Controlled-Accuracy Approximation of Nonlinear Functions for Soft Computing Applications

A high performance co-proccessor for intelligent embedded systems

Inés del Campo, Javier Echanobe, Estibaliz Asua, and Raúl Finker
Department of Electricity and Electronics
University of the Basque Country UPV/EHU
Leioa, Vizcaya, Spain

***Abstract*— Intelligent embedded systems can be found everywhere in a variety of innovative applications. The main challenge consists in developing small-size single-chip embedded systems with low power consumption, capable of processing data and intelligent algorithms with the required speed. These key issues are normally carefully analyzed during the design process of embedded systems with the aim of meeting the required specifications. However, the problem of accuracy is hardly ever explored in the early stages of the design flow, even though too low accuracy could limit digital hardware performance in a crucial way. This piece of work proposes a controlled accuracy approximation scheme of nonlinear functions based on Taylor's Theorem and the Lagrange form of the remainder. A hardware co-processor based on a Field Programmable Gate Array (FPGA) is developed. The co-processor is suitable for efficient computation of nonlinear functions involved in typical soft computing techniques such as: activation functions (neural netwoks); membership functions (fuzzy systems); or kernel functions (support vector machines). The method is applied to the development of an intelligent embedded system for a smart scenario. Experimental results are provided for both online training and feed-forward computation of a single-layer feed-forward neural network.**

I. INTRODUCTION

Many embedded platforms have emerged in the market and are in use in our daily activities. With the aim of promoting an autonomous and flexible lifestyle, with new levels of comfort, safety and productivity, embedded systems are being introduced in almost all areas. They can be found everywhere in a variety of applications, from control systems in automotive sectors, to consumer and multimedia products, among others. An embedded system is a special-purpose computing platform able to perform dedicated functions. It is often designed for a particular kind of application that is required to work under a certain constraints. An increasing number of these applications require certain degree of knowledge and intelligence to perform properly. Soft computing techniques, such as artificial neural networks (ANNs) or neuro-fuzzy systems (NFSs), provide a rich and powerful framework in order to endow embedded system with intelligence. The main challenge consists in developing small-size single-chip intelligent embedded systems, with low power consumption, capable of processing data and algorithms with the required speed and without loss of accuracy [1]-[3].

It is well known that there is a trade-off between computation time and hardware size. In addition, high operation frequencies and large amounts of hardware resources contribute to increasing power consumption [4]. These key issues are normally carefully analyzed during the design flow of an embedded system with the aim of meeting the required specifications. However, the problem of accuracy is hardly ever explored in the early stages of the design flow, even though it could limit hardware performance in a crucial way. Moreover, the functionality of most embedded systems is designed using floating-point arithmetic and computer-aided design tools. Meanwhile, the hardware implementation of the system is carried out using fixed-point arithmetic and finite word-length. For these reasons, the degradation of the system performance, due to signal and system quantization, should be considered in the design of cutting-edge applications. The main problem consists in designing fixed-point hardware able to provide the required accuracy. Quantization errors are easy to manage in common arithmetic operations (e.g. sums or products). However, the implementation of nonlinear functions, such as the activation function of ANNs, or the membership functions of NFSs, is not an easy task, mainly when online training algorithms are involved.

Despite the importance of proper specification of the accuracy of this kind of nonlinear functions, very few methods incorporate it as a design parameter [5], [6]. In this paper, a controlled-accuracy scheme, suitable for the implementation of nonlinear functions is proposed. The scheme is based on Taylor's Theorem and the Lagrange form of the remainder. A systematic design methodology which guarantees the accuracy of the approximation is provided. The development of a hardware co-processor that implements the proposed scheme using a field programmable gate array (FPGA) is presented. In addition, the method is applied to the development of an individualized monitoring system for real-time support of a smart environment.

This work has been partially funded by the Basque Government under Grant IT733-13, and the Spanish Ministry of Economy and Competitiveness under Grant TEC2013-42286-R.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.95

Experimental results are provided for both online training and feed-forward computation.

The rest of the paper is organized as follows: Section II presents the proposed method. The approximation of two nonlinear functions, widely used in soft computing algorithms, the sigmoid function and the hyperbolic tangent, is explained in detail. Section III addresses the development of an efficient hardware co-processor that computes the proposed scheme and presents the implementation performance using an FPGA of the Xilinx Virtex 6 device family. In Section IV representative experimental results are provided. Finally, some concluding remarks are presented.

