EasyChair Preprint

№ 583

# Design methodology of sigmoid functions for Neural Networks using lookup tables on FPGAs

Santiago Tomás Pérez Suárez



October 21, 2018

# Design methodology of sigmoid functions for Neural Networks using lookup tables on FPGAs

Santiago T. Pérez Suárez
Signals and Communications Department
University of Las Palmas de Gran Canaria (ULPGC)
Las Palmas de Gran Canaria, Spain
santiago.perez@ulpgc.es

***Abstract*—Sigmoid functions are used in artificial Neural Networks (NN). Normally, NNs are implemented in processors systems. In some scenarios these systems do not reach real time operation. In these cases, the NNs can be implemented in a Field Programmable Gate Array (FPGA). The sigmoid functions are not directly implementable in fixed point format, and some approximations must be used. A very used one is the lookup table technique. In this paper, an advanced design method based on Matlab and Simulink is presented. The Signal to Noise Ratio power is used to measure the approximation functionality. The automatic generation code to a hardware description language can be made.**

***Keywords—design methodology, digital circuit, FPGA, Matlab, Simulink, VHDL, Verilog, fixed point, floating point, sigmoid function, Neural Network, lookup table, Signal to Noise Ratio***

## I. INTRODUCTION

The sigmoid functions [1-8], particularly the hyperbolic tangent [9], are widely used as transfer functions in artificial Neural Networks (NN) [10]. The artificial NNs are usually implemented on a computer or microprocessor system in floating point arithmetic, for the training and testing phases. But sometimes these systems are not fast enough, in such cases the NNs implementations must be developed with others technologies for reach real time operation. It is also necessary to change the NN support if the level of power consumed or area are exceeded.

Floating point arithmetic can be used on digital devices, but these operations need a high number of clock cycles, great consumption of power and hardware resources [11]. However, floating point numerical representation has a wide range with a good resolution. It is possible to use fixed point versus floating point arithmetic, justified by its better performances. Fixed point increases the operating speed, the latency can be reduced to one clock cycle. The hardware resources and consumed power in fixed point are smaller than floating point. The drawback in fixed point is that the designer must care the range of the signals with the necessary resolution. This can be solved by signal studies with appropriate mathematical resources and strategies [12]. It should be emphasized that in fixed point the data format is totally arbitrary, which improves the performances. On the other hand, the use of floating point requires the election of a standard [13], increasing the number of bits and decreasing physical performances. Using a non-standardized floating point representation complicates the design process. For these reasons, this study is based on fixed point format, allowing to the sigmoid functions be included in NN for high performance operation.

Field Programmable Gate Arrays (FPGA) can be used for the implementation. The great advantage of FPGAs is that they are programmable by the designer, without having to be sent it to a factory. Others FPGAs advantages are that they have low cost of non-recurring engineering, minimum development time, ease debug and verification, shorter time to market, high data parallelism, flexible data format, and flexible pipelined [10]. For these reasons the modelling is presented for FPGA devices.

## II. SIGMOID FUNCTIONS

Transfer functions used in NNs are usually continuous and derivable, the derivability is a desirable requirement for NN training algorithms. Normally, two sigmoid functions are used, which satisfy these conditions. One of them is unipolar, simply called sigmoid, or *logsig* in Matlab notation. Its mathematical expression is given by equation 1, and its representation is in Fig. 1. The output of this function is restricted to the interval (0,+1). The second function is bipolar and coincides with the hyperbolic tangent, it is called *tansig* in Matlab notation. Its mathematical expression is given by equation 2, its representation is in Fig. 1. The output of this function is restricted to the interval (-1,+1). It can be easily demonstrated that equations 3 and 4 are satisfied, which shows that they are analogous functions.

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

$$y(x) = tansig(x) = \frac{e^{+x} - e^{-x}}{e^{+x} + e^{-x}} \tag{2}$$

$$tansig(x) = 2logsig(2x) - 1 \tag{3}$$

$$logsig(x) = \frac{1}{2}\left[tansig\left(\frac{x}{2}\right) + 1\right] \tag{4}$$

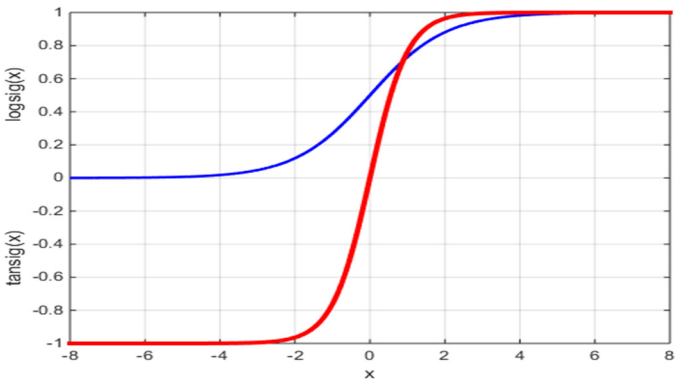


Fig. 1. The unipolar and bipolar sigmoid functions.

## III. State of the Art of Hardware Implementation for Sigmoidal Functions

It should be emphasized that both functions are non-linear, because they include division and exponential operations, so they are the bottleneck of the NN design. That is, the implementation of sigmoid functions is not immediate in fixed-point arithmetic and they are usually approximated with some method. One approach is using Look-up Tables (LUTs), which store samples of the function; this method needs a lot of hardware resources but gets a high speed [4]. Other approximation form is the Piece Wise Lineal (PWL), which approaches each section with a straight line [14]. It is also possible to approximate each section with polynomials, usually of grade two [14]. Other authors use piecewise Taylor approximations [15]. Other approaches propose specific shape functions that consider the characteristics of the sigmoid functions; firstly where its derivative tends to a constant, and secondly its symmetry characteristics [2, 4]. Other authors use functions that look like sigmoid functions [16], in such cases the error is bounded.

The comparison of these solutions is based on the functionality, which is measured with the error. There are several types of errors, and sometimes implementations are compared with different error values and error types [1, 17, 18]. It should be noted that the effect of the number of bits in the representation is chosen discreetly or arbitrarily by authors [4, 14], usually based on the experience or previous works.

Usually, authors extract area and speed, but almost never the consumed power [4, 14]. Many authors study the speed as the latency of the system, number of clock cycles required, but without estimating the maximum frequency [14].

## IV. Objectives

The objective is to fix a methodology for sigmoidal functions designed on digital programmable devices, in particular, the hyperbolic tangent. The approximation type will be based on LUTs implemented with logical elements.

