International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-4 Issue-1, June 2014

Comparison Between Two Hardware Implementations of a Formal Neuron on FPGA Platform

Boussaa Mohamed, Bennis Abdelattif, Atibi Mohamed

Abstract*— The formal neuron is equivalent to a simple processor that performs a series of mathematical operations more or less complex on real data. The chosen representation to encode these data is the 32 bits floating point representation; this makes possible to achieve satisfactory precision in calculation. This paper presents a hardware comparison between two formal neurons, one is associated with the sigmoid activation function and the other to the gaussian activation function. This comparison is designed firstly to compare the hardware results obtained respectively from these two implementations with software results, and secondly, to make comparison between the two hardware implementations in terms of the consumed material resources and execution time. These neurons are implemented by using a number of specific blocks called megafunction, on an FPGA platform of Altera DE2-70 which offers several advantages, including flexibility, efficiency and speed.

Index Terms*—formal neuron, FPGA, hardware resources, execution time, mega function.

I. INTRODUCTION

The formal neuron is the basic element for the construction of all artificial neuron networks; the use of these networks gives satisfactory solutions in wide applications of image processing, robotics and pattern recognition. The formal neuron function is to simulate a few tasks of biological neuron in the human brain, such as memorization, learning and parallel functioning. But these features are still far from the human neuron performances, like synapse sharing and membrane activation [1]. The formal neuron implementation requires a platform which is capable of implementing the performances of this calculation unit (formal neuron), the most used platform is "Field Programmable Gate Array (FPGA)" which has demonstrated its capabilities in its hardware implementation. These capabilities are seen in several applications in the real world, like implementation of complex control algorithms of the high speed robot movements [2], the efficient production of multipoint random distributed variable [3], the development of hardware platforms and software dedicated to the car industry [4] or in the energy field [5].

Manuscript published on 30 June 2014.
***Correspondence Author(s)**
Mr. Mohamed Boussaa, Hassan II - Mohammedia – Casablanca University, Laboratory of Information Processing, Cdt Driss El Harti, BP 7955 Sidi Othman Casablanca, 20702, Morocco, USA.
Dr. Abdellatif Bennis, Hassan II - Mohammedia – Casablanca University, Laboratory of Information Processing, Cdt Driss El Harti, BP 7955 Sidi Othman Casablanca, 20702, Morocco, USA.
Mr. Mohamed Atibi, Hassan II - Mohammedia – Casablanca University, Laboratory of Information Processing, Cdt Driss El Harti, BP 7955 Sidi Othman Casablanca, 20702, Morocco, USA.

The major problem frequently encountered in the hardware implementation of a formal neuron is the use of activation functions with approximations, which has a great impact on the precision of calculations made by the neuron [6]. First, as a solution to this problem, this paper proposes the use of megafunctions provided by the FPGA platform constructors which make it possible to implement complex operations to handle real format data coded in 32 bits floating point without making calculating approximations. Secondly, this paper presents a comparative study between two formal neurons, one of them with a sigmoid activation function and the other with the gaussian function. The paper is organized as follows: The second section examines the background documents of different hardware architectures. The third section presents a theoretical study of formal neuron with different activation functions as well as the existing data formats. The fourth section is dedicated to the Hardware implementation details of our two formal neurons. The fifth section presents the tests and comparisons between the two implementations in terms of speed and capacity, and the last section is dedicated to the conclusions.

II. RELATED WORKS

Several approaches of implementation of formal neuron in the FPGA platform were made. However, huge challenges were encountered during these implementations among which the problem of using approximations in the implementation. As it is shown in Fig 1, in 2014 (SAMY ELMOUKHLISS) has implemented, in his paper [7], a neuron architecture using the sigmoid activation function. The challenge of this architecture is the use of approximation for the implementation of the transfer function.

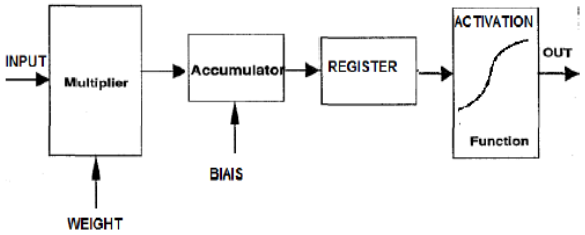

Fig.1: Neuron Structure

In 2001 (DENIS F. WOLF) has designed an architecture that brings together blocks and a Softcore NIOS microprocessor. This implementation has as a challenge the conversion of floating point numbers into integers and replacing the sigmoid by a linear approximation [8].