II. COMPUTATION OF NONLINEAR FUNCTIONS IN INTELLIGENT EMBEDDED SYSTEMS

Most soft computing techniques involve the computation of nonlinear functions. Thereby, artificial neural networks (ANNs) use nonlinear activation functions for weighting the neuron outputs. The class of S-functions, such as the sigmoid and hyperbolic tangent, is commonly used. Fuzzy systems (FSs) and hybrid neuro-fuzzy systems (NFSs) use membership functions to define fuzzy sets. Gaussian and S-shaped functions have proven very useful for modelling fuzzy information. Gaussian and S-functions are also widely used in kernel-based algorithms such as support vector machines (SVMs) or deep-learning architectures, among others.

The calculation of the above functions involves the computation of exponentiations, sums, products and divisions. Exponentiation and division are costly to implement in digital hardware, mainly if high precision is required. Several implementation methods have been proposed over the years. The most commonly used are look-up tables (LUTs) [7], bit-level mapping [8], piecewise linear methods [9], [10], [11], Taylor series expansion [12], and hybrid methods [13], [14].

The selection of the approximation method and its hardware implementation are key aspects that constrain the accuracy and the performance of the algorithm. Thus, too low accuracy produces poor performance, while an excess of it unnecessarily increases hardware resources and reduces the processing speed. Even though some researchers report the accuracy provided by their approximation methods, only a few of them provide a systematic design methodology able to guarantee the accuracy of the approximation.

Another important aspect to be considered is the differentiability of the approximated function. This is because meaningful learning and adaptation methods, based on gradient evolution, involve both the function and its derivative.

The approximation scheme could be applied to any differentiable function suitable for Taylor approximation. In particular, S-shaped and Gaussian curves present additional properties that allow further simplifications of the method. These families of curves have a bounded output range and a bounded *active input range* where Taylor theorem is used to approximate the function (i.e. Taylor regions). Outside the active input range, the function derivatives vanish and the function saturates to a constant value (i.e. saturation regions).

Let $f(x):\mathbb{R}\to\mathbb{R}$ be a (k+1)-times differentiable function around a given point $a\in\mathbb{R}$. Then, $f(x)$ can be approximated in any interval I containing a by means of a kth order Taylor polynomial:

$$f(x)=f(a)+f^{(1)}(a)(x-a)+\frac{f^{(2)}(a)}{2!}(x-a)^2+\ldots+\frac{f^{(k)}(a)}{k!}(x-a)^k. \quad (1)$$

The approximation error (i.e. remainder) in I can be bounded using the Lagrange form of the remainder or error

$$\left|R_k(x)\right|\le\frac{\left|x-a\right|^{k+1}}{(k+1)!}M_k \text{ if } \left|f^{(k+1)}(x)\right|\le M_k. \quad (2)$$

Equation (2) provides a means of dealing with the maximum allowable approximation error (i.e. $\varepsilon=\left|R_k(x)\right|$) as a design parameter. The proposed scheme can be organized into the following steps:

1) The allowed error, ε, that provides the required approximation accuracy is to be defined.
2) Both the Taylor regions and the saturation regions are to be determined. For simplicity, consider a single positive Taylor region $[0,t]$, and a single saturation region $[t,+\infty]$. The precise boundary between the Taylor region and the saturation region, t, is obtained as follows:

$$\left|f(t)-s\right|=\varepsilon, \quad (3)$$

with s being the value of the function inside the saturation region.

3) The Taylor range $[0,t]$ is split into a number of intervals of width $2r$ and centre a where a local approximation of the function is computed. Using (2), and taking into account that $\left|x-a\right|\le r$, $\forall x\in I=[a-r,a+r]$,

$$\left|R_k(x)\right|\le\frac{r^{k+1}}{(k+1)!}M_k\le\frac{r^{k+1}}{(k+1)!}\max\left|f^{(k+1)}(x)\right|, \quad (4)$$

where M_k is selected to be equal to the maximum of the absolute value of the (k+1)-order derivative of the function,

$M_k = \max\left|f^{(k+1)}(x)\right|$.