There are many design methods, this development focuses on Simulink [19] of Matlab [20] using fixed point arithmetic. This design flow is fast and flexible, allowing to check different architectures and the effect of the binary format; this makes possible to scan the number of bits in a systematic and extensive form.

Once the systems have been chosen, which reach the functionality with the smaller sizes of fixed-point representation, from Simulink can be generated the project in a standard Hardware Description Language (HDL). One of them is Very High Speed Integrated Circuit Hardware Description Language (VHDL), and the other is Verilog. The generated project is formed by the digital implementation and files with input and output signals, to perform the necessary simulations.

It is also possible to design in Simulink for the two main FPGA providers, which are Altera [21] and Xilinx [22]; with their own tools, which are respectively DSP Builder [23] and System Generator [24]. In the design tools of these manufacturers it is possible to extract the performances for the chosen device [25, 26].

To show this design method the manufacturer Altera has been chosen, and the project has been generated in Verilog language. In this work it is assumed that NN training is performed outside the device, this is called offline type [27], so it is not considered the approximation of the first derivative of the function. For example, for offline training can be used the Neural Network Toolbox [28] from Matlab.

## V. The Measure of Functionality

There are four parameters that can be evaluated in digital implementations: the functionality, the area, the power and the system speed. The intention is to analyse approximations with equal or similar functionality; afterwards, the three remaining parameters are contrasted. The first question is how to measure functionality, clearly associated with the error, but with different versions.

Let the function $y=f(x)$ be the one to be approximated, and let $ya=fa(x)$ be the expression of the approximation; the error is defined as $E(x)=ya-y=fa(x)-f(x)$, which generally has null mean value. The absolute error is defined as $E_{abs}(x)=|E(x)|$.

For comparing designs, the maximum value of the absolute error $|E(x)|_{max}$ can be used, as in [14]. Although, this is an important measure, a good approximation can have a high value error only in a small interval; anyway, it is important to take it in account.

Other measure is the mean value of the absolute error [14]; which measures an approximation on an interval, but high error values can be masked. Therefore, both measures can be combined, as in [7]. Occasionally, the square root of the mean value of the square error has been used.

The relative error, defined as $E_{rel}(x)=(ya-y)/y=(fa(x)-f(x))/f(x)$ can also be used, no reference of it has been found. The relative error can be used for comparing same type functions, or different type, for the same input range. The relative error grows enormously when the value to be approximated tends to zero; moreover, it is not defined if the function value is zero and the approximation value is not zero. The absolute peak value, or average absolute value, of relative error can be considered, with the same previous observations. The relative error allows compare different function types, as it is shown in Fig. 1, because the numerator introduces the error and the denominator introduces the value of the function; besides, it can be expressed in per centum values.

To avoid uncertainty when the signal is zero, the error can be measured against the peak-to-peak function value (equation 5). This value could be used to compare approximations of different functions.

$$E_{pp}(x) = \frac{E(x)}{y_{max} - y_{min}} \quad (5)$$

Different authors use different error types, which makes comparison difficult. A relative measure is proposed, which is the quotient between function and error, which can also be called signal and noise, respectively. The sampled signals, input *x* and output *y*, before quantizing and coding, are discrete time analog values; and also the generated noise signal. The Signal to Noise Ratio (SNR) [29] is defined in expression 6.

$$SNR = \frac{\sum_{i=0}^{k} y[i]^2}{\sum_{i=0}^{k} E[i]^2} \quad (6)$$

The numerator is the energy of *(k+1)* samples of the signal to be approximated, the denominator is the energy of *(k+1)* samples of the corresponding error. Therefore, the equation 6 is the ratio of the energies of signal and error. The equation 6 coincides with the relation between the signal power and the noise power. This concept is common in analog and digital communications. Often, this relation is expressed in decibels (equation 7) [29].

$$SNR(dB) = 10log_{10}(SNR) \quad (7)$$

## VI. The Model and its Parameters

The Simulink block diagram for the approximation is shown in Fig. 2, where the arithmetic used is two's complement. The *1-D Lookup Table* block is the element that stores the samples for the approximation. In Fig. 2 the LUT stores 16 words, so its address bus has 4 bits; Simulink type is denoted as *ufix4*, 4 bits unsigned fixed-point. The words in the LUT have 1 sign bit, 1 bit for the integer part and 7 fractional bits, (*sfix9_En7*) type; so in the output the values ±1 are representable, and the function is saturable. Besides, the *1-D Lookup Table* block does not allow using a sign bit without any bit for the integer part. The input format has 1 sign bit, 2 bits for the integer part and 6 fractional bits, (*sfix9_En6*) type. The represented input range is [-4,+4-$2^{-6}$]. The multiplier and adder in Fig. 2 perform a conversion of the input *x* to the LUT address bus. In this case, the conversion is from [-3.75,+3.75] to [0,+15]. The constants *G* and *C* can be represented without error under certain conditions, this will be explained below.

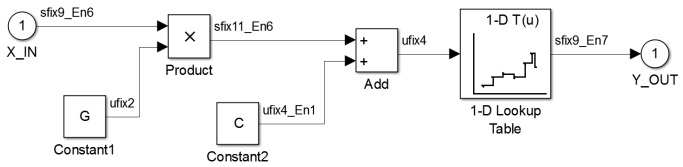


Fig. 2. The Simulink approximation model for 16 words LUT.

Fig. 3 shows the hyperbolic tangent function for [-4,+4] input range; also, the approximation and the error are shown. It should be noted that the LUT samples are evenly spaced in the input range. The SNR measured is 23.36 dB, according to expression 7.

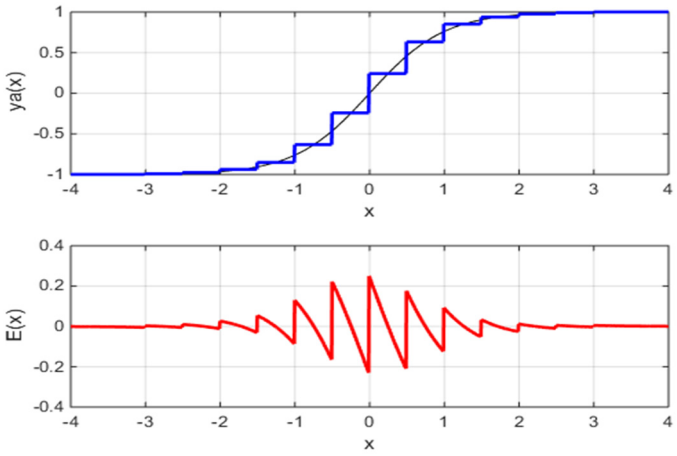


Fig. 3. The hyperbolic tangent function, 16 words LUT output and error.

