ANN in Hardware with Floating Point and Activation Function Using Hybrid Methods

Maicon A. Sartin\textsuperscript{a,b}, Alexandre C. R. da Silva\textsuperscript{b}
\textsuperscript{a} Department of Computing, Universidad do Estado de Mato Grosso - UNEMAT, Colider, MT, Brazil
Email: mapsartin@unemat.br
\textsuperscript{b} Department of Electrical Engineering, Universidade Estadual Paulista - UNESP, Ilha Solteira, SP, Brasil
Email: acsilsa@dee.feis.unesp.br

Abstract— Artificial neural networks are bio-inspired models used mainly in the problems solution with nonlinear behavior. Reconfigurable devices (FPGA) are widely employed in the implementation of artificial neural networks. The contributions of this work is in the implementation of a Multilayer ANN with four neurons, one hidden layer and hyperbolic tangent for activation function. A nonlinear function approximation performed for system validation of artificial neural network. This paper makes analysis of nine scenarios with several methods to implementation of the activation function in ANN. Two hybrid methods are used in the approximations related to PWL method, Combinatorial and RALUT. Results are compared between distinct scenarios and the activation function with recent literature. The results were examined by performance, FPGA area and errors absolute and relative. Activation function has achieved the best average error in $10^{-4}$ and the ANN in $10^{-3}$. ANN system is suitable for target application and with portability to others platforms.

Index Terms— Artificial neural networks, floating point, FPGA, activation function, hyperbolic tangent.

I. INTRODUCTION

A rtificial Neural Networks (ANN) are popular between researchers, especially for provide the development with parallel processing techniques, robustness in the results and wide range of applications in robotics, image processing, automation and control, among others. ANN often are used in solving nonlinear problems and real-time applications, which mathematical model is expensive for hardware implementation.

The use of FPGA to implement in hardware ANN has increased, manly by performance associated with a high level of parallelism and flexibility provided by reconfigurable devices. Three characteristics are essential in the development of ANN in reconfigurable device, data representation, type of architecture and method for nonlinear activation function (AF).

The data representation is divided into point fixed and floating. The fixed point is used for simplify the mathematical operations, but this affect the accuracy by implementation of the truncation. The floating point going to facilitate the maintenance and expansion of the system, because IEEE Std 754 determines of organized form the sign, exponent and fraction of the data.

The type of architecture is a challenge especially by math operations and nonlinearity of the activation function. There are many works investigating improvements in the implementation of artificial neurons and activation functions [1]–[3]. An important feature is the optimization in all components to find a tradeoff between area and performance individually and expanding for all system. The type of architecture and dedicated hardware should be chosen with care, because going to have a large impact in the system optimization and development [2]. An artificial neuron optimized can drive to a more efficient ANN and allow development of great systems, this can be made by dedicated hardware [2], digital techniques [4] or by concept of virtual layers [1].

The type of system architecture determines the flow of execution and delays in the neuron data computing and consequently in ANN. The four main architectures are serial, parallel partially, fully parallel and PCNN (Pulse-Coupled Neural Network). Serial system contains accumulator type structures to minimize the utilization of area, but this can cause loss of performance in the execution the application. Parallel partially is defined by parallel multipliers and cascade adders to bring a threshold between area and performance [3]. Fully parallel has a completely parallel adder that belong its structure and be its main difference about the partial system, this fact increases its performance and size. The PCNN method has its structure differentiated to implement the neuron based on radial basis function networks.

In the neuron is applied a threshold according with the type of AF chosen after the data processing. The direct implementation of AF (hyperbolic tangent) is impractical, because it is an exponential series infinite. Several techniques are used in the employment of nonlinearity in AF, the four main are: polynomials with different orders [5], LUT (Look-Up Table) [6], PWL (PieceWise Linear) [7] and Hybrid [8].