4) If n_k is the number of intervals, then $r = t/2n$. Replacing in (4), the minimum number of intervals is:

$$n_k = \frac{t}{2}\left(\frac{\max\left|f^{k+1}(x)\right|}{\varepsilon(k+1)!}\right)^{\frac{1}{k+1}} \tag{5}$$

The scheme has been implemented using a fixed-point fractional data format. Therefore, the word-length has to be able to provide the required precision. If N_i is the number of bits used to represent the integer part of the input data and N_f is the corresponding fractional part, then $x = x_{N_i-1}...x_0 \cdot x_{-1}...x_{N_f}$, where N_i depends on the width of the Taylor region:

$$2^{N_i} \geq t, N_i \geq \frac{\ln t}{\ln 2}, N_i \in \mathbb{Z}, \tag{6}$$

In addition, N_f should be able to represent those changes in the input Δx that produce changes in the function Δf equal to the maximum allowable error, that is to say:

$$\frac{\Delta f}{\Delta x} = \frac{\varepsilon}{2^{-N_f}}, \tag{7}$$

for small increments, $\Delta f / \Delta x \cong f^{(1)}(x)$. Therefore, introducing the maximum absolute value of the first order derivative in (7), the number of fractional bits can be obtained as follows:

$$N_f \geq \frac{1}{\ln 2}\left[\left|\ln \varepsilon\right| + \ln(\max\left|f^{(1)}(x)\right|)\right]. \tag{8}$$

In sum, given the desired error ε, and the order of the polynomial, k, the frontier of the Taylor region, t, is obtained using (3), the minimum number of intervals is provided by (5), and the local approximation of the function in each interval $[a-r, a+r]$ is obtained by means of (1). The data word-length should be selected accordingly.

Next, the approximation of two representative functions, widely used in soft computing methods, is provided: the sigmoid function and the hyperbolic tangent. The sigmoid is defined as follows:

$$f_S(x) = \frac{1}{1+e^{-x}}, \tag{9}$$

and the hyperbolic tangent is:

$$f_T(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{10}$$

The above functions and their first three derivatives are shown in Fig. 1 and Fig. 2, respectively. As can be seen, (9) and (10) share some common characteristics. Both functions exhibit some kind of symmetry with respect to the y-axis, the sigmoid verifies $f_S(-x) = 1 - f_S(x)$, while the hyperbolic tangent is strictly symmetric, that is to say, $f_T(-x) = -f_T(x)$. In addition, their output ranges are bounded; the sigmoid is bounded within (0,1) and the hyperbolic tangent within (-1,1). Moreover, both functions concentrate their main "activity" in a limited range of the x-axis (i.e. Taylor regions). Outside these ranges the functions saturate to constant values (i.e. saturation regions).

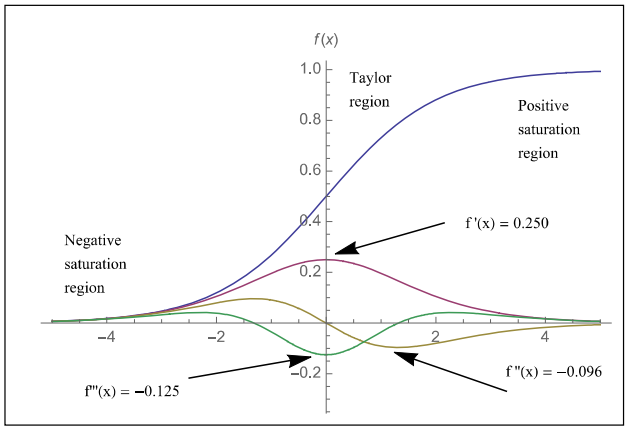

Fig. 1. Sigmoid function and its first three derivatives.

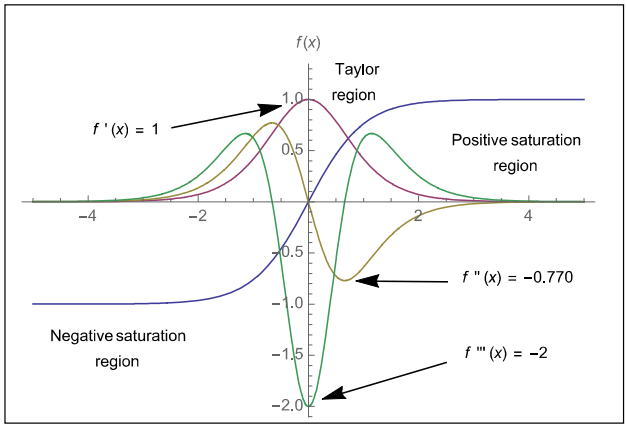

Fig. 2. Hyperbolic tangent and its first three derivatives.