Once the model has been established, a set of parameters can be discussed: the format of the input signal, the number of words stored in the LUT, and the format of the words stored in the LUT. The number of words stored in the LUT is a power of 2; so this takes advantage of the address bus. Then, *M* words will be addressed by *n* address bits, such that $M=2^n$. The output format has 1 bit sign and 1 bit for integer part, as was explained previously. But the number of bits of the fractional part (*nbfo*) can be modified.

In the input the size of fractional bits (*nbfi*) can be changed, which affects the resolution. The question is how many bits to use for the input integer part (*nbii*). When the number of bits of the input fractional part is large the input range tends to [-$2^{nbii}$,+$2^{nbii}$); in fact, the range will be [-$2^{nbii}$,+$2^{nbii}$-$2^{nbfi}$]. Then, with *nbii* equal to 1 the representation range is [-2,+2-$2^{nbfi}$], with *nbii* equal to 2 the range is [-4,+4-$2^{nbfi}$], with *nbii* equal to 3 the range is [-8,+8-$2^{nbfi}$], etc. It is proposed to represent the function in an interval centred in the origin, and saturate the output to ±1 values outside this interval, since the function has two horizontal asymptotes. For this purpose, the SNR is measured when a sawtooth signal is introduced in the range [-16,+16]. Fig. 4 shows the saturation error outside the range [-4,+ 4], which is the input range with 2 bits for the integer part when the number of fractional bits grows indefinitely. That is, the saturation approximation is given by expression 8. The SNR obtained is 81.25 dB, which is a high value. This is the maximum SNR for the input range [-16,+16] when the hyperbolic tangent is approximated in the interval [-4,+4].

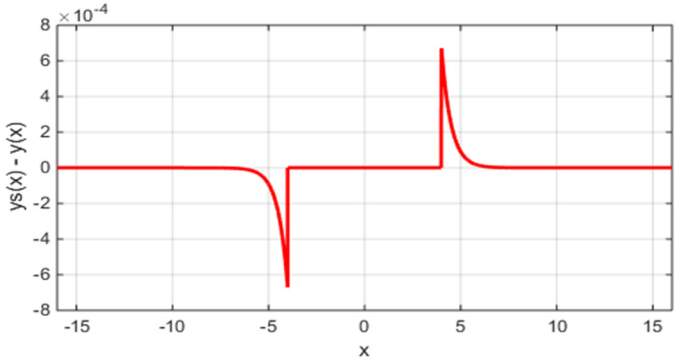


Fig. 4. The saturation error outside the range [-4,+ 4].

$$ys(x) = \begin{cases} -1 & x < -4 \\ tansig(x) & -4 \le x \le +4 \\ +1 & +4 < x \end{cases} \quad (8)$$

With the saturation out of the range [-2,+2], with 1 integer bit, the SNR for input range [-16,+16] is 46.61 dB; a high value but easily surmountable. The saturation out of the range [-8,+8], for 3 integer bits, the SNR for input range [-16,+16] is 150.73 dB, extremely high and unnecessary value. Therefore, the number of integer bits in the input is set to 2, the approximation will be performed within the range [-4,+4], and the circuit will saturate outside this range (equation 9). For measure the SNR of the approximation, the input signal will be a sawtooth signal in the range [-4,+4]. This coincides with the assumption that all values in the input for that range are equally likely.

$$ysa(x) = \begin{cases} -1 & x < -4 \\ ya(x) & -4 \le x \le +4 \\ +1 & +4 < x \end{cases} \quad (9)$$

Finally, the parameters to be varied will be: the number of fractional bits of the input signal (*nbfi*), the number of bits of the LUT address bus (*n*), and the number of fractional bits for stored words in the LUT (*nbfo*). It is convenient to revise the final format of the input signal; 1 sign bit, 2 bits for the integer part and *nbfi* fractional bits; *sfix(1+2+nbfi)_En(nbfi)* type. Similarly, the output signal has 1 sign bit, 1 integer bit and *nbfo* fractional bits; this is a *sfix(1+1+nbfo)_En(nbfo)* type.

The conversion of the input signal to the LUT address is the expression 10; obviously, *A* (address) is an integer between 0 and $(2^n-1)$.

$$A = nearest\ enteger\ (Gx + C) \quad (10)$$

The constants values *G* and *C* are given by expressions 11 and 12; where $x_{min}$ is -4, $x_{max}$ is +4 and *M* is $2^n$, and *n* is the number of bits for LUT address. Since the extremes of *x* and *M* are entire powers of 2, these constants can be expressed as powers of 2 and have an exact representation in fixed-point format.

$$G = \frac{M}{x_{max} - x_{min}} = 2^{(n-3)} \quad (11)$$

$$C = \frac{M-1}{2} = 2^{(n-1)} - 2^{(-1)} \quad (12)$$

## VII. Scan of Model Parameters

For *n* equal to 4 (*M=16*) the number of fractional bits in input and output (*nbfi*, *nbfo*) was varied between 0 and 24. The SNR was measured in dBs. The maximum SNR obtained was 23.36 dB, and the system gets this value with 6 input fractional bits and 7 output fractional bits. It should be emphasized that for 16 words LUT, 625 configurations were simulated. The value of fixed point output signal is stored in Matlab variables space in floating point format; afterwards, the SNR is measured in dBs. Fig. 5 shows the shape of SNR in dBs for 16 words LUT against input and output fractional bits. In other words, in order to reach the maximum SNR with the 16 words LUT, at least 6 input fractional bits and 7 output fractional bits are required, which is marked in Fig. 5. Increasing fractional bit numbers above these values do not increase the SNR.

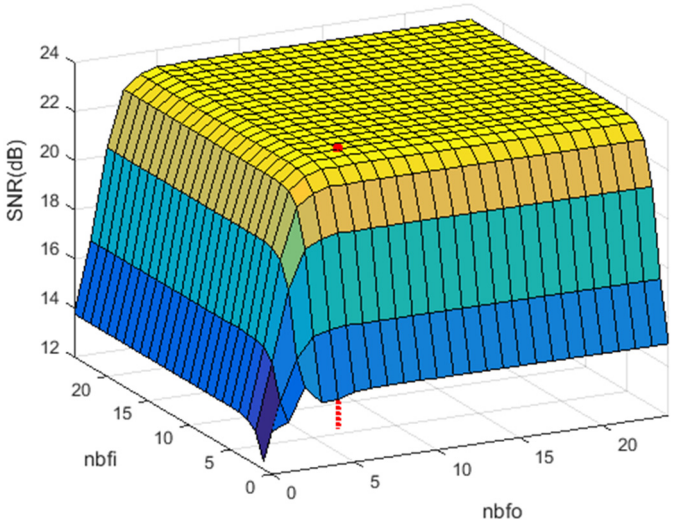


Fig. 5. The shape of SNR in dBs for 16 words LUT.

