PRECISE CALCULATION UNIT BASED ON A HARDWARE IMPLEMENTATION OF A FORMAL NEURON IN A FPGA PLATFORM

Mohamed ATIBI, Abdelattif BENNIS, Mohamed BOUSSAA
Hassan II - Mohammedia – Casablanca University, Laboratory of Information Processing,
Cdt Driss El Harti, BP 7955 Sidi Othman Casablanca, 20702, Maroc

ABSTRACT
The formal neuron is a processing unit that performs a number of complex mathematical operations on real format data. These calculation units require hardware's architectures capable providing extremely accurate calculations treatments. To arrive upon more accurate hardware architecture in terms of the calculation, the new proposed method uses data coding in single precision floating point. This allows handling of infinitely small and infinitely large data and; consequently, a diverse field of application. The formal neuron implementation requires an embedded platform whose implementation must be flexible, efficient and fast. This article aims at presenting in detail a new precise method to implement this calculation unit. It uses a number of specific blocks described in VHDL hardware description language in an embedded FPGA platform. The data handled by these blocks are coded in 32-bit floating point. The implementation of this new method has been developed and tested on an embedded FPGA platform of Altera DE2-70. The calculation results on the platform and those obtained by simulation are very conclusive.

KEYWORDS: *FPGA, precision, formal neuron, Floating point, HARDWARE implementation.*

I. INTRODUCTION

The artificial neurons networks (ANN) present heuristic models whose role is to imitate two basic skills of the human brain:

- Learning from examples.
- The Generalization of knowledge and skills learned through examples to others which are unseen in the learning phase [1].

The ANN is configured through a learning process for a specific application, this process involves adjusting the synaptic connections between neurons. These models are used in a wide range of applications such as patterns recognition, the classification, robotics, signal and image processing... etc. For example in the field of information processing, these models simulate the way biological nervous systems process information.

The ANNs are networks based on a simplified model of neuron called formal neuron, this model can perform a number of functions of the human brain, like the associative memory, supervised or unsupervised learning, parallel functioning…etc. Despite all these features, formal neuron is far from having all the performances of biological neurons that human being possess like synapse sharing and membrane activation [2].

A major problem of the use of formal neurons in the ANNs, is the lack of Hardware method to implement in embedded platforms [3] [4]. The respect of, on the one hand, the neurons architecture, and on the other hand, the format of neurons manipulated data which takes often the form of a real number has a great impact on the calculation results of this neuron and their precision. This is especially true in the case of an application requires an architecture consisting of a large number of neurons.

Several attempts have allowed implementing formal neurons as integrated circuits. The field-programmable gate array (FPGA) is the preferred reconfigurable hardware platform. It has proven its capacity through several applications in various fields [5]; implementing complex control algorithms for high speed robot movements at [6], efficient production of multipoint random distributed variable [7], the design of hardware platforms / software for the car industry [8], or in applications of energy production [9]. However, the design of the neuron presents several challenges, the most important is to choose the most effective format of arithmetic representation to ensure both good precision and processing speed.

The article examines in detail the precision of formal neuron design with a sigmoid activation function on FPGA, the architecture is tested with a floating point arithmetic format by using an advanced hardware description language (VHDL).

The article is organized as follows. Section II provides a global overview on different Hardware architectures. Section III presents a theoretical study of the formal neuron with its different activation functions, and the existing data formats. Section IV dedicated to the details of the implementation of formal neuron Hardware detail. Section V presents the tests of efficiency of this implementation. Finally Section VI presents the conclusion.

II. RELATED WORK

Several approaches of architecture have been proposed for hardware implementation of the formal Neuron in a platform such as FPGA, as shown in Figure 1. In 2007, Antony W. Savich has made a detailed study on FXP and FLP representations and the effect of the accuracy on the implementation of the multilayer perceptron. The obstacle found in this study was related to the implementation of formal neuron with the sigmoid activation function which requires complex operations such as the exponential and division [3].