In 2007, Antony W. Savich has done a detailed study on FXP and FLP representations in comparison to the effect of the accuracy on the implementation of the multilayer perceptron. The obstacle found in this study was associated with the implementation of formal neuron with the sigmoid activation function (fig 2), which requires complex operations such as the exponential and division [6].

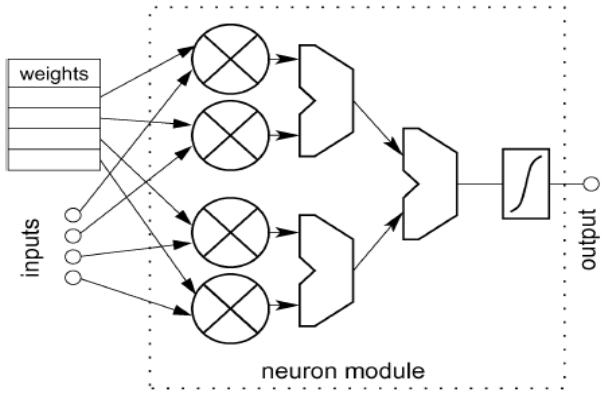

Fig. 2: Architecture for Neuron

III. THEORY OF FORMAL NEURON AND MEGAFUNCTIONS

A. *Formal Neuron*

The formal neuron is a model that simulates the biological neuron functioning. This neuron can perform a mathematical operations series more or less complex as the summation and multiplication. The first formal neuron model proposed by McCulloch and Pitts in 1949 describes in detail the operations fulfilled. The artificial neuron consists of several inputs which receive two vectors; a parameter vector of the object studied and another vector of connection weights called synaptic weights, and a single output. The values of these synaptic weights are already set during the learning phase. The formal neuron performs a weighted summation generally between the two input vectors, and the result of this operation is transferred to the output through an activation function.

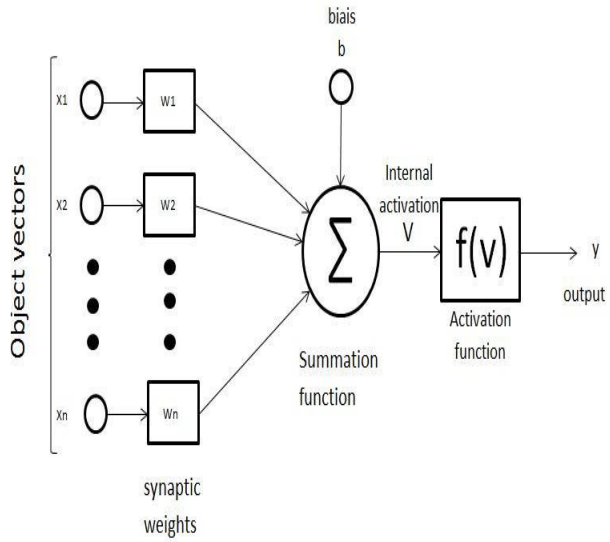

Fig.3: Schema of a Formal Neuron

With:

X1……Xn: the neuron object vector.

W1……Wn: synaptic weights contained in the neuron.

∑: a function which calculates the sum of the multiplication between the object vector and the synaptic weights according to equation (1).

B: the bias of the summation function.

F (V): the neuron activation function.

Y: the formal neuron output.

First the neuron calculates the internal activation value in accordance with the input vector and the vector of synaptic weights according to equation (1) and secondly the output value according to a transfer function which limits the neuron output in the range [0, 1].

$$V = \sum_{i=0}^{n} Wi.Xi \qquad (1)$$

The sigmoid and gaussian functions are the two most commonly used activations functions in formal neurons, the first (figure 4) is defined by:

$$Y(X) = f(V(X)) = \frac{1}{1+e^{-V(X)}} \qquad (2)$$

The sigmoid is a function with values in the range [0,1], which allows to interpret the output of the neuron as a probability. Moreover, it is not polynomial and is infinitely continuous and differentiable; the second (fig 4) is defined by:

$$Y(X) = f(V(X)) = e^{-x^2/\sigma^2} \qquad (3)$$

The gaussian is a function that depends on the point's center of the input space and width, besides, it is a continuous and differentiable function.