LUT method is often used for AF approximation by its simplicity in the development with satisfactory per-
performance. Dedicated memory on the FPGA is capable of achieving high accuracy with few used logical resources. RALUT (Range Addressable Lookup Tables) is an improvement of the LUT; this method maps a set of inputs to one output. The PWL method uses a certain amount of linear parts for simplify the complexity of the function and reduce the area. The literature contains several different types of hybrid methods combining two or more methods, the hybrid methods bring a more efficient approach [9].

The standardization is a form to facilitate the development, maintenance, expansion and understanding of system. This paper presents a standardized hardware architecture for an ANN Multi-Layer Perceptron (MLP) that performs the approximation of nonlinear function by (1). The ANN is divided in three main parts a control unit (CU), processing unit (PU) and storage unit (SU). The main feature in the CU is the implementation of three finite state machine (FSM) and counters that provide the correct flow of data throughout the system. PU is divided in two parts the Processing Element (PE) and AF to implement the three layers of the ANN, with a total of 4 neurons and topology 1–2–1. The SU does storage of data and results of system using different technologies in dedicated memory. The computation of the data were done in floating point IEEE Std 754 (32 bits). All neurons contain a hyperbolic tangent AF developed with two approaches based on the hybrid method, HPR (Hybrid PWL with RALUT) and HPC (Hybrid PWL with Combinational).

The rest of the paper is organized as follows: Section II describes the system architecture and methodology employed in the development of the main components. The results and the comparison between implementations of high and low level are presented in Section IV. Section V presents the conclusion the article with a discussion of contributions achieved.

II. SYSTEM ARCHITECTURE

The ANN was developed in programming language of high-level with the Matlab (without toolbox). The Matlab has the function of data preparation and comparison of the results between hardware and software. The designer is assisted in the choice of components of low-level with development in high-level.

For supervised training of the ANN was adopted the delta rule in high-level language and offline mode. The stopping criterion used was the quadratic error with limit of $10^{-12}$, the convergence occurred with 56,987 epochs. The training uses 256 samples of input, according to desired value at the output of the network. In all implementations in hardware were transported the weights to the FPGA device with the same values in high-level. In the development of ANN in hardware (FPGA) was used the tool CAD (Computer Aided Design) ISE 14.2 of the Xilinx.

The system implements an ANN MLP with three layers, four neurons and hyperbolic tangent AF in all. ANN system was validated using a nonlinear function to the sinusoid approximation as (1), the value of the system input is $x$ and $i$ is the index of the samples.

$$S(x_i) = \frac{x_i}{e^{0.03}} \times \sin(0.03 \times x_i) \quad (1)$$

The standardization is a form to facilitate the development, maintenance, expansion and understanding of system. This paper presents a standardized hardware architecture for an ANN Multi-Layer Perceptron (MLP) that performs the approximation of nonlinear function by (1). The ANN is divided in three main parts a control unit (CU), processing unit (PU) and storage unit (SU). The main feature in the CU is the implementation of three finite state machine (FSM) and counters that provide the correct flow of data throughout the system. PU is divided in two parts the Processing Element (PE) and AF to implement the three layers of the ANN, with a total of 4 neurons and topology 1–2–1. The SU does storage of data and results of system using different technologies in dedicated memory. The computation of the data were done in floating point IEEE Std 754 (32 bits). All neurons contain a hyperbolic tangent AF developed with two approaches based on the hybrid method, HPR (Hybrid PWL with RALUT) and HPC (Hybrid PWL with Combinational).

The rest of the paper is organized as follows: Section II describes the system architecture and methodology employed in the development of the main components. The results and the comparison between implementations of high and low level are presented in Section IV. Section V presents the conclusion the article with a discussion of contributions achieved.

II. SYSTEM ARCHITECTURE

The ANN was developed in programming language of high-level with the Matlab (without toolbox). The Matlab has the function of data preparation and comparison of the results between hardware and software. The designer is assisted in the choice of components of low-level with development in high-level.