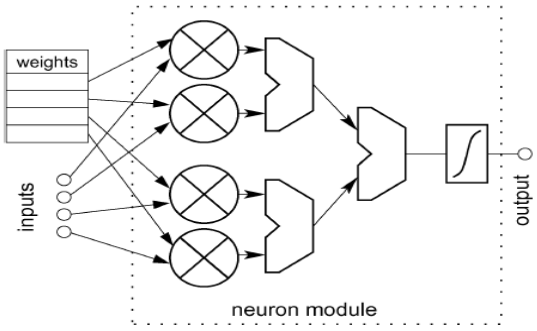

Figure 1. Neuron structure

In 2011 (Horacio Rostro-Gonzalez) has presented [4] a numerical analysis of the role of asymptotic dynamics in the design of hardware implementations of neural models like GIF (generalized integrate-and-fire). The implementation of these models was carried out on an FPGA platform with fixed-point representation (figure 2).

In 2012 (A. Tisan) has introduced an implementation method of the learning algorithm of networks artificial neural on FPGA platform. The method aims at constructing a network of specific neurons using generic blocks designed in the math Works Simulink environment. The main features of this solution are mainly the implementation of the learning algorithm on high capacity chip of reconfiguration and functioning of real time constraints [5].

In 2011 (Cheng-Jian Lin) has presented in his article the hardware implementation of neurons and neural networks, with a representation of real numbers in fixed-point format, using the perturbation method as a method of network's learning [2].

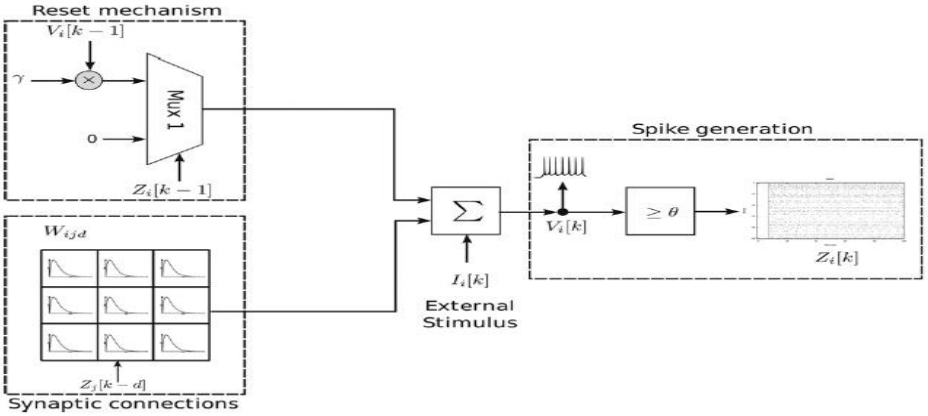

Figure 2. Architecture for a single neuron

III. THE FORMAL NEURON THEORY

3.1. History

In 1943, McCulloch and Pitts have proposed a model that simulates the functioning of biological neuron. This model is based on a neurobiological inspiration, which is a very rudimentary modeling the neurons functioning, in which the accumulation the neuron synaptic activities are ensured by a simple weighted summation [1]. The interconnections of a set of such units provide a connectionist neural system, also referred to as neural network.

These networks can perform logical functions, complexes arithmetic and symbolic. Figure 3 shows the schema of a formal neuron:

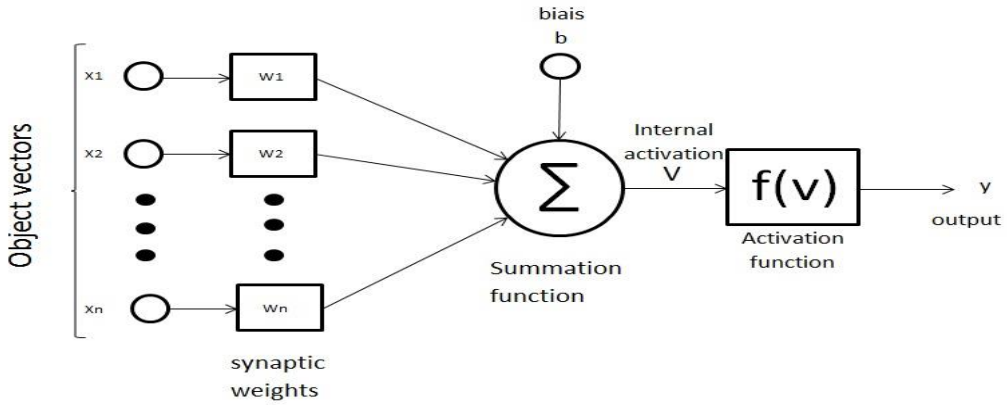

Figure 3. Schema of a formal neuron

With:

X1……Xn: the neuron vector object.