A. Approximation of the Sigmoid

Different accuracies will be investigated, so, let ε be the maximum allowable approximation error. Then, to determine where the positive saturation region starts (see Fig. 1) we have to apply (3) with s=1,

$$\left|\frac{1}{1+e^{-t}}-1\right|=\varepsilon\,, \tag{11}$$

Solving (11) results in $t=\ln(\varepsilon^{-1}-1)$, with t being the boundary between the Taylor region and the saturation region.

Using (5) for a k-order Taylor scheme,

$$n_k \geq \frac{\ln(\varepsilon^{-1}-1)}{2}\left(\frac{\max\left|f^{(k+1)}(x)\right|}{\varepsilon(k+1)!}\right)^{\frac{1}{k+1}}. \tag{12}$$

The number of intervals for a first order scheme is:

$$n_1 \geq \frac{\ln(\varepsilon^{-1}-1)}{2}\left(\frac{0.096}{2\varepsilon}\right)^{\frac{1}{2}} = 0.109\varepsilon^{-1/2}\ln(\varepsilon^{-1}-1)\,, \tag{13}$$

where the maximum value of the second order derivative is 0.096. For a second order scheme:

$$n_2 \geq \frac{\ln(\varepsilon^{-1}-1)}{2}\left(\frac{0.125}{6\varepsilon}\right)^{\frac{1}{3}} = 0.138\varepsilon^{-1/3}\ln(\varepsilon^{-1}-1)\,, \tag{14}$$

with the maximum absolute value of the third order derivative being equal to 0.125 (see Fig.1).

Table I provides the boundary of the Taylor region and the minimum number of intervals for different errors using a first order and a second order Taylor approximation scheme, according to (13) and (14), respectively. In addition, this scheme provides suitable word-lengths, obtained by means of (6) and (8). As can be seen, the larger the order of the polynomial, the lower the required number of intervals in the Taylor region. In other words, the accuracy of the approximation scheme can be improved by using more terms in (1) or by refining the Taylor region segmentation. The first option would need more computation time and hardware resources, while the second one would increase memory size.

B. Approximation of the Hyperbolic Tangent

The positive saturation region starts where $f(x)$ satisfies (3) with s=1,

TABLE I. APPROXIMATION OF THE SIGMOID FUNCTION

Error (ε)	**Parameters of the approximation scheme**				
	Taylor region width (t)	***Number of intervals (n_1) 1st order scheme***	***Number of intervals (n_2) 2nd order scheme***	N_i	N_f
0.1	2.20	1	1	2	2
0.01	4.60	6	3	3	5
0.001	6.91	24	10	3	8
0.0001	9.21	101	28	4	12

$$\left|\frac{e^{t}-e^{-t}}{e^{t}+e^{-t}}-1\right|=\varepsilon\,, \tag{15}$$

The boundary between the Taylor region and the saturation region is $t=0.5\ln(2\varepsilon^{-1}-1)$. Using (5) for an k-order Taylor scheme,

$$n_k \geq 0.25\ln(2\varepsilon^{-1}-1)\left(\frac{\max\left|f^{k+1}(x)\right|}{\varepsilon(k+1)!}\right)^{\frac{1}{k+1}}, \tag{16}$$

The number of intervals for a first order scheme is:

$$n_1 \geq 0.25\ln(2\varepsilon^{-1}-1)\left(\frac{0.770}{2\varepsilon}\right)^{\frac{1}{2}} = 0.155\varepsilon^{-1/2}\ln(2\varepsilon^{-1}-1)\,, \tag{17}$$

where the maximum of the second order derivative is 0.770. For a second order scheme:

$$n_2 \geq 0.25\ln(2\varepsilon^{-1}-1)\left(\frac{2}{6\varepsilon}\right)^{\frac{1}{3}} = 0.177\varepsilon^{-1/3}\ln(2\varepsilon^{-1}-1)\,, \tag{18}$$

with the maximum absolute value of the third order derivative being equal to 2 at x=0 (see Fig.2).

Table II provides the results obtained for a first order and a second order approximation of the hyperbolic tangent according to (17) and (18). Minimum word-lengths computed by means of (6) and (8) are also included. As can be seen, given the approximation error, the Taylor region is wider for the sigmoid function than for the hyperbolic tangent. However, the second function needs more intervals than the first one, and, therefore, more bits are required. In other words, the computation of the hyperbolic tangent requires larger word-lengths than the sigmoid

function. It can be concluded that the digital implementation of the sigmoid function will use less resources and will require less computation time than the hyperbolic tangent, for the same accuracy. Fig. 3 shows the approximation of the sigmoid function obtained by applying the proposed method with an approximation error $\varepsilon = 0.01$ (see Table I).