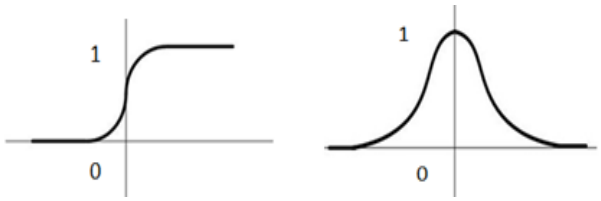

Fig.4: Activation Functions

B. *Megafunction*

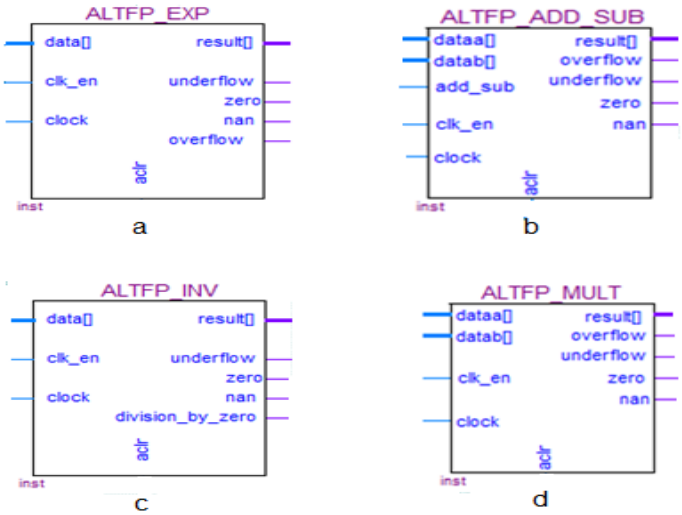

Fig.5: Block MegaFunction

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

A megafunction is a specific block that includes a set of components (memory, registry, multiplier...). This block is described in hardware description language (VHDL) and it has as an objective the realization of a transaction or a series of more complex arithmetic operations. These blocks have become effective methods of designs to make complex applications in various fields such as robotics and image processing. These blocks are IP (Intellectual Properties) proposed by the simulation software Quartus II 9.1, which is optimized for Altera circuits. The IPs are collected in libraries, including « Library of Parameterized Modules » (LPM), containing the most complex functions that will be useful in the design of formal neuron. These megafunctions have helped to save valuable time during the design, by avoiding coding a new logical block. As a structure, these blocks offer the opportunity to adjust their size by adjusting some parameters and the access to specific features of the architecture in the memory; DSP blocks, shift registers, and other simple and complex functions. Among these megafunctions, multiplication block of two floating point numbers (Fig 5d), the addition of two floating point numbers (Fig 5b), inversion a floating point number (Fig 5c), exponential (Fig 5a) ... etc...

IV. PROPOSED DESIGN

The design of a formal neuron is divided into two parts; the first part consists in calculating the weighted sum of the input vector and the synaptic weight vector (1), it represents the internal activation of the neuron. The second part calculates the output of the neuron through an activation or transfer function, this paper deals with the case of two formal neurons; the first with the sigmoid activation function and the second with the Gaussian function.

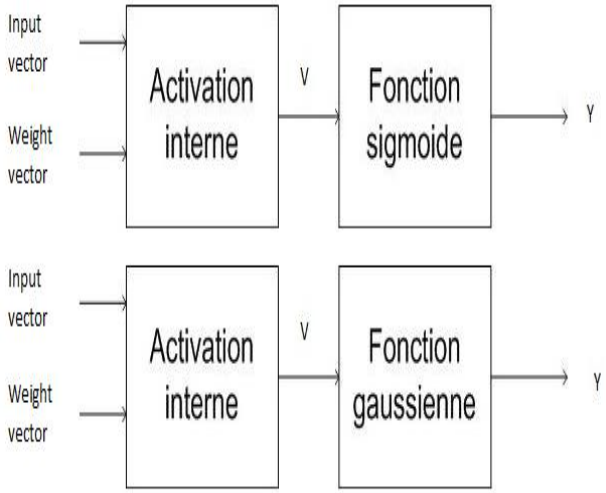

Fig.6: Design of the two Formal Neuron

A. Implementation Detail of the Internal Activation

For the two neurons; the sigmoid and the gaussian neurons, the implementation of the internal activation part is the same. It consists of a set of interconnected megafunction blocks in order to calculate the weighted sum (1) of the two vectors (input vector and synaptic weight vector). For example in the case of a neuron with four inputs, the calculation of the internal activation requires 4 multiplication blocks megafunctions, that allow to multiply the elements of each vector (input vector and synapse weight vector), and then adding 3 megafunction blocks that calculate the sum of these multiplications to generate an output V, entitled internal activation formal neuron (Fig 7).