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.

A "formal neuron" (or simply "neuron") is a nonlinear algebraic and bounded function. In fact the neuron receives at its input an object vector of which each object parameter is multiplied by a synaptic weight. The sum of these multiplications and the bias constitute internal activation:

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

V will gets, at the end, to an output through an activation function. There are many activations functions of a formal neuron, the most used are:

- The threshold function (Figure 4): In 1949, Mculloch and Pitts have used this function as an activation function in their formal neuron models. Mathematically this model of neuron generates an application from Rn to {0,1}.

$$Y(X)=f(V(X))=\begin{cases}1 & \text{si } V(X)\geq \Theta \\ 0 & \text{sinon}\end{cases} \quad (2)$$

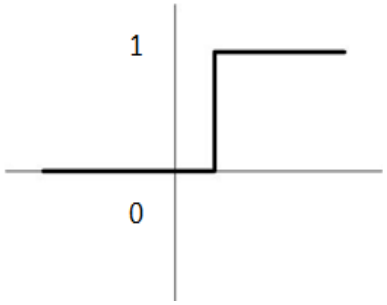

Figure 4. Threshold function

- The sigmoid function (Figure 5): this is a function proposed by by Rosenblat in 1962, also called logistic function, defined by:

$$Y(x)=f(V(x))=\frac{1}{1+e^{-V(X)}} \quad (3)$$

It is a function with values in the interval [0,1], which allows to interpret the output of the neuron as a probability. In addition, it is not polynomial and is infinitely continuously differentiable.

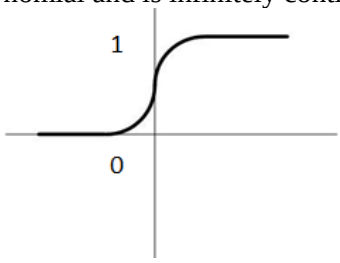

Figure 5. Sigmoid function

- The Gaussian function (Figure 6): It is a function proposed in 1989 by MOODY and DARKEN in order to be used in specific networks called radial based networks (RBF). It is defined by:

$$Y(X)=f(V(X))=e^{-x^2/\sigma^2} \quad (4)$$

It is a function that depends on the center points of the input space and its width. In addition it is a continuous and differentiable function.

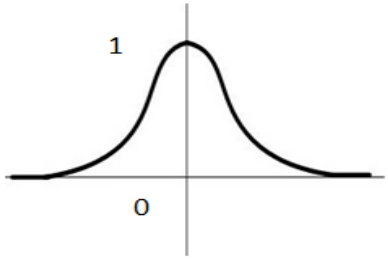

Figure 6. Gaussian function

3.2. Data formats handled by a formal neuron

The formal neuron, in most cases, makes its calculations with real numbers. To represent a real number, there are a finite number of bits and one can imagine different ways to represent it with this bit set. The two well-known methods of representation are fixed-point representation and floating point representation.

3.2.1. Fixed-point representation

This is the usual representation as is done on paper except for the sign and the decimal point. For the sign, we keep a bit for it; in general the left bit. Also the point is not represented, working with an implicit point located at a definite place.

This representation is not widely used because it has a low accuracy because of the numbers lost to the right of the decimal point. Another problem with this representation is that it does not represent very large numbers.

3.2.2. The floating point representation

Inspired by the scientific notation (ex : +1,05 x 10-6) and in which a number is represented by the product mantissa and a power of 10 (decimal) or a power of 2 (binary). To normalize the floating point representation of real numbers [3], the IEEE-754 norm is recommended by the Institute of Electrical and Electronics Engineers. It is widely used to represent real numbers. Each number is represented by:

- bit for the sign (s).
- Ne bits for the signed exponent (E).
- Nm bits for the absolute value of the mantissa (M).

