nonlinear self-adaptive dynamic systems, which simulate human's neural system structure. They are ideal to solve early warning system of the commercial bank credit risk [3], [4].
Traditionally to solve early warning problems the following methods could be used:

- *Fuzzy logic* technique has been in its wide-ranging use in modelling of uncertainties, vagueness, impreciseness and the human thought process. The main problem of this approach is the fact that the credit analyst needs to analyse and assume a large number of differently valuated factors in a short time.
- *Monte Carlo* method usually takes a long time to simulate rare event, the failure event of repaying loans is treated as rare event due to the relatively low probability, and the failure probability of repaying loans is taken as the criterion to measure the level of credit risk [5].

The commercial bank's credit risk management could be analysed also adopting the following most recent, methods:

- *Bayesian Network* is a graphical representation of statistical relationships between random variables, through a Direct Acyclic Graph (DAG), widely used for statistical inference and machine learning. This method consists in two parts: in the first part is implemented a function that scores each DAG based on how accurately it represents the probabilistic relation between variables based on the realization in a generic dataset. In the second part a search procedure, that selects which DAG_s to score within the set of all possible DAG_s [6].
- *Support Vector Machine* (SVM) is an excellent method used to solve this kind of problem. This theory was initially developed by Vapnik. It is a learning machine based on statistical theory, and is used for classification and regression. The SVM is used to solve the over-fitting problem. Empirical risk is defined to be just the measured mean error rate on the training set [7].

To implement the bank credit risk management system, depicted in the section 3, an ANN is sufficiently suitable. Their usage avoids the very difficult task to implement one of the above mentioned methods on a low cost FPGA device.
In this paper the authors present the design and the implementation of a complete ANN on an ALTERA® Cyclone III EP3C25F324C8 FPGA evaluation board.
The implementation of an Artificial Neuron is widely described in [8].
The authors have developed a floating point math library for $FPGA_s$, called *sfloat24* [9], [10]. This library has been used, in this paper, for the Artificial Neural Network implementation. Respect to the classical IEEE 754 [11], [12] floating point number format, the numbers are stored in a 24 bit word length [13].

Using this library the smallest number, neglecting the sign (Most Significant Bit), is represented by the following word:

S **Exponent field** BIAS=63 **Fractional part of mantissa** [15 … 0]

that correspond to a decimal value of $1.08423e^{-19}$, the greatest value is represented by:

S **Exponent field** BIAS=63 **Fractional part of mantissa** [15 … 0]

that corresponds to: $18.4467e^{+18}$.

2 Artificial Neural Network ─ Back Propagation algorithm operating principle

The back-propagation (BP) learning process is widely adopted as a successful learning rule to find appropriate values of the weights for ANN_S. The BP learning process requires that the activation function is derivable. An example of a derivable function is the sigmoid.

A sigmoid function has the following expression:

$$f(x) = \frac{1}{(1+e^{-x})} \tag{1}$$

This function has been implemented using the CORDIC theory [14].

Here is discusses only the application of the CORDIC theory to implement the hyperbolic functions in order to show an optimized version of the algorithm used to implement this kind of math functions and consequently to implement the sigmoid function.

As said in [8] the hyperbolic functions can be formulated using the following expressions:

$$\vartheta_i = \tanh^{-1}(2^{-i}) \tag{2}$$

$$x_{i+1} = x_i + y_i d_i 2^{-i}$$

$$y_{i+1} = y_i + x_i d_i 2^{-i} \tag{3}$$

$$\vartheta_{i+1} = \vartheta_i - d_i \vartheta_i$$

where $d_i = -1$ if $\vartheta_i < 0$, +1 otherwise.

The hyperbolic functions are present in the x_i and y_i variables. In this case, the steps of *for* … *loop* construct are 24, because the input variable can assume any value; the hyperbolic are not circular math functions.