TABLE II. APPROXIMATION OF THE HYPERBOLIC TANGENT

Error (ε)	Parameters of the approximation scheme				
	Taylor region width (t)	*Number of intervals (n_1) 1st order scheme*	*Number of intervals (n_2) 2nd order scheme*	N_i	N_f
0.1	1.47	2	2	2	3
0.01	2.65	9	5	2	7
0.001	3.80	39	14	3	10
0.0001	4.95	157	37	3	14

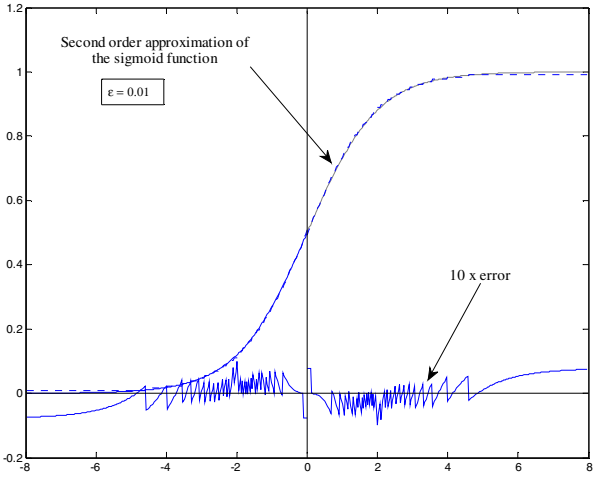

Fig. 3. Approximation of the sigmoid function ('---') and approximation error magnified by 100 for a maximum allowable error ε=0.01. The reference full-precision sigmoid is included in the background.

III. HARDWARE IMPLEMENTATION OF THE SECOND ORDER SCHEME

The architectural and technological features of FPGAs make them especially suited to developing single-chip embedded systems or hardware co-processors for algorithm acceleration. Current FPGA families combine logic blocks and interconnects, typical of traditional reconfigurable devices, with embedded cores, peripherals, and memory blocks. FPGA technology continues to be improved, bringing new challenges to the scope of intelligent embedded systems.

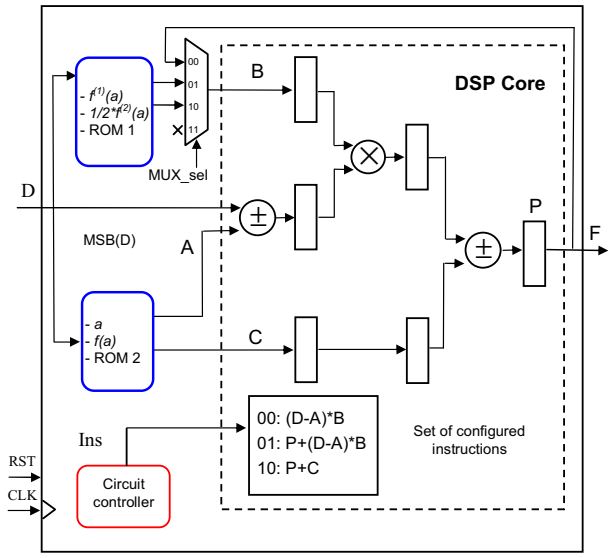

Fig. 4. Internal architecture of the circuit that performs the computation of a second order scheme. The main input to the circuit is the value of x (D). The output is the approximated function $f(x)$, (F).

Fig. 4 depicts the block diagram of the implementation of a nonlinear function using a second order approximation scheme. The proposed architecture is specially suited for high performance FPGA implementation. The main computation unit is a typical Digital Signal Processing (DSP) core. This embedded core is very useful for implementing computations like the one in (1) because it is faster and occupies a smaller area than a logic-based implementation. In this piece of work, without loss of generality, the Xilinx DSP48 has been used [15]. It is an 18x18 bit two-complement multiplier followed by a 48-bit sign-extended adder/subtracter and accumulator. These arithmetic operations are widely used in digital signal processing (e.g. digital filters). The DSP unit can be dynamically changed by enabling the specification of multiple operations using a set of user-defined arithmetic expressions (i.e. instructions). The specified operations are enumerated and can be selected through a single port on the generated core. As can be seen, the main input to the circuit is the (N_i+N_f)-bit data D. The output signal is the approximated function F.