For the implementation the values 6 and 7 are chosen for *nbfi* and *nbfo* because the maximum SNR is obtained with the minimum number of bits, this system is shown in Fig. 2. From Simulink the Verilog project was generated for the Altera device EP2AGX260FF35I5 of Arria II GX family. The project was compiled with Quartus II [30] and simulated with ModelSim-Altera Edition [31] and the Simulation Waveform Editor included in Quartus II. The schematic circuit is in Fig. 6; which shows the input and output of the function, the clock, the reset, and input and output enable signals. In short, only one FPGA implementation was generated, the most convenient case of the 625 Simulink simulations.

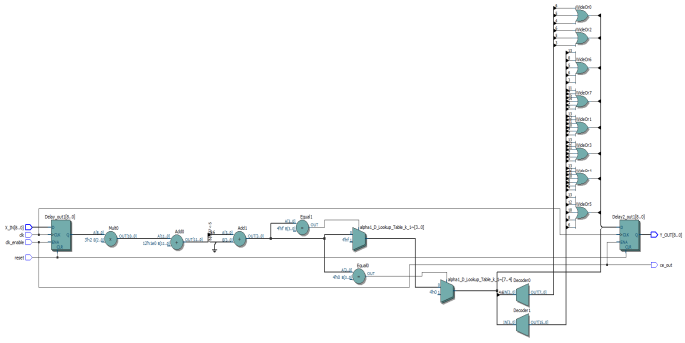

Fig. 6. The schematic circuit of 16 words LUT.

The ModelSim simulation is shown in Fig. 7 for 25 MHz clock frequency. Only the input and output signals are shown in this figure; avoiding auxiliaries signals for simplification. The simulation input rate is $25 \cdot 10^{+6}$ values per second. The input and output registers, shown in Fig. 6, set the latency of the system in 2 clock cycles. These registers, which are not shown in Fig. 2 for simplification, are necessary in Simulink for generating the clock signal in the project. The output Y_OUT of Fig. 7 is equal to the output signal in Simulink. This can be ensured because when the project is generated from Simulink, the input and output test signals (*testbench*) are also generated. The fixed point

input signal is used in circuit simulation with ModelSim. On the other hand, the output signal of this simulator is compared with the Simulink fixed point output signal, at the end the message "test completed passed" indicates that output signals have the same values.

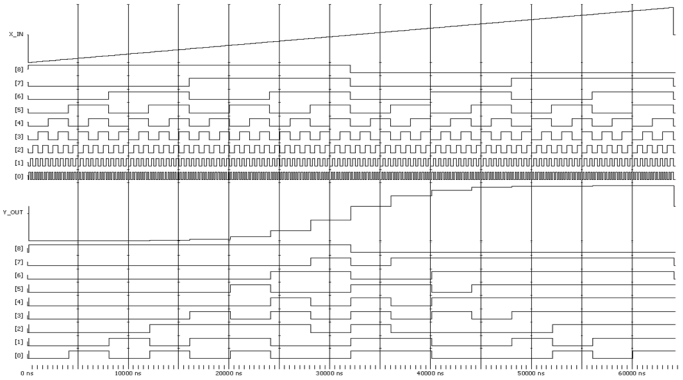

Fig. 7. The circuit simulation with ModelSim of 16 words LUT.

In Quartus II hardware resources were estimated, for the previous device the area was evaluated as the number of Combinational Adaptive Lookup Tables (ALUTs); for this implementation were needed 8 ALUTs. With the TimeQuest Timing Analyser of Quartus II it was found that the maximum operating frequency of the system was 909.09 MHz; for the worst case, which among other things, takes into account the temperature at which the device operates. Finally, with the PowerPlay Power Analyser of the Quartus II, for a clock frequency of 25 MHz, a dynamic power of 0.80 mW and static power of 858.86 mW was obtained, that gives a total of 859.66 mW.

The power estimation was done with a fixed clock frequency, in order to compare the consumed power of different designs for the same data rate. This clock frequency must be less than the smallest of maximum frequencies of the compared designs. On the other hand, all power estimations were made for an ambient temperature of 25°C, without heatsink and no forced air flow. The power estimation was performed after loading the Value Change Dump File, which sets the form of change of the input signals. This is an input file for the power analyser, which stores the change rates and static probabilities of signals. The Value Change Dump File was generated with the Simulation Waveform Editor, where the clock frequency was set to 25 MHz and the binary input signals were randomly varied.

## VIII. Design Flow

Fig. 8 shows the design flow process. Between the floating point input signal and the fixed point model, a data type converter block exists and is not shown for simplification in Fig. 2. Similarly, a data type converter block exists at the output of the Simulink model. These converter blocks are not implemented in hardware, the input and output of the approximated function are in fixed point format. The Simulink model can be loaded, from Matlab, with the number of LUT address bits ($n$), the number of input fractional bits ($nbfi$) and the number of output fractional bits ($nbfo$).

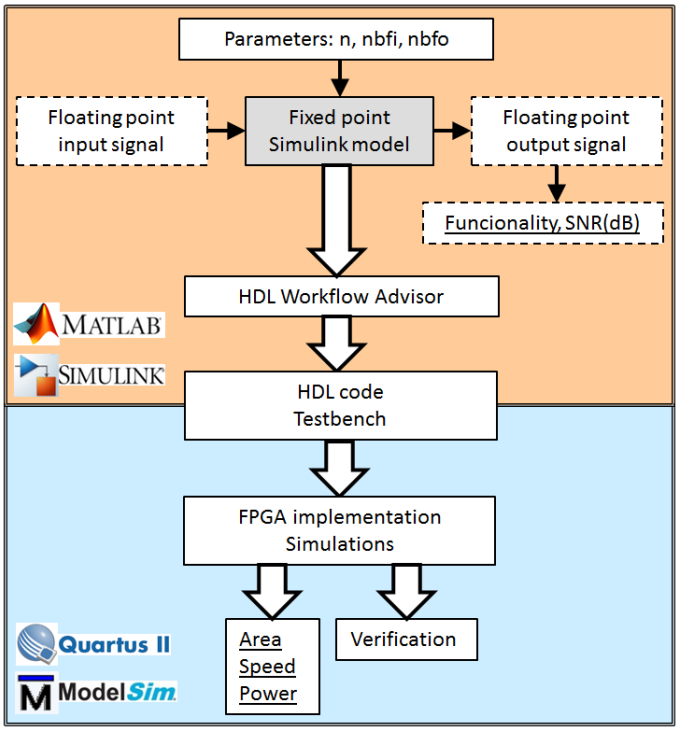


Fig. 8. The design flow process from floating point to FPGA implementation.