The exponential functions can be easily derived from the hyperbolic functions as following:

$$e^x = \sinh(\vartheta) + \cosh(\vartheta) \quad e^{-x} = -(\sinh(\vartheta) + \cosh(\vartheta)) \tag{4}$$

At this point it easy to formulate the expression (1) using the second expression (4). With reference to the Fig. 1, the output value of the hidden-layer and the output-layer should meet the following conditions:

$$a_j = f(\sum_{i=0}^{n} w_{[j][i]} - W_{[i][n+1]}) \quad \forall i \in N \quad y_j = f(\sum_{i=0}^{j} w_{h[j][i]} - W_{h[j][n+1]}) \; \forall i \in K \qquad (5)$$

where the quantities $W_{[i][n+1]}$ and $W_{[j][n+1]}$ are the thresholds of the input and the hidden layer respectively.

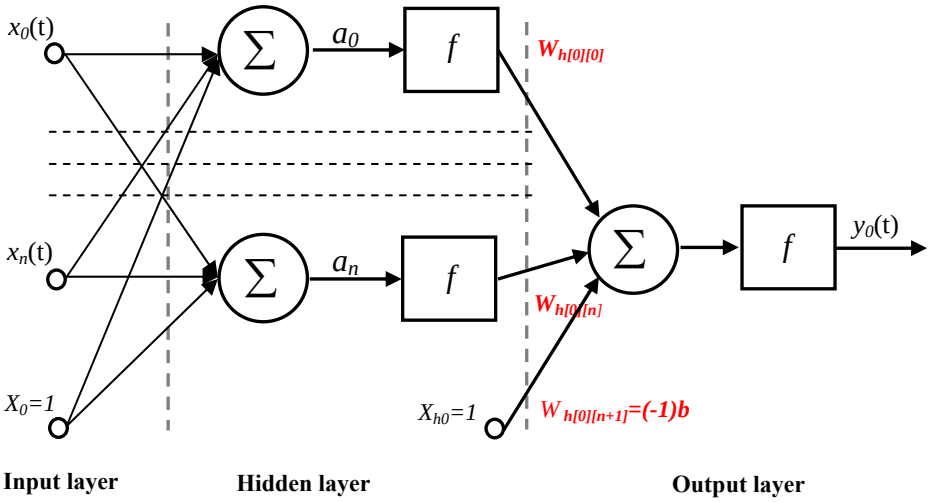

Fig. 1. Schematic diagram of a generic neural network.

The hidden layer node and the output layer node's transfer function uses, in general, the expression (1).
The purpose of the network training is to find a set of weights that minimize the following quantity:

$$E = \frac{1}{2}\sum_{i=0}^{k-1}(\overline{y}_i - y_i)^2 \qquad (6)$$

where $\overline{y}_i$ is the desired output and y_i is the actual output of the trained network.
In a BP ANN, the performances are heavily influenced by the weights correction method.

3 FPGA Implementation – Simulation results –

It is possible, on the basis of the expressions (2), (3) and (4), to build the sigmoid function.
The code of the hyperbolic functions has been written in VHDL. If the VHDL code is written similar to the C code depicted in [8], using, the *for* ….. *loop* instruction the *sfloat24* sigmoid activation function occupies the 44% of total logic elements of the ALTERA® Cyclone III EP3C25F324C8 device. In this case, the compilation, the

synthesis and the fitting process are very expensive in terms of requested time to perform all the mentioned operations.
Using the expression (1) to implement a full Artificial Neural Network similar to the one shown in Fig. 1, any Cyclone® III FPGA device family is not capable to perform this operation. In this case an ALTERA® Stratix FPGA could be required [8].
To avoid this, to reduce the occupation of the device logic elements, the authors have considered [15] the following approximation of the sigmoid function:

$$f_{ss} = 0.5 + (0.5 \cdot \frac{a}{1+|a|}) \tag{7}$$

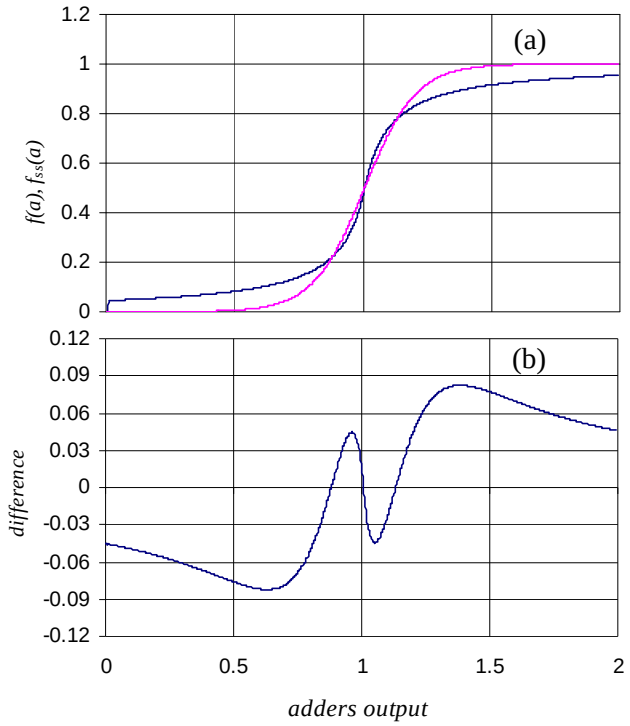

Fig. 2. Simulation results – (a) Sigmoid activation function and the approximation of the same, blue line. (b) The absolute difference between two math functions.

Implementing the (7), always using the *sfloat24* math library, the occupation the logic elements on the same FPGA device is only 7%, the optimization is obtained also during simulation phase.
The shortcoming of this approximation is that the BP algorithm numeric convergence time is greater respect to the case of an ANN that use as activation function a pure sigmoid. The BP algorithm has been written in MS® Visual Studio 2008. In Fig. 2 (b) is depicted the error between the expressions (1) and (7).
To improve the sigmoid function algorithm performances and to reduce its occupation into FPGA device, the *for … loop* construct has been substituted by a counting event of an external counter. This operation allows faster compilation, synthesis and fitting

device operations under ALTERA® Quartus II environment and with the same number of iterations.

In order to perform 25 iterations a 5 bit external counter is required, an auto-reset operation occurs when its value is greater of 24.

The counter is implemented on the same FPGA device, the term “external” means that it is not a component of the sigmoid code, but gives only the event (rising edge) to execute the implemented optimized code, as depicted in Fig.3.

In this case, considering 25 iterations, the device occupation of the sigmoid function VHDL code is 15%, instead 44% relative to the case of usage of the algorithm depicted in [15].

Using this version of the algorithm instead that described in [15] the compilation, synthesis and fitting operation in to FPGA device is 20 time faster.

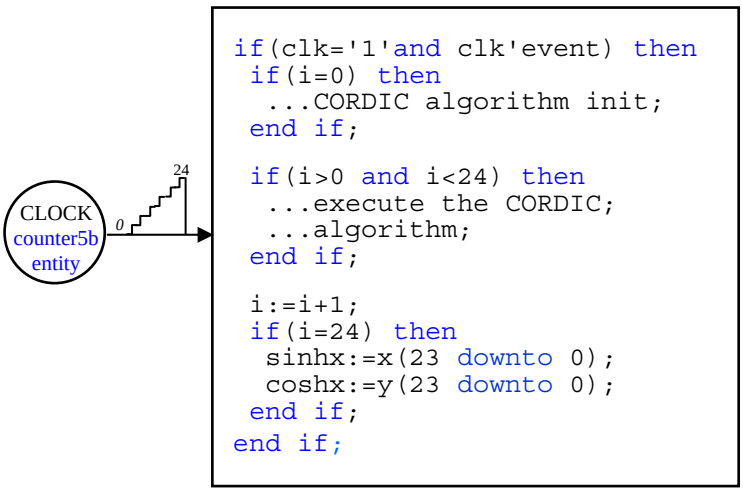

Fig. 3. Operating principle of the sigmoid optimized algorithm obtained using an external counter.