Real numbers are represented either in 32 bits (simple precision), 64 bits (double precision) or 80 bits (extended).

Example of a real number represented in 32 bits:

Table 1. Floating point representation

	Bits		
Bits (31 down to 0)	31	30 - 23	22 - 0
Contents (s, E, M)	Sign (s) 0 = positive 1 = négative	Exponent (E) An 8 bits integer	Mantissa (M) An 23 bits integer

$X = (-1)^S \times 2^{E-127} 1, M$ with $0 < E < 255$

IV. IMPLEMENTATION DETAILS

This section reviews the different steps and necessary modules for the design of formal neuron with sigmoid activation function. The formal neuron module consists of many multipliers, Additionners, and a block of sigmoid activation function.

Among the problems of the design of a formal neuron in a FPGA platform with the VHDL language, is that the real numbers are not synthesized in this language, the solution which has proven this implementation is to design a formal neuron that manipulates these data, representing them in floating-point . This provides an efficiency in terms of the calculation precision.

To achieve this accuracy, blocks called mega functions , which are blocks offered by the constructors of the FPGA, have been used . These blocks are written in VHDL to handle complex arithmetic operations with floating point representation (32 or 64 bit), these blocks are useful for the calculation accuracy in the formal neuron.

4.1. Megafunctions

As the design complexity increases in a fast manner, the use of specific blocks has become an effective method of design to achieve complex applications in different domains such as robotics, signal and image processing …etc. The simulation software QUARTUS offers a number of IP (« Intellectual Properties ») synthesizing complex functions (memory, multipliers, comparators etc ...) optimized for Altera circuits. These IP, designated by the term « megafunction », are grouped into libraries, including « Library of Parameterized Modules » (LPM), containing the most complex functions that are useful for the design of formal neuron.

The use of megafunctions replacing in the coding of a new logic block saves precious time for design. In addition to the functions provided by Altera allows us to offer more efficient logic synthesis for the realization of the application. It also allows the opportunity to redimensionning the size of these

megafunctions by adjusting the settings and to access specific functionality of the architecture in the memory, DSP blocks, registers shift, and other simple and complex functions [10].
The formal design of the neuron was based on the sigmoid activation function; it's an indefinitely differentiable function.
The design detail is divided into two parts (Figure 7):
The first part: the design of the internal activation.
The second part: the design of the sigmoid activation function.

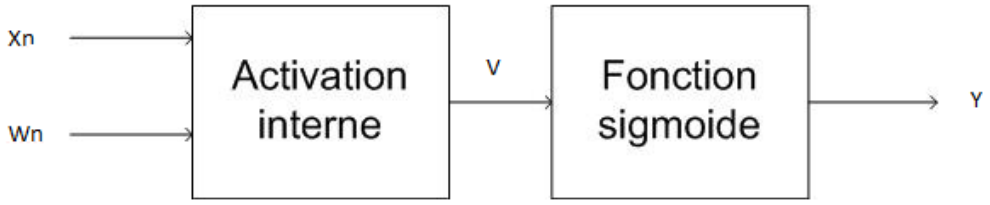

Figure 7. Design of the formal neuron

4.2. Design detail of the internal activation

The formal neuron receives as input an object vector X=(X1,X2,…..,Xn) which is represent forms to recognize in the example of application of pattern recognition and a vector of synaptic weights W=(W1,W2,…,Wn) representing the connection between the neuron and one of its inputs (Figure 8). The function of the neuron consists of calculating firstly the weighted sum of its inputs. The sum output is called internal neuron activation (1).

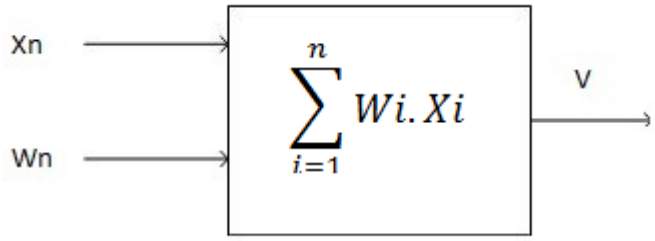

Figure 8. Internal activation

This module implements this operation using megafunctions multiplication and addition according to Equation (1).