The SNR is calculated with the exact floating point value of the hyperbolic tangent and the floating point values of the fixed point output signal. For the model with the desired functionality, and using HDL Workflow Advisor [32], it is generated the HDL project and the testbench. Not all Simulink blocks are supported by the HDL Workflow Advisor. In particular, only blocks of the HDL Coder library are implementable. But not all configurations of these blocks permit implementation, although they are simulable. Generating these files manually can be an impossible task, but this automatic generation can be performed in minutes. It should be noted that in HDL Workflow Advisor a synthesis tool is chosen, which has been installed previously in the computer; in this case Quartus II of Altera. Besides, the type of FPGA is chosen, and for the testbench allows set the clock, reset and enable signals. Once the HDL Workflow Advisor has completed the files generation, it is possible to implement the FPGA and obtain the hardware resources, the maximum frequency and the consumed power. With the Simulation Waveform Editor can be performed functional and timing simulations; also, testbench simulations can be performed with ModelSim. These last simulations take the input signal that was used in Simulink. On the other hand, ModelSim simulation compares its output with the Simulink output signal, which verifies the circuit operation. The implementation and simulations must not be dissociated, some simulations are necessary to set parameters for the power estimator.

Different designs are characterized by performances, underlined in Fig. 8: the functionality, measured as the SNR in dB; the area, hardware resources occupied in the FPGA; the speed, measured as the maximum frequency operation; and the consumed power for a certain operating frequency.

## IX. Experiments and Results

In the previous section has been set: the model, the parameters, the design methodology and the measurement of performances. This section presents the results when the design parameters are varied. For this purpose, the number of bits in LUT address bus (*n*) was varied until 16. For each *n* value, the number of input and output fractional bits, *nbfi* and *nbfo* respectively, were varied from 0 to 24. A similar study to the previous section was done, the results of SNR were stored in matrices of 25 by 25 elements, similar behaviours were obtained to Fig. 5. For each *n* value, only the case of maximum SNR with the minimum numbers of bits was implemented, and its performances were evaluated. Fig. 9 shows the 16 responses obtained. Obviously, if the number of words in the LUT increases, the maximum SNR grows, but it is necessary to increase the number of input and output fractional bits. In Simulink the number of simulations for generating Fig. 10 was 10,000 (16x25x25), which were performed by running a loop for each *n* value.

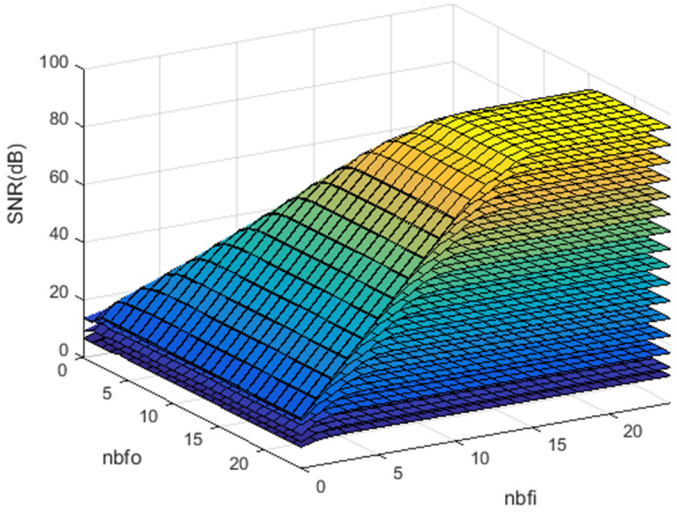


Fig. 9. The SNR in dBs versus input and output fractional bits, for each number of bits in LUT address bus.

### A. Measurements and results with no device dependency

Fig. 10 shows the maximum SNR for each n value. The SNR is almost linear versus n, and increases 6 dB per bit. The SNR in equation 6 is multiplied by 4 when the number of words is doubled in the LUT. In Fig. 11 it is shown the input fractional bits versus the address bus size for the maximum SNR. In general, the number of input fractional bits is equal to the number of address bits plus four. This tendency indicates that it is necessary to increase one bit in the input when a bit is increased in the address bus; that is, the number of words in the LUT is doubled. Fig. 12 shows the number of output fractional bits versus the size of the address bus, necessary to reach the maximum SNR, an almost linear tendency exists. In general, the number of bits required in the output is equal to the number of address bits plus four. It is necessary to increase one bit in the output when a bit is increased in the address bus; that is, the number of words is doubled in the LUT.

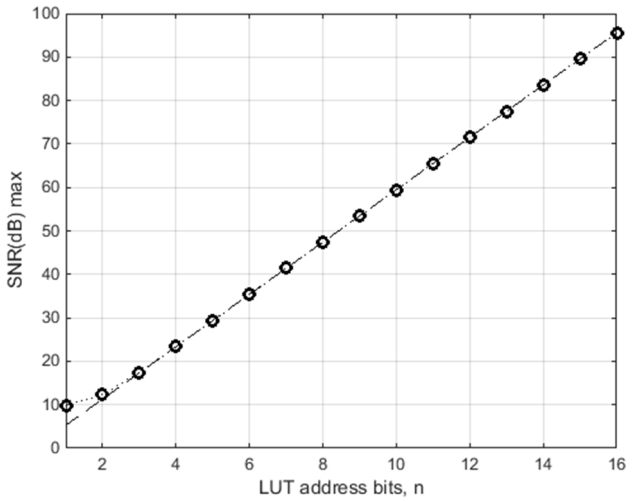


Fig. 10. The maximum SNR in dBs for each n value .