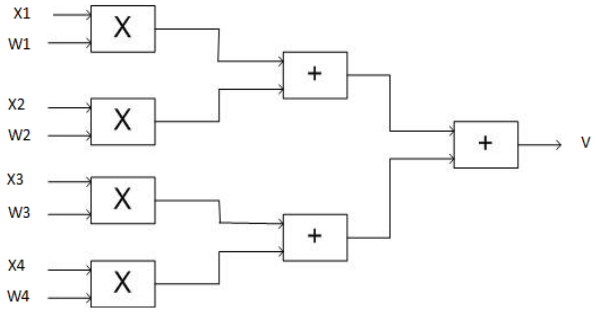

Fig.7: Design of Internal Activation

B. Implementation Detail of the two Functions of Activations

This implementation consists in transferring the output of the internal activation V to the output neuron, using an activation function. The chosen activations functions in this paper are the sigmoid and gaussian activations that are both continuous and differentiable. The sigmoid function implementation requires a set of megafunction blocks like the multiplication, addition, the exponential and the division. The internal activation propagates through this activation function via a series of cascaded blocks in the following order: multiplication, exponential, addition and division (Fig 8).

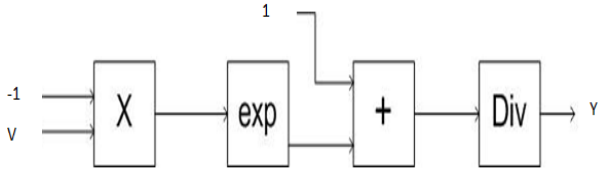

Fig.8: Design of Sigmoid Function

The implementation of the gaussian activation function consists in using a series of cascaded megafunction blocks in the following order: multiplication, multiplication and exponential illustrated in Fig 9.

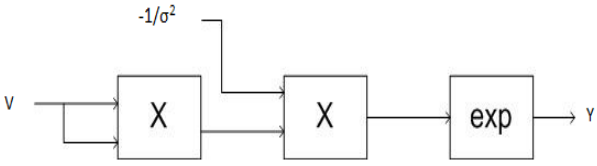

Fig. 9: Design of Gaussian Function

C. Complete implementation

In order to fully implement the formal neuron with the sigmoid activation function (Fig 8) and the gaussian activation function (Fig 9), there should be a regrouping of, on the one hand, the internal activation and the sigmoid function to construct the first neuron (Fig 10) and, on the other hand, the internal activation and the gaussian activation function to have the second neuron (Fig 11).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