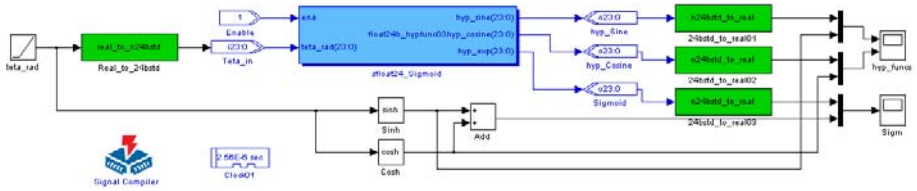

Fig. 4. Hyperbolic and Sigmoid functions Simulink® simulation layout.

In the following is shown the obtained sigmoid function compared to the same function built with Simulink® blocks. It is important to underline that the Matlab® operates with double precision floating (64 bit) point numbers. The simulations have been performed using the ALTERA® DSP Builder tool, in particular using the HDL import block, as depicted in Fig. 4.

Fig. 5 shows the simulation results when the sampling time T_s has been fixed to a 2.56 µS, in this case the output error varies in the range ±0.006 about. Other simulations have been performed with different sampling times. In all the tests, results that the entire system has latency time at maximum of 6 clock cycles.

The system presents a stable performance comparing to the any external disturbance.

Two additional routines, green blocks depicted in Fig. 4, allow to test the Artificial Neuron performances. These routine, convert respectively a given floating point number in to *sfloat24* number layout format and vice versa.

These routine, due to their complexity, have been written in C/C++ language and implemented as S-Function as depicted in [10].

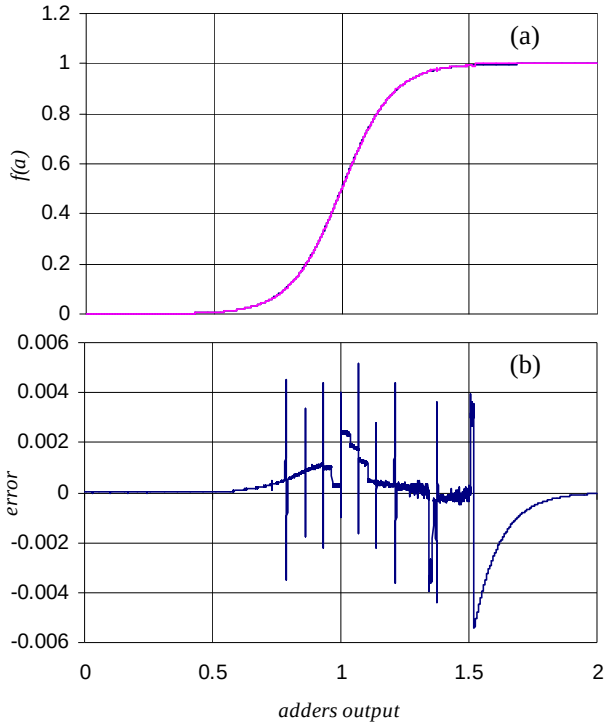

Fig. 5. Simulation results – (a) Sigmoid activation function and (b) the absolute error between the sigmoid generated by the *sfloat24* math library and one generated by Simulink® math function block.

As example of a decision system credit the situation of a current account holder that requires a loan at proper credit institute with its risk evaluation has been taken into consideration.

The risk is represented by the variable *R*. The variable x_0 indicates whether the current account holder has got (value = 1) a real estate property or hasn't (value = -1). The current account holder has got a mortgage loan ($x_1 = 1$) or hasn't ($x_1 = -1$), or ($x_1 = 0$) if he does not own any property. The variable x_2 represents the availability of a profit (value = 1) or not (value = -1) if the profit is non-existent or insignificant ($x_2 = 0$). The

loan applicant has a good behaviour (x_3 = 1), middle (x_3 = 0) or bad (x_3 =-1) with the credit institute. The case study is summarized in the following table:

Table 1. Training set of the implemented ANN, in this case, only for simplicity, are considered 12 examples, 13 and 14 represent the validation sets.