4.2.1. Multiplication:

The multiplication block used is a megafunction block that implements the functions of the multipliers. It follows the IEEE-754 norm for representations of floating point numbers in simple precision, double precision and single extended precision. More, it allows the representation of special values like zero and infinity.
The Representation followed in this paper is the representation of single precision 32 bits as follows; this is a High Precision representation which consumes less space compared to 64 bits:

$$X = (-1)^{S} \times 2^{E-127} \times 1,M$$

The result (R) of the multiplication algorithm of two real inputs (A and B) represented in floating point used by this megafunction is calculated as follows:

$$R= (Ma \times 2^{Ea}) \times (Mb \times 2^{Eb}) = (Ma \times {}^{Mb}) \times 2^{Ea+Eb}$$

Where:

R: multiplication result.
Ma: the mantissa of a number A.
Mb: the mantissa of a number B.
Ea: the exponent of a number A.
Eb: the exponent of a number B.
Sign: (sign of A) XOR (sign of B).

4.2.2. Addition:

The addition block used is a megafunction block that implements the functions of addition and subtraction; it follows the IEEE-754 norm for representations of floating point numbers in single precision, double precision and single extended precision, with handling of selecting operation between addition and subtraction.

The result (R) of the addition algorithm of two real inputs (A and B) represented in floating point used by this megafunction is calculated as follows:

$$R = (-1)^{Sa} \times 2^{Ea} \times 1,Ma + (-1)^{Sb} \times 2^{Eb} \times 1,Mb$$

Where:

R: addition result.
Ma: the mantissa of a number A.
Mb: the mantissa of a number B.
Ea: the exponent of a number A.
Eb: the exponent of a number B.
Sa:the sign of number A.
Sb:the sign of number B.

These 2 blocks are the basis for designing the internal activation of the formal neuron. The following Figure 9 shows an example of the implementation of this internal activation.

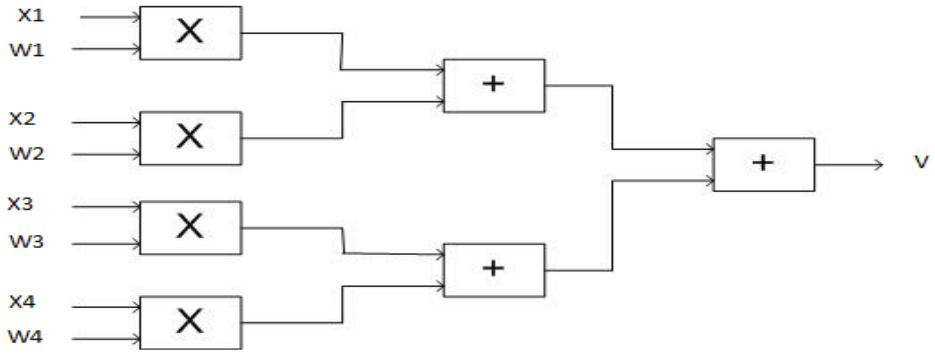

Figure 9. Design of internal function

4.3. Design detail of the sigmoid function

The second block is a transfer function called activation function. It limits the output of the neuron in the range [0,1]. The most used function is the sigmoid function (Figure 10).

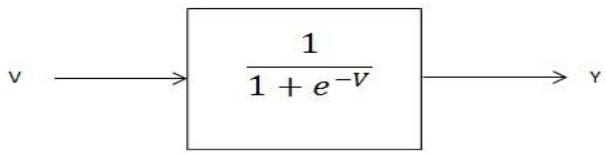

Figure 10. Sigmoid function

The implementation of this function requires a number of complex operations such as division and exponential. It requires the use of the exponential and division megafunctions.