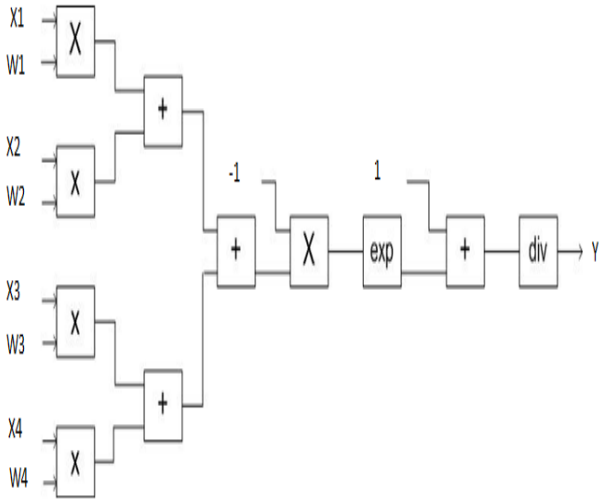

Fig.10: Design of Sigmoid Neuron

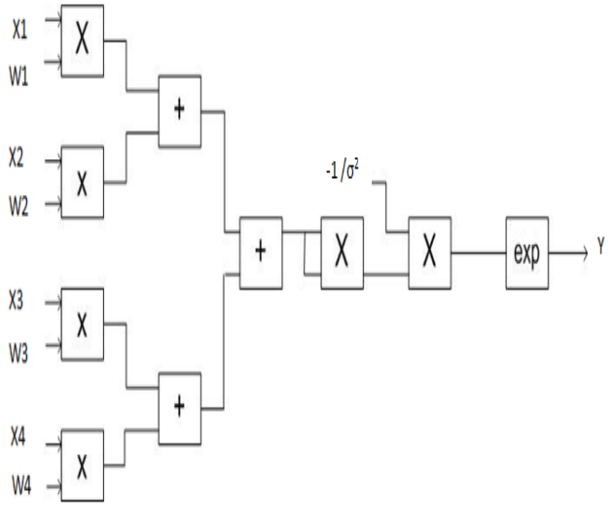

Fig.11: Design of Gaussian Neuron

V. TESTS AND RESULTS

This section aims at two objectives; the first is to test the performance of the two proposed formal neurons in terms of calculation precision, and the second is to compare the two designs that are implemented without approximation in capacity and time of execution. The chosen platform to perform these two tests is the Altera FPGA CYCLONE II EP2C70F896C6.

A. *The First Test*

In this test, the formal neuron with sigmoid activation function or the gaussian activation function receives in its input two vectors (input vector and synaptic weight vector) which have random values, to generate an output; this output will be compared with the results of a software calculation in developed language C + +. Tables (I and II) show that the calculation obtained with the two neurons is efficient; this stems from the use of megafunction blocks which give the possibility to implement complex functions without making approximation and also to the floating point representation of the data.

Table I
Hardware and Software Result (Sigmoid)

Input vector				Weight vector				Sigmoid Hardware result	Sigmoid Software result
X1	X2	X3	X4	W1	W2	W3	W4		
1	1	0.5	0.5	1	0.5	-0.5	-1	0.67917	0.679178
1	0.5	-0.5	-0.5	1	0.5	-0.5	-1	0.88	0.88
0.5	0.5	0.5	0.5	1	0.5	-0.5	-1	0.5	0.5
0.5	0.5	-0.5	-0.5	1	0.5	-0.5	-1	0.817	0.817574

Table II

Input vector				Weight vector				Gaussian Hardware result	gaussian Hardware result
X1	X2	X3	X4	W1	W2	W3	W4		
1	1	0.5	0.5	1	0.5	-0.5	-1	0.976	0.977
1	0.5	-0.5	-0.5	1	0.5	-0.5	-1	0.85	0.85214
0.5	0.5	0.5	0.5	1	0.5	-0.5	-1	1	1
0.5	0.5	-0.5	-0.5	1	0.5	-0.5	-1	0.91	0.91393

Hardware and Software Result (Gaussian)

B. *The Second Test*

The second test aims at comparing the two architectures of two neurons in terms of capacity (material resources in the FPGA Embedded Platform) and calculation time by varying the number of elements of the input vector and the number of elements of the synaptic weights (4, 8,16).

Table III shows that the formal neuron architecture with sigmoid activation function occupies more resources and requires more time (Fig12) than the formal neuron architecture with gaussian function. These results are related to the number of Megafunction blocks forming each architecture and the execution time and material resources used by each Megafunction block as shown in table IV.

Table III
Comparison of Resources and Time Execution

	Addition	Multiplication	Exponential	division
Total logic elements (%)	1101 (2%)	939 (1%)	1166 (2%)	764 (1%)
Total combinational functions (%)	912 (1%)	856 (1%)	1089 (2%)	529 (1%)
Total registers (%)	789 (1%)	530 (1%)	491 (1%)	696 (1%)
Execution time	140 ns	50 ns	170 ns	200 ns

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

Table IV
Resources and Time Execution of Mega Function

	4 Inputs		8 Inputs		16 Inputs	
	Sigmoid	Gaussian	Sigmoid	Gaussian	Sigmoid	gaussian
Total logic elements (%)	10430 (15%)	9148 (13%)	11433 (17%)	10174 (15%)	15444 (23%)	10841 (16%)
Total combinational functions (%)	9088 (13%)	8551 (12%)	9954 (15%)	9464 (14%)	13429 (20%)	10077 (15%)
Total registers (%)	6635 (10%)	5437 (8%)	7331 (11%)	6150 (9%)	10038 (15%)	6606 (10%)
Execution time	890 ns	600 ns	1030 ns	740 ns	1170 ns	880 ns