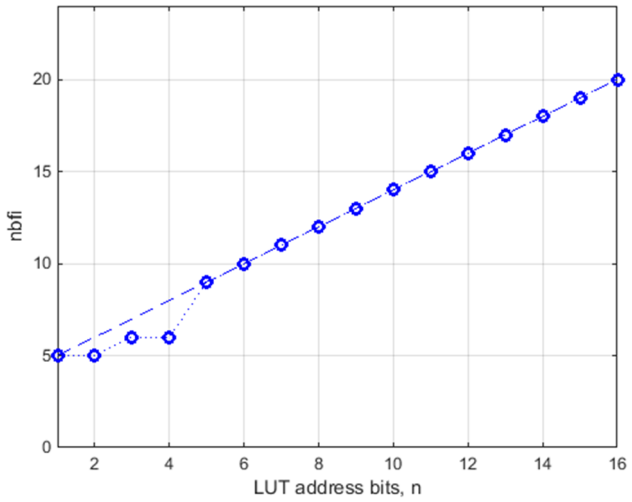


Fig. 11. The input fractional bits against the address bus size for the maximum SNR.

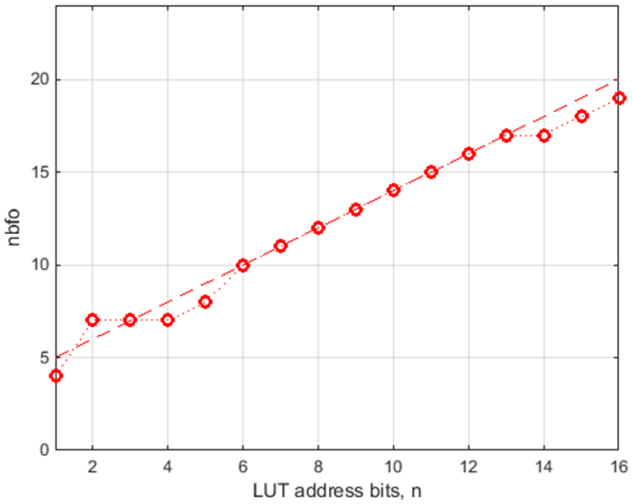


Fig. 12. The output fractional bits against the address bus size for the maximum SNR .

It should be emphasized that for 16 bits in the address bus the Verilog design size is 65,723 lines, and the testbench is 13,107,541 lines, including blank lines and comments; a hand coded design using a HDL would have been impossible to generate.

*B. Measurements and results with device dependency. The Altera Arria II GX family*

For this family, the device EP2AGX260FF35I5 was chosen. One implementation was developed for each size of the LUT address bus; it was for the maximum SNR and minimum numbers of input and output fractional bits. The results of area, speed and power performances are shown below. It should be noted that the HDL Workflow Advisor does not allow to generate the project for one bit in the address bus; as this is a trivial case, the study from 2 to 16 bits in this bus is presented. The area versus the number of words in the LUT is shown in Fig. 13, which has a nearly linear shape. The area blocks of this device are Combinational ALUTs (Adaptive Lookup-Tables). The area depends heavily on the number of words but weakly on the number of bits, for the maximum SNR. On average, 0.2 Combinational ALUTs per word stored in the LUT is needed.

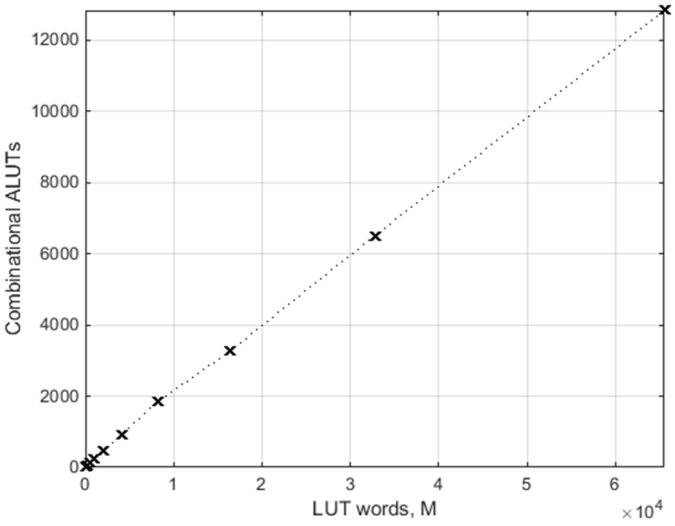


Fig. 13. Hardware resources against the number of words in the LUT (Altera device EP2AGX260FF35I5 of Arria II GX family).

The maximum frequency versus $n$, (and versus $M$) for the maximum SNR, are not linear. The minimum allowed period for each case is represented in Fig. 14, it has almost linear behaviour by zones. In any case, a clear tendency is not observed for the speed.

Finally, the power was estimated for the previous cases (Fig. 15), the clock frequency was 25 MHz, smaller than the minimum. The represented power is the sum of dynamic and the static powers, the behaviour is almost linear against the number of words (0.0026 mW/word).

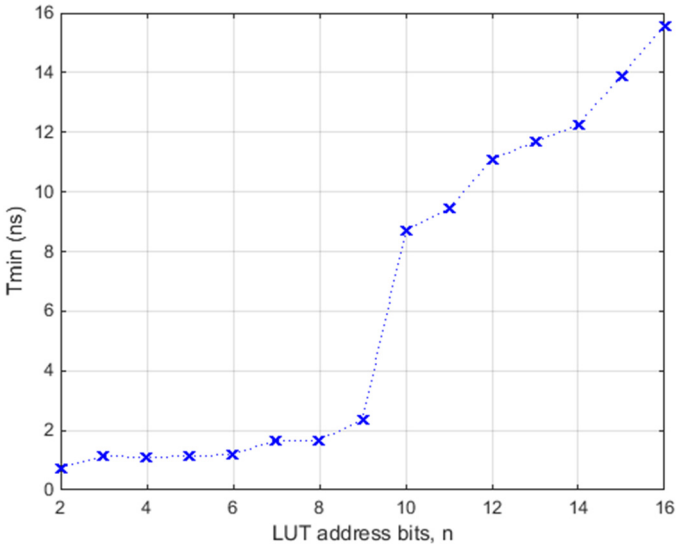


Fig. 14. The minimum period against the size of the LUT address bus for the maximum SNR (Altera device EP2AGX260FF35I5 of Arria II GX family).