Two ROM memories are used: ROM1 stores the function derivatives ($f^{(1)}(a)$,$f^{(2)}(a)$), while ROM2 is used to store the pairs of values (a,$f(a)$) for every Taylor interval. The memory words are addressed by means of the most significant bits of the input data. In addition, a 4-input multiplexer (MUX) is used to configure one of the DSP inputs (register B). Only three MUX inputs are required to implement a second order scheme: the value of the first order and second order derivative computed in the centre of the interval, and a feedback signal. The derivatives

are read from ROM1, while the feedback signal comes from the DSP core output.

Three different instructions are configured to compute the approximation:

$$f(x) = f(a) + f^{(1)}(a)(x-a) + \frac{f^{(2)}(a)}{2}(x-a)^2. \qquad (19)$$

The sequence of computations involved in (19) requires 7 clock cycles with the following configuration:

Cycle 1: Instruction = 00, MUX_sel = 10

Cycles 2, 3 and 4: Instruction = 00, MUX_sel = 00

Cycles 5 and 6: Instruction = 01, MUX_sel = 01

Cycle 7: Instruction = 10, MUX_sel = 01

The second order scheme was developed in VHDL language with the aid of the ISE Design Suite. The circuit was implemented using a Xilinx Virtex 6 family device. It requires only 1 embedded DSP block. Slice resources depend on the required word-length, which in turn depends on the allowed error. For example, the implementation of the sigmoid for a maximum allowable error $\varepsilon = 0.01$ depicted in Fig. 3 requires only 7 slices (11 LUTs); the core performs the computation of the sigmoid in only 7 clock cycles and is able to operate at 373.5 MHz. In other words, the evaluation of a nonlinear function, with an approximation error less than 0.01, using the second order scheme is performed in less than 20ns with the proposed FPGA implementation.

IV. LEARNING AND ADAPTATION WITH CONTROLLED ACCURACY

It is well known that learning and adaptation in ANNs and NFSs is difficult to implement in a hardware solution, mainly because of the complexity of the nonlinear calculus involved in the computations [4]. On the other hand, the evaluation of the algorithms, after training, is less precision-demanding and easier to implement than the learning process.

In the next part, a single hidden-layer feed-forward ANN, trained using a typical back-propagation (BP) gradient descent method (GDM), will be used to show the advantages of using the proposed controlled-accuracy approximation method. An Ambient Intelligence application (i.e. smart environment) is selected with the aim of highlighting the usefulness of soft computing techniques in the development of embedded systems [16]. The method should be integrated in the design flow of an embedded system in order to guarantee that the hardware implementation will behave properly. That is to say, the performance of the designed system will not be affected by the limitations of using fixed-point hardware implementations with finite word-length approximation of nonlinear functions.

The data set used in this work was obtained at the Essex intelligent dormitory (iDorm) [17]-[19]. It is a real ubiquitous computing environment comprising a number of embedded sensors, actuators, processors, and a heterogeneous network. A user spent five consecutive days in this room, and during this period, the interaction of the user with the environment was recorded. Seven input sensors were monitored: internal light level, external light level, internal temperature, external temperature, chair pressure, bed pressure, and time measured as a continuous input on an hourly scale. The controlled actuators were four variable intensity spot-lights, the desk and bed side lamps, window blinds, the heater, and the two PC-based applications. In this paper, the four continuous outputs (variable intensity spot lights) were considered.

The experimental data were split into a training set and a testing set consisting of 200 and 208 instances, respectively. Firstly, the ANN was trained using BP-GDM with the 64-bit floating point precision provided by Matlab tool. The number of neurons in the hidden layer was selected large enough to obtain a testing performance RMSE = 0.1. This performance is reached with a 7-32-4 topology. Only power-of-two sizes of the hidden layer were considered with the aim of simplifying digital hardware implementations of the network. Fig. 5 and Fig. 6 depict the evolution of the RMSE for the set of testing data, where the overall RMSE values were obtained as the average RMSE of all the four outputs. The full-precision curve is identified with the label "reference", and the corresponding RMSE values will be considered as the design objective for further fixed-point experiments.