4.3.1. Exponential and division

The used blocks of the exponential and the division are megafunction blocks that implement the functions of division and exponential. These blocks require a number of resources for their designs, the following 2 tables show these resources:

Table 2. Exponential resources

Precision	**Output latency**	**Logic usage**				**Fmax (MHz)**
		ALUTs	**Registers**	**18 bit DSP**	**memory**	
Single	17	527	900	19	0	274,07
Double	25	2905	2285	58	0	205,32

Table 3. Division resources

Precision	Output latency	Logic usage				Fmax (MHz)
		ALUTs	Registers	18 bit DSP	memory	
Single	20	314	1056	16	0	408,8
Double	27	799	2725	48	0	190,68

4.3.2. Sigmoid function implementation

The implementation of the sigmoid function in addition to these displays two blocks, the already mentioned blocks of multiplication and addition, as it is shown in the following diagram (Figure 11):

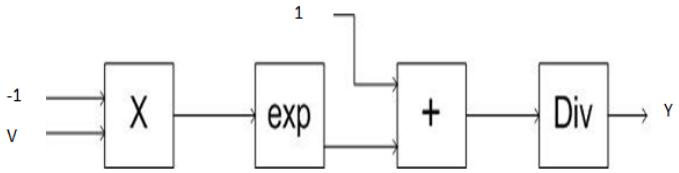

Figure 11. Design of sigmoid function

V. TEST AND RESULT

This test is designed to evaluate the precision of calculation of the formal neuron with VHDL language by comparing it with the software results. Before performing this test, it is necessary to initialize the synaptic weights. The following table summarizes the initialization (the case of 4 inputs), representing the floating point data:

Table 4. Values of the synaptic weight

	value	**Floating point representation in 32 bits (hexadecimal)**
W1	1	3F800000
W2	0.5	3F000000
W3	-0.5	BF000000
W4	-1	BF800000

These values of the synaptic weights will be used during the entire test phase for design.

The simulation of the complete implementation of a formal neuron using the sigmoid activation function, in the FPGA platform of the family Cyclone II Version EP2C70F896C6. Requires a number of steps in the simulation software Quartus II 9.1:

1. Create a new project by specifying the reference of the chosen platform.
2. Choose the Block Diagram/Schematic file.
3. Draw the model of the formal neuron by combining between required megafunctions blocks (figure 12).

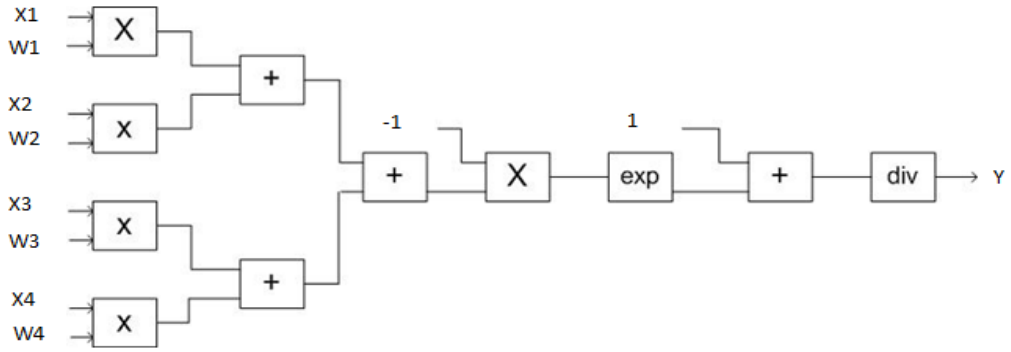

Figure 12. Design of the complete formal neuron

4. Compile the project by clicking START COMPILER.
5. Choose VECTOR WAVEFORM FILE by specifying the values of the manipulated inputs (table 4) and outputs.
6. Start SIMULATOR TOOL to simulate a module of the formal neuron.
7. View the simulation result (Figure 13).