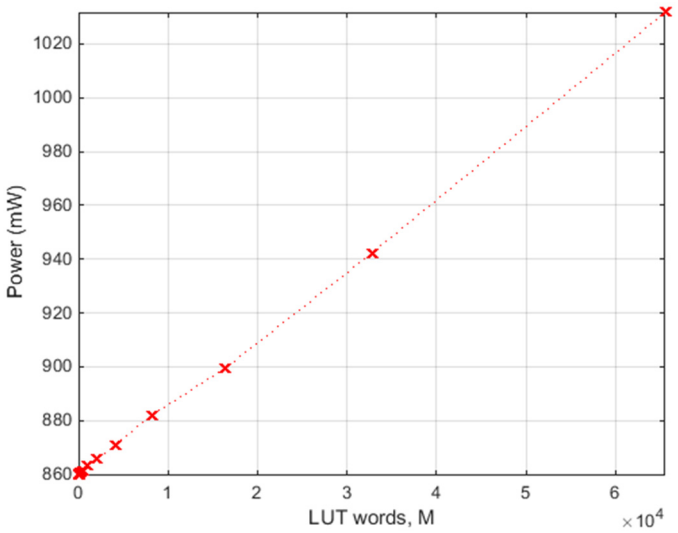


Fig. 15. The consumed power against the number of words in the LUT for 25 MHz clock frequency (Altera device EP2AGX260FF35I5 of Arria II GX family).

It is possible to define a Quality Factor (expression 13), which includes the physical performances and functionality; the SNR obeys expression 6, not in dBs. In expression 13 the maximum frequency is expressed in hertz, the area is the number of Combinational ALUTs and the power is introduced in watts. The Quality Factor has an almost linear behaviour for $n$ greater than 10, generally this value increases with $M$, according to Fig. 16. This factor can be used to compare different designs.

$$QF = \frac{f_{max} \cdot SNR}{Area \cdot Power} \tag{13}$$

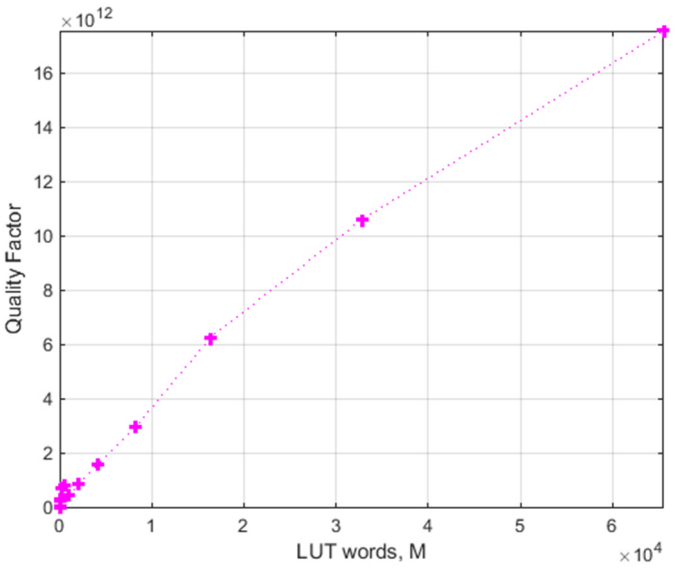


Fig. 16. The Quality Factor versus the number of words in the LUT (Altera device EP2AGX260FF35I5 of Arria II GX family).

### C. *Measurements and results with device dependency. The Altera Cyclone IV GX, MAX 10 and Stratix V families*

Analogous results were obtained for other devices of different Altera FPGA families:

- the EP4CGX150DF31I7AD device of the Cyclone IV GX family,

- the 10M50SFE144I7G device of the MAX 10 family,

- the device 5SGXMBBR3H43I4 of the family Stratix V.

In the four devices, included the Arria II GX family, the area is strongly linear versus the number of words in the LUT; for the speed is not observed a clear rule, vaguely the minimum period allowed in the clock signal is almost linear piecewise versus the number of bits of the LUT address bus; finally, the consumed power is strongly linear versus the number of words in the LUT. In all cases the defined Quality Factor grows with the number of words; sometimes it is almost linear.

It should be noted that the logical blocks of the FPGAs families are:

- Combinational ALUTs (Adaptive Lookup-Tables), for Arria II GX family;

- Logic Elements, similar but not equal, for Cyclone IV GX and MAX 10 families,

- and Adaptive Logic Modules for the Stratix V family.

For the largest circuit, with 16 bits in the LUT address bus and maximum SNR; the area for the devices is shown in Table I. The Logic Elements of the Cyclone IV GX and MAX 10 families have similar behaviour for this design. Analogous, the Combinational ALUTs of the Arria II GX family and the Adaptive Logic Modules of the Stratix V are similar for implementing this design.

TABLE I. THE HARDWARE RESOURCES FOR 16 BITS IN THE LUT ADDRESS BUS

| FAMILY | DEVICE | ELEMENT | Area for 16 bits in LUT address bus |
|---|---|---|---|
| Arria II GX | EP2AGX260FF35I5 | Combinational Adaptive Lookup-Tables | 12,828 |
| Cyclone IV GX | EP4CGX150DF31I7AD | Logic Elements | 45,108 |
| MAX 10 | 10M50SFE144I7G | Logic Elements | 45,172 |
| Stratix V | 5SGXMBBR3H43I4 | Adaptive Logic Modules | 12,009 |

## X. CONCLUSIONS

One of the objectives of this contribution is to show a fast and flexible design method for digital devices, which allows verifying different architectures and data formats for a system. In other words, the method allows exploring the space solutions. These advanced design techniques are embedded in Matlab for floating point models in files with Matlab extension [33] or for Simulink systems [34] and are connected to a digital synthesis tool. This method allows studying the effect of the number of bits in a wide range, which many authors only study in a discreet form by the limitation of the used method. Optimized systems can be transferred to a FPGA.

Altera and Xilinx have their own environments for designing as a block diagram in Simulink; these are DSP Builder [23] and System Generator [24] respectively, but handling fixed point format is more difficult.

The SNR has been introduced for measuring the functionality, this allows to compare the quality of an approximation of the sigmoid function (expression 1) to an approximation of the hyperbolic tangent (expression 2), for the same input range. On the other hand, the functionality estimation with the SNR in dBs allows observing linearities in performances.

## ACKNOWLEDGMENT

Acknowledgment is expressed to Altera for donating software and licenses within its University Program.

## REFERENCES

[1] A. Armato, L. Fanucci, G. Pioggia and D. D. Rossi, "Low-error approximation of artificial neuron sigmoid function and its derivative," *Electronics Letters,* vol. 45, no. 21, pp. 1082- 1084, 2009.

[2] K. Basterretxea, J. M. Tarela y I. del Campo, «Approximation of sigmoid function and the derivative for hardware implementation of neural networks,» *IEE Proceedings-Circuits Devices and Systems, vol. 151, nº 1,* pp. 18-24, 2004.

[3] I. d. Campo, R. Finker, J. Echanobe y K. Basterretxea, «Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons,» *Electronics Letters,* vol. 49, nº 25, p. 1598–1600, 2013.