1) Training the ANN using full precision and testing the network using finite word-length

In the first round of experiments, the same 7-32-4 ANN topology was trained using full-precision, while the evaluation of the testing set was performed using a finite word-length approximation of the sigmoid function based on the second order scheme. This procedure provides an insight into the consequences of using a computer-aided design tool for the development of soft-computing applications that will be implemented using a fixed-point digital hardware approach. Different accuracies were considered to implement the activation function (i.e. sigmoid): ε= 0.1, 0.01, and 0.001. As can be seen in Fig. 5, the performance of the system can be seriously degraded due to the lack of accuracy in the approximation of the activation function. Moreover, for the iDorm case application, a digital hardware implementation with $\varepsilon > 0.01$ should be carefully analyzed.

2) Training and testing the ANN using finite word-length

In the second round of experiments, the 7-32-4 ANN was trained and tested using the finite-precision fixed-point approximation scheme, the results are shown in Fig. 6. It can be concluded that the loss of performance introduced by using a finite word-length approximation of the sigmoid function is smaller when the network is trained using the target approximation. Even for large errors (e.g. see Fig. 6 with ε=0.1), the RMSE remains close to the reference curve. Therefore, the use of the controlled-accuracy scheme has two main advantages:

i) the consequences of using a fixed-point approximation of the activation function can be carefully analyzed, and ii) the word-length of the whole system can be sized to achieve the desired performance. As a consequence, both hardware resources and power consumption can be reduced.

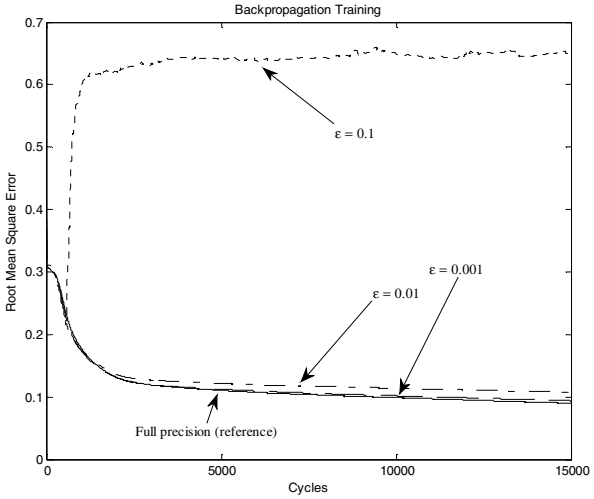

Fig. 5. Evolution of RMSE using Backpropagation Learning algorithm with a controlled accuracy approximation of the sigmoid function for different values of the maximum allowed error, ε. Training has been performed using full-precision 64-bit floating point computation, while testing has been performed with the fixed-point approximation of the sigmoid function

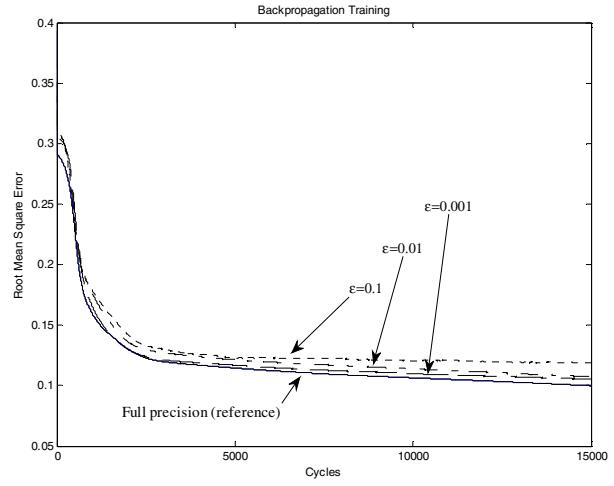

Fig. 6. Evolution of RMSE using Backpropagation Learning algorithm with a controlled accuracy approximation of the sigmoid function for different values of the maximum allowed error, ε. Both training and testing have been performed using the fixed-point approximation of the sigmoid function.

V. CONCLUSIONS

The development of intelligent embedded systems based on soft computing techniques require the approximation of nonlinear functions such as the sigmoid function, the hyperbolic tangent or the Gaussian function, among others. To tackle this problem, a controlled-accuracy approximation scheme, suitable for digital hardware implementation, is proposed. The method should be integrated into the design flow of embedded systems.