N.	x_0	x_1	x_2	x_3	O	R
1	1	1	0	1	0.5	0.5
2	1	1	1	0	0.5	0.5
3	1	1	-1	0	0	1
4	1	1	0	-1	0	1
5	1	-1	0	1	1	0
6	1	-1	1	0	1	0
7	-1	0	1	0	0.5	0.5
8	1	-1	-1	-1	0	1
9	-1	0	1	1	1	0
10	1	-1	-1	0	0.5	0.5
11	1	-1	1	-1	0.5	0.5
12	-1	0	-1	1	0.5	0.5
13	-1	0	0	0	0	1
14	-1	0	-1	-1	0	1

where O represents the desired output of the implemented ANN

Training, using the first 12 examples of Table 1, an ANN constituted only by 3 neurons in the hidden layer and 1 neuron in the output layer, the maximum error, difference between desired output and the actual ones, reached is 0.1%.
The examples 13 and 14 are used as validation set.

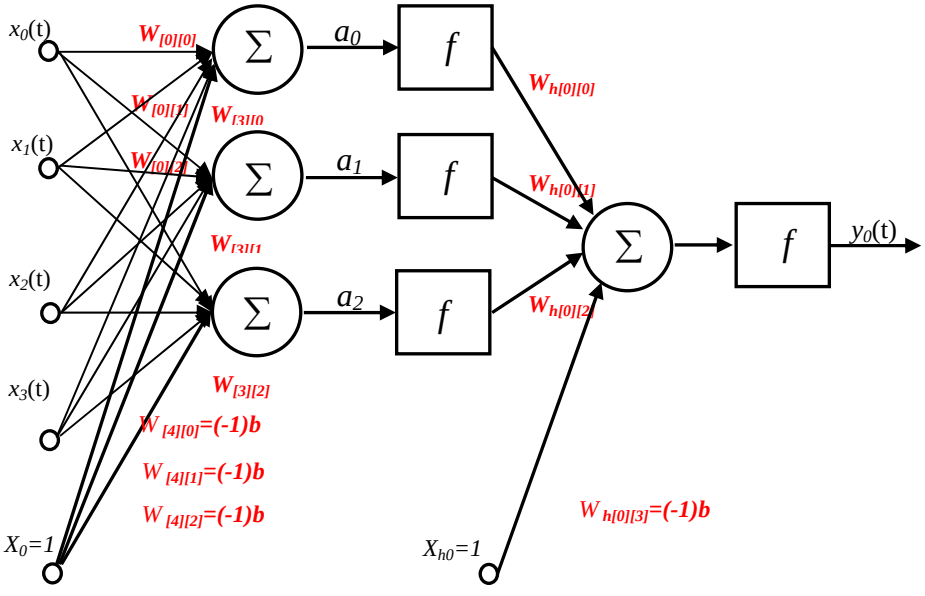

Fig. 6. ANN used for the case study depicted in the Table 1.

The device occupation of the entire ANN of Fig. 6, on the above mentioned FPGA device, is equal to 92% of logic elements.

The risk of the credit institute is simply evaluated as the opposite of the desired output as depicted in the last column of the Table 1.

Table 2. ALTERA® Quartus II report of the implemented ANN.

Total logic elements	22,757/24,624 (92%)
• Total combinational functions	22,734/24,624 (92%)
• Dedicated logic registers	2,236 (9%)
Total registers	2236

To examine more complex situations it is necessary to implement ANN with a major number of neurons. With the used low cost FPGA device this operation is not possible, because it is not capable to accommodate other neurons; this is depicted in the Table 2.

4 Conclusions

The obtained results show, through an optimized algorithm of the sigmoid function, the validity and the feasibility of the implementation of a complete trained ANN, using the developed *sfloat24* math library.

The speed execution or latency of the ANN can be precisely controlled with the amount of reuse of *sfloat24* arithmetic elements.

From the angle of theory combined with the practice, in this paper have been analysed two aspects: the implementation of an ANN on a FPGA device and the profession and causation of credit risk in bank and/or a credit institute. The depicted concepts lead to the following conclusions:

- Back-propagation neural networks are good candidate for the applications concerning in loans risk evaluation.
- Fault-tolerant ability. Because the network knowledge information adopts the distributed memory topology, the individual unit's damage cannot cause a mistake output.