[4] P. K. Meher, «An Optimized Lookup-Table for the Evaluation of Sigmoid Function for Artificial Neural Networks,» de *18th IEEE/IFIP International Conference on VLSI and System-on-Chip (VLSI-SoC 2010)*, 2010.

[5] M. C. Miglionico y F. Parillo, «A novel approach for implementing of a log-sigmoid function on a FPGA device using Sfloat24 Math library - An modelling,» de *Proceedings of the International Symposium on the Analytic Hierarchy Process*, 2013.

[6] S. Ngah, R. A. Bakar, A. Embong y S. Razali, «Two-steps implementation of sigmoid function for artificial neural network in Field Programmable Gate Array,» *ARPN Journal of Engineering and Applied Sciences,* vol. 11, nº 7, pp. 4882-4888, 2016.

[7] M. T. Tommiska, «Efficient digital implementation of the sigmoid function for reprogrammable logic,» *IEE Proceedings-Computers and Digital Techniques,* vol. 150, nº 6, pp. 403-411, 2003.

[8] C.-H. Tsai, Y.-T. Chih, W. H. Wong y C.-Y. Lee, «A Hardware-Efficient Sigmoid Function With Adjustable Precision for a Neural Network System,» *IEEE Transactions on Circuits and Systems - II: Express Briefs,* vol. 62, nº 11, pp. 1073-1077, 2015.

[9] M. Panicker y C.Babu, «Efficient FPGA Implementation of Sigmoid and Bipolar Sigmoid Activation Functions for Multilayer Perceptrons,» *IOSR Journal of Engineering (IOSRJEN),* vol. 2, nº 6, pp. 1352-1356, 2012.

[10] A. R. Omondi y J. C. Rajapakse, FPGA Implementations of Neural Networks, Springer, 2006.

[11] M. A. Cavuslu, C. Karakuzu, S. Sahin y M. Yakut, «Neural network training based on FPGA with floating point number format and it's performance,» *Neural Computing and Applications,* vol. 20, nº 2, pp. 195-202, 2011.

[12] MathWorks, «Fixed-Point Designer,» [En línea]. Available: http://www.mathworks.es/products/fixed-point-designer. [Último acceso: june 2017].

[13] IEEE, «IEEE Standard for Floating-Point Arithmetic,» [En línea]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=4610933, permanent link. [Último acceso: june 2017].

[14] S. Vassiliadis, M. Zhang y J. G. Delgado, «Elementary Function Generators for Neural-Network Emulators,» *IEEE Transactions on Neural Networks,* vol. 11, nº 6, pp. 1438-1449, 2000.

[15] B. Lee y N. Burgess, «Some results on Taylor-series function approximation on FPGA,» de *37th Asilomar Conference on Signals, Systems and Computers*, 2003.

[16] E. Soria-Olivas, J. D. Martín-Guerrero, G. Camps-Valls, A. J. Serrano-López, J. Calpe-Maravilla y L. Gómez-Chova, «A Low-Complexity Fuzzy Activation Function for Artificial Neural Networks,» *IEEE Transactions on Neural Networks,* vol. 14, nº 6, pp. 1576-1579, 2003.

[17] A. Armato, L. Fanucci, E. P. Scilingo y D. De Rossi, «Low-error digital hardware implementation of artificial neuron activation functions and their derivative,» *Microprocessors and Microsystems,* vol. 35, nº 6, pp. 557-567, 2011.

[18] S. Roy y P. Banerjee, «An algorithm for trading off quantization error with hardware resources for MATLAB-based FPGA design,» *IEEE Transactions on Computers,* vol. 54, nº 7, pp. 886-869, 2005.

[19] MathWorks, «Simulink,» [En línea]. Available: http://www.mathworks.com/products/simulink. [Último acceso: june 2014].

[20] MathWorks, «MATrix LABoratory de MathWorks (Matlab),» [En línea]. Available: http://www.mathworks.com. [Último acceso: june 2014].

[21] Altera, «Altera Corporation,» [En línea]. Available: http://www.altera.com/. [Último acceso: june 2014].

[22] Xilinx, «Xilinx Corporation,» [En línea]. Available: http://www.xilinx.com. [Último acceso: june 2014].

[23] Altera, «DSP Builder,» [En línea]. Available: http://www.altera.com/products/software/products/dsp/dsp-builder.html. [Último acceso: june 2014].

[24] Xilinx, «System Generator for DSP,» [En línea]. Available: http://www.x.com/products/design-tools/vivado/integration/sysgen.html. [Último acceso: june 2014].

[25] Altera, «Quartus II,» [En línea]. Available: https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html. [Último acceso: june 2014].

[26] Xilinx, «Vivado Design Suite,» [En línea]. Available: https://www.xilinx.com/products/design-tools/vivado.html. [Último acceso: june 2017].

[27] S. T. Pérez, C. Osorio, J. L. Vásquez, J. B. Alonso y C. M. Travieso, «Design methodology of a fully parallelized Neural Network on a FPGA,» de *8th WSEAS International Conference on Circuits, Systems, Signal and Telecommunications (CSST '14)*, 2014.

[28] MathWorks, «Matlab Neural Network Toolbox,» [En línea]. Available: http://es.mathworks.com/products/neural-network. [Último acceso: june 2014].

[29] A. V. Oppenheim y R. W. Schafer, Discrete-time signal processing, Prentice-Hall, 1989.

[30] Altera, «Intel Quartus Prime Design Software Overview,» [En línea]. Available: https://www.altera.com/products/design-software/fpga-design/quartus-prime/overview.html. [Último acceso: june 2017].

[31] Altera, «ModelSim-Altera Edition,» [En línea]. Available: https://www.altera.com/products/design-software/model---simulation/modelsim-altera-software.html. [Último acceso: june 2017].

[32] MathWorks, «HDL Workflow Advisor,» [En línea]. Available: https://www.mathworks.com/examples/matlab-hdl-coder/mw/hdlcoder_product-mlhdlc_tutorial_hdlcodegen-basic-hdl-code-generation-with-the-workflow-advisor. [Último acceso: june 2017].

[33] MathWorks, «Fixed-Point Advisor,» [En línea]. Available: https://es.mathworks.com/help/fixedpoint/ug/fixed-point-advisor.html. [Último acceso: june 2017].

[34] MathWorks, «HDL Workflow Advisor,» [En línea]. Available: https://www.mathworks.com/examples/matlab-hdl-coder/mw/hdlcoder_product-mlhdlc_tutorial_hdlcodegen-basic-hdl-code-generation-with-the-workflow-advisor. [Último acceso: june 2017].