For supervised training of the ANN was adopted the delta rule in high-level language and offline mode. The stopping criterion used was the quadratic error with limit of $10^{-12}$, the convergence occurred with 56,987 epochs. The training uses 256 samples of input, according to desired value at the output of the network. In all implementations in hardware were transported the weights to the FPGA device with the same values in high-level. In the development of ANN in hardware (FPGA) was used the tool CAD (Computer Aided Design) ISE 14.2 of the Xilinx.

The system implements an ANN MLP with three layers, four neurons and hyperbolic tangent AF in all. ANN system was validated using a nonlinear function to the sinusoid approximation as (1), the value of the system input is $x$ and $i$ is the index of the samples.

$$S(x_i) = \frac{x_i}{e^{0.03}} \times \sin(0.03 \times x_i) \quad (1)$$

The system validation is applied a methodology divided in three steps: data preparation, system implementation and analysis of the results. In preparation of the data were defined the weights, data entry of the system and AF, all respecting with the IEEE Std 754 (32 bits) for carry to the hardware.

The implementation and comparison of the system was done in level high and low of abstraction. The results obtained in hardware (FPGA) by the functional simulation were decoded in high-level to generate graphs and tables. The analysis of the results are presented in the form of errors and area usage. Results were obtained for different implementations of ANN, especially in changing the AF.

The relationship between the modules and components used in the system architecture is presented in Fig. 1, as the divisions in CU, PU, and SU. ANN System is a parallel partially architecture and prototyped in the Spartan-3e platform (Xilinx) with 50 MHz of operation. The development kit is relatively cheap and contains many user interfaces. For implementation in hardware (FPGA) was used VHDL, float point unit (FPU) and dedicated memory of the manufacturer.

![Figure 1. System Architecture of ANN](image)

A. Control Unit

The ANN flow of data is not admitted problems caused by concurrent execution of operations, e.g., the output of one layer can not be replaced before of be acquired by next layer. The CU is responsible by correct execution of the data flow in the ANN through counters and three state machines. The Fig. 2 presents system FSM of ANN formed by six states (Start, Load, Layer1, Layer2, Layer3, Store).

Each state has its output only updated during the execution of the state by way of a enable ready signal. In the rising edge of the clock signal can recover the
layer output data correctly. Then, the state enables the execution of the next state as the input variables acquired of other ANN modules. The states also consist of a reset signal to prevent the propagation of undesirable values in other times, different of the global signal for FSM. After execution all layers and storage of the result of ANN in memory, the state "Store DataOut", checks if the current entry is the last sample to end system, otherwise makes a new reading of the next sample and reruns the ANN.

![Figure 2. FSM Diagram for the ANN System.](image)

B. Storage Unit

SU consists memories type embedded used in the system, but usually the choose depend on the manufacturer and the family of FPGA. The system employ two different types of BRAM memory, a read only (ROM) and one for data writing (RAM). Both technologies use memory type for single port access data. The ROM type memory is related to the input data and the activation function. ANN results are stored in RAM to allow future transfers on buses. The words in the memory use single precision (32 bits). The output data of the system are stored in file by way of functional simulation for read and compare the results in the Matlab.

C. Processing Unit

PU contains the weighted sum of the its inputs and the weights related with the neuron, according to IEEE Std 754 (32 bits). FPU based on an IP Core (Xilinx Coregen) is used to do this processing. PU makes the data computing in neuron through two main parts AF and PE. The number of components (PE and AF) in the system is directly related with the number of inputs to each neuron and number of neurons in a given layer and the amount of layers of ANN. For each entry in a neuron contains a processing element, and if \( n \) is the number of inputs associated with neuron.

To have greater control of processing element was necessary create a structure that organizes the inclusion of inputs and outputs in the modules of floating point operations. This FSM subsystem have three states to make the organization of the inputs, outputs and control of different types of floating point operations.

E. Activation Function

Nonlinear AF is more complex to be made directly in circuits or low-end implementation hardware (FPGA). However, the advantage in its use is the high precision and robustness of the ANN results making its employ necessary for applications that requiring a precision more refined.