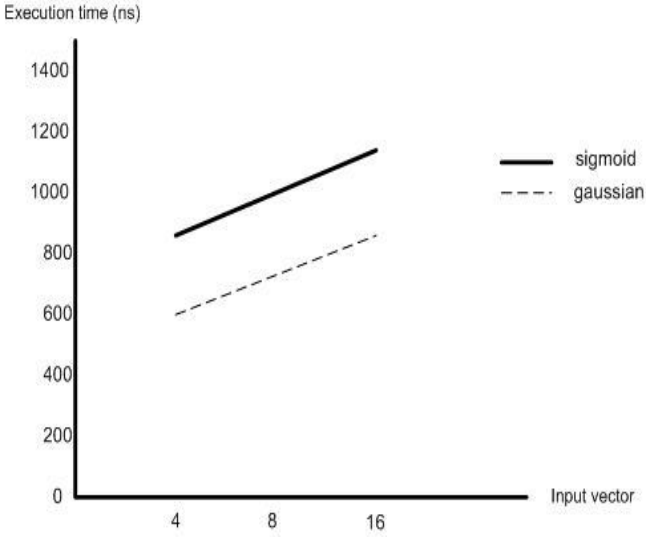

Fig.12: Comparison of the Sigmoid and the Gaussian (Execution Time)

VI. CONCLUSION

This paper aimed at implementing two different architectures of formal neuron, using both the sigmoid and gaussian activation function without making calculation approximations .The objective behind this is better precision and minimal use of material resources of the FPGA board and this is thanks to the calculation performances provided by megafunction blocks. The obtained results from the comparison between the formal neurons on the basis of the sigmoid and Gaussian activation functions pave the way for the implementation on a FPGA platform of complete and powerful network of artificial neuron in terms of performances and used resources.

REFERENCES

1. Lin, C. J., & Lee, C. Y. (2011). Implementation of a neuro-fuzzy network with on-chip learning and its applications. Expert Systems with Applications, 38(1), 673-681.
2. Shao, X., & Sun, D. (2007). Development of a new robot controller architecture with FPGA-based IC design for improved high-speed performance. Industrial Informatics, IEEE Transactions on, 3(4), 312-321.
3. Bruti-Liberati, N., Martini, F., Piccardi, M., & Platen, E. (2008). A hardware generator of multi-point distributed random numbers for Monte Carlo simulation.Mathematics and Computers in Simulation, 77(1), 45-56.
4. Salewski, F., &Kowalewski, S. (2008). Hardware/software design considerations for automotive embedded systems. Industrial Informatics, IEEE Transactions on, 4(3), 156-163.
5. Bueno, E. J., Hernandez, A., Rodriguez, F. J., Girón, C., Mateos, R., &Cobreces, S. (2009). A DSP-and FPGA-based industrial control with high-speed communication interfaces for grid converters applied to distributed power generation systems. Industrial Electronics, IEEE Transactions on, 56(3), 654-669.
6. Savich, A. W., Moussa, M., & Areibi, S. (2007). The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study. Neural Networks, IEEE Transactions on, 18(1), 240-252.
7. El Moukhlis, S., Elrharras, A., &Hamdoun, A. FPGA-Based Handwritten Signature Recognition System. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-3, Issue-11, April 2014
8. Wolf, D. F., Romero, R. A., & Marques, E. D. U. A. R. D. O. (2001, November). Using embedded processors in hardware models of artificial neural networks. InVSimposioBrasileirode automaçãointeligente, Brazil.
9. Bosque, G., del Campo, I., &Echanobe, J. (2014). Fuzzy systems, neural networks and neuro-fuzzy systems: A vision on their hardware implementation and platforms over two decades. Engineering Applications of Artificial Intelligence..
10. Rostro-Gonzalez, H., Cessac, B., Girau, B., & Torres-Huitzil, C. (2011). The role of the asymptotic dynamics in the design of FPGA-based hardware implementations of gIF-type neural networks. Journal of Physiology-Paris, 105(1), 91-97.
11. Tisan, A., &Cirstea, M. (2013). SOM neural network design–A new Simulink library based approach targeting FPGA implementation. Mathematics and Computers in Simulation, 91, 134-149.
12. Online in: http://www.altera.com.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)