In addition, a high-performance hardware co-processor is developed. The proposed methodology is applied to the development of an ANN for a smart scenario. Experimental results show that the ANN performance is close to the full-precision performance (i.e. the reference) when the ANN is trained using the controlled-accuracy approximation scheme, even when large approximation errors are considered. On the contrary, when the ANN is trained using full-precision floating-point tools, the performance of the system could be seriously degraded when the ANN is implemented using a finite-precision hardware approach.

Further research will be done with the aim of analyzing the advantages of using the proposed scheme in the implementation of embedded systems based on different soft computing techniques.

REFERENCES

[1] A. R. Omondi and J.C. Rajapakse (Eds)., "FPGA Implementations of Neural Networks," Springer, The Netherlands, 2006.

[2] J. Misra, and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, pp. 239–255, 2010.

[3] I. del Campo, K. Basterretxea, J. Echanobe, G. Bosque, and F. Doctor, "A System-on-Chip Development of a Neuro-Fuzzy Embedded Agent for Ambient Intelligence Environments," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 42, pp. 501-512, 2012.

[4] G. Bosque, I. del Campo, and J. Echanobe, "Fuzzy systems, neural networks and neuro-fuzzy systems: A vision on their hardware implementation and platforms over two decades," Engineering Applications of Artificial Intelligence, vol. 32, pp. 283-331, June 2014.

[5] I. del Campo, R. Finker, J. Echanobe, and K. Basterretxea, "Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementations of artificial neurons," Electronics Letters, vol 49, pp. 1598-1600, 2013.

[6] B. Zamanlooy, and M. Mirhassani, "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function," IEEE Trans. Very Large Scale Integration Systems, vol. 22, pp. 39-48, 2014.

[7] K. Leboeuf, A. H. Namin, R. Muscedere, H. Wu, and M. Ahmadi, "High Speed VLSI Implementation of the Hyperbolic Tangent Sigmoid Function," in Proc. 3rd Int. Conf. on Convergence and Hybrid Information Technology, 2008, pp. 1070-1073.

[8] M. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," IEE Proceedings-Computers and Digital Techniques, vol. 150, pp. 403–411, 2003.

[9] C. Alippi, and G. Storti-Gajani, "Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning," Proc. of the IEEE International Sympoisum on Circuits and Systems, pp. 1505-1508, 1991.

[10] K. Basterretxea, J.M. Tarela, and I. del Campo, "Digital design of sigmoid approximator for artificial neural networks," Electronics Letters, vol 38, pp. 35-37, 2002.

[11] A. Armato, I. Fanucci, G. Pioggia, and D. De Rossi, "Low-error approximation of artificial neuron sigmoid function and its derivative," Electronics Letters, vol 45, pp. 1082-1084, 2009.

[12] Ch-J Lin and H-M Tsai, "FPGA implementation of a wavelet neural network with particle swarm optimization learning," Mathematical and Computer Modelling, vol. 47, pp. 982–996, 2008.

[13] K. Basterretxea, J.M. Tarela, I. del Campo., and G. Bosque, "An experimental study on nonlinear function computation for neural/fuzzy hardware design," IEEE Trans. Neural Networks, vol. 18, pp. 266–283, 2007.

[14] A. Armato, L. Fanucci, E.P. Scilingo., and D. De Rossi, "Low-error digital hardware implementation of artificial neuron activation functions and their derivative," Microprocessors and Microsystems, vol. 35, pp. 557–567, 2011.

[15] "LogiCORE IP DSP48 Macro v3.0, PG148," http://www.xilinx.com/, accessed 12nd July 2015.

[16] F. Sadri, "Ambient Intelligence: A Survey," ACM Computing Surveys, vol. 43, pp. 36:1–36:66, October 2011.

[17] F. Doctor, H. Hagras, and V. Callahan, "A Fuzzy Embedded Agent-Based Approach for Realizing Ambient Intelligence in Intelligent Inhabited Environments," *IEEE Transactions on System, Man, and Cybernetics-Part A*, vol. 35, no. 1, pp. 55-65, Jan. 2005.

[18] F. Doctor, H. Hagras, and V. Callahan, "A Type-2 Fuzzy Embedded Agent to Realise Ambient Intelligence in Ubiquitous Computing Environments," *Information Sciences*, vol. 171, pp. 309-334, 2005.

[19] H. Hagras, F. Doctor, V. Callahan, and A. Lopez, "An Incremental Adaptive Life Long Learning Approach for Type-2 Fuzzy Embedded Agents in Ambient Intelligent Environments," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 1, pp. 41-55, Feb. 2007.