Acknowledgement

The authors would like to thank the anonymous reviewer for his valuable suggestions, useful to improve this work.

References

1. M.A. Banuelos, j. Castillo Hernandez, S. Quintana Thierry, R. Damian Zamacoma, J. Valeriano Assem, R.E. Cervantes, R. Fuentes Conzalez, G. Calva Olmos, J.L. Perez Silva, Implementation of a Neuron Using FPGAS. Journal of Applied Research and Technology, Vol. I numero 003, 3 October 2003, Universitad Nacional Autonoma de Mexico Distrito Federal, Mexico, 248 - 255.

2. Phil Lapskey, Jeff Bier, Amit ShoHam, Edward A. Lee ,DSP Processor Fundamentals – Architecture and features, IEEE PRESS, 1997.
3. Shu-Fang Zhao, Li-Chao Chen, The BP Neural Networks applications in Bank Credit Risk Management System, ICCI '09. 8th IEEE International Conference on Cognitive Informatics, 527 – 532.
4. Yufang Wang, Hongsen Yan and Xiangang Meng, Matching Decision Model for Self-adaptability of Knowledge manufacturing System, (ICIST), International Conference on Information Science and Technology, 891 – 895, 2011.
5. Zhou H., Wang J., Qiu Y, Application of the Cross Entropy Method to the Credit Risk Assessment in an Early Warning System, International Symposiums on Information Processing (ISIP), 2008, 728-732, 23-25 May 2008.
6. Quer G., Meenakshisundaram H., Tamma B., Manoj B.S., Rao R., Zorzi M., A Cogni-tive Network Inference through Bayesan Network Analysis, Global Telecommunica-tions Conference (GLOBECOM 2010), 2010 IEEE, 1-6, 06-10 December 2010.
7. Xiu-Li Pang, Yu-Qiang Feng, An Improved Economic Early Warning Based on Rough Set and Support Vector Machine, International Conference on Machine Learning and Cybernetics, 2444-2449, 13-16 August 2006.
8. M.C. Miglionico, F. Parillo, Modelling a neuron using a custom math library *sfloat24* – Implementation of a sigmoid function on a FPGA device –, ISHAP Conference Sorrento Italy: 15 – 18 June 2011, http://204.202.238.22/isahp2011/dati/autor.html, Online Proceedings ISSN 1556-8296, Proceedings of the International Symposium on the Analytic Hierarchy Process for Multicriteria Decision Making, Publication date: 15 June 2011.
9. M.C. Miglionico, F. Parillo, A Current Hysteresis Controller for Reduction of Switch-ing Losses in a Full-Bridge Inverter – FPGA implementation by using a custom devel-oped 24 bit Floating Point Math Library –. IEEE Conference UPEC 2011, Soest, Ger-many,1-6, 05-08 September 2011.
10. M.C. Miglionico, F. Parillo, FPGA implementation of *sfloat24* digital PI. IEEE Conference PEDES 2010 Power India, New Delhi, 20-23 December 2010.
11. IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE 754 1985.
12. W. Kahan, Lecture notes on the Status of IEEE Standard 754 for Binary Floating Point Arithmetic. Electrical Engineering and Computer Science – University of California Berkeley CA 94720-1776, 31 May 1996.
13. C. Attaianese, F. Parillo, G. Tomasso, Dual Boost High Performances control strategy on a Power Factor Correction (PFC) implementation by using a 24 bit custom floating point library. Journal of Electrical Engineering [http://www.jee.ro], Vol. 10 edition 4, 23 December 2010.
14. Samuel Ginsberg, Compact and Efficient Generation of Trigonometric Functions using a CORDIC algorithm. Cape Town, South Africa, January 2002.
15. M.C. Miglionico, F. Parillo, A BP Neural Network Application in Bank Credit Risk Management System using a sfloat24 custom math library – FPGA implementation –, A.M.A.S.E.S. Meeting, XXXV Edition, September 15-17 2011, Pisa, ITALY.