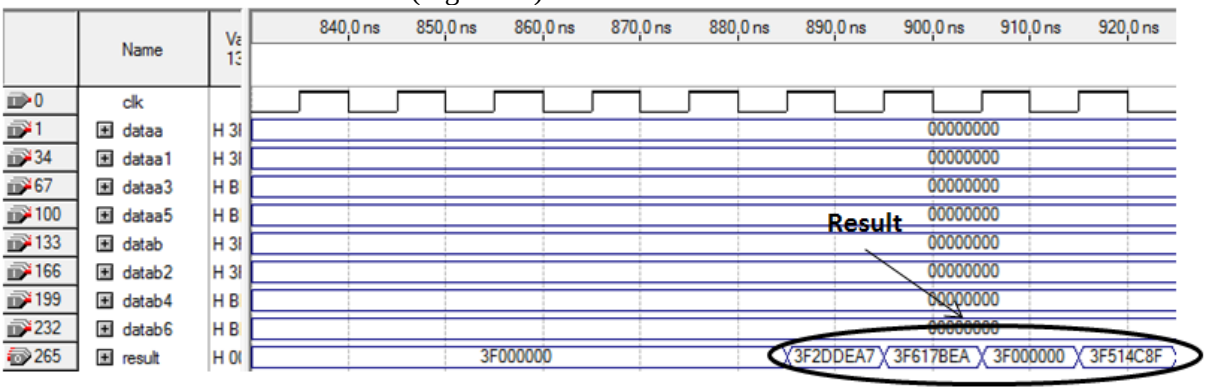

Figure 13. Test result

The following table shows the obtained results of the formal neuron test based on the sigmoidal activation function.

Table 5. Hardware and Software result

vector object				Hardware result	Software result
X1	X2	X3	X4		
1	1	0.5	0.5	0.67917	0.679178
1	0.5	-0.5	-0.5	0.88	0.88
0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	-0.5	-0.5	0.817	0.817574

4 test vectors are tested in this neuron. The table shows the output result of these 4 inputs with the synaptic weights in the table (4). These data are represented in the FPGA with floating point format at 32 bits, leading to a good precision while doing the calculation in this neuron. Moreover, the table shows also a comparison of the same neuron calculations carried in software. The result of these calculations has shown great precision thanks to the representation of floating point data. This precision is due to the use of megafunctions blocks (multiplier, additionner, exponential etc.).

VI. CONCLUSION AND FUTURE WORKS

This article has examined the technics of Hardware implementation of the formal neuron with sigmoid activation function in the FPGA platform using a floating point 32-bit format, of neuron processed data. The objective of this Hardware implementation is to materialize the formal neuron as a specific component in the calculations and can, therefore, be added to the library of the Quartus software.

For future work, this module of a formal neuron will be the base for the design of architecture of artificial neuron networks, as the architecture of the multilayer perceptron (MLP). And apply this network to various applications such as image processing, signal processing, pattern recognition...

REFERENCES

[1]. Sibanda, W., & Pretorius, P. (2011). Novel Application of Multi-Layer Perceptrons (MLP) Neural Networks to Model HIV in South Africa using Seroprevalence Data from Antenatal Clinics. International Journal of Computer Applications, 35.
[2]. 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.
[3]. 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.
[4]. 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.
[5]. 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.
[6]. 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.
[7]. 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.
[8]. Salewski, F., & Kowalewski, S. (2008). Hardware/software design considerations for automotive embedded systems. Industrial Informatics, IEEE Transactions on, 4(3), 156-163.
[9]. 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.
[10]. Online in: http://www.altera.com.

AUTHORS

ATIBI Mohamed received his master degree in information processing from Faculty of Science Ben M'sik, Hassan II University Mohammedia-Casablanca in 2013, he is preparing his PhD thesis in the same university, and his area of interest includes: application of image processing and artificial neuron networks in road safety.

BENNIS Abdellatif Is a professor of higher education at the Laboratory of Information Processing, Faculty of Science ben m'sik, , Hassan II University Mohammedia-Casablanca, and responsible for the software engineering and telecommunications team in the same laboratory.

BOUSSAA Mohamed received his master degree in information processing from Faculty of Science Ben M'sik, Hassan II University Mohammedia-Casablanca in 2013, he is preparing his PhD thesis in the same university, and his area of interest includes: application of signal processing and artificial neuron networks in the cardiac signals.