The \((th(x))\) response of the hyperbolic tangent function corresponds to (2). The curve “S” function is a characteristic with positive and negative values, Fig. 6, different of the sigmoid (Logistics) composes just positive values. In (2) the \( x \) variable is related with the partial output of neuron and \( \alpha \) slope of the hyperbolic tangent. Thus, the AF inserts the nonlinearity in ANN depending of a feature in its behavior.

\[
Th(x) = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}} \tag{2}
\]

The behavior of the AF is nonlinear and has a steeper curve before saturation of the two ends. Three features to be observed in the hyperbolic tangent function are: (i) there is symmetry between the \( y \) axis of function \([6], [10]\) (ii) For very large values of \( x \), independent of the signal \( Th(x) \rightarrow \pm 1 \); (iii) the central part of the function is near a linear equation and the variation is greater. The output function \((Th(x))\) is equal to its input values for \( x \) near zero \([6]\).

III. PROPOSED METHODS FOR THE APPROXIMATION OF THE HYPERBOLIC TANGENT FUNCTION

ANN was developed using two different methods, five HPR customized (scenarios 1 to 5) and four HPC (scenarios 6 to 9), Table I. The computing of the data in the neurons are in accordance to the IEEE 754 (32 bit) and AF module works with custom format, but the input and output of the modules are available with the same standard.

The nine implementations contains a multiplexer to make the selection of the intervals from the entries, using exponent and part of fraction, as presented in the Fig. 3 and 4. Several intervals and different amounts of elements are used to satisfy the accuracy of the system. The input encoding was performed to reduce the mapping address (Scenarios 1 to 5) or the simplified Boolean function (scenarios 6 to 9) corresponding to the selected interval.

The implementations discussed in the method contains features RALUT, because the number of bits used in the IEEE Std 754 and ANN are greater than used by the AF
TABLE I.
ANALYZED SCENARIOS FOR HYPERBOLIC TANGENT AF.

<table>
<thead>
<tr>
<th>N.</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HPR of 180 Elements</td>
</tr>
<tr>
<td>2</td>
<td>HPR of 212 Elements</td>
</tr>
<tr>
<td>3</td>
<td>HPR of 314 Elements</td>
</tr>
<tr>
<td>4</td>
<td>HPR of 340 Elements</td>
</tr>
<tr>
<td>5</td>
<td>HPR of 404 Elements</td>
</tr>
<tr>
<td>6</td>
<td>HPC of 180 Elements</td>
</tr>
<tr>
<td>7</td>
<td>HPC of 212 Elements</td>
</tr>
<tr>
<td>8</td>
<td>HPC of 340 Elements</td>
</tr>
<tr>
<td>9</td>
<td>HPC of 404 Elements</td>
</tr>
</tbody>
</table>

module. The amount of extra bits can reach 16 bits, i.e. the possibility of variation within a range.

The tests were performed with the AF separated before coupling with the ANN. In this subsystem are applied 500 input samples and stored in dedicated memory BRAM (Block RAM), with a simple state machine for making available data. Partial results of the AF with this set of components are more detailed in [11].

The amount of addressing bits of the scenarios are divided in two groups, the scenarios 1 and 2 with 8 bits and the scenarios 3 to 5 mapped in 9 bits addressing. In memory only positive values are stored of the response of AF (ROM). Before provide the output data is done a selection by the signal input value, as Fig. 7. Thus, with a word of 31 bits, in the worst case, the scenarios 1 and 2 can be stored in less than half a BRAM (16K). Scenarios 3 to 5 are not occupying one full BRAM and the remainder can store other data, such as weights and temporary data.

The division in smaller intervals allows a mapping for the dedicated memory by association the number of elements appropriate for each interval based in absolute errors, as presented in Table II. Just within of the AF module was used a data representation reduced, the values in all implementations were $S = 1$, $E = 3$, $F = 7$. Thus, “S” represents the sign, “E” exponent and “F” the fraction of the number-based in approach to floating point to make of the LUT mapping.

The scenarios 6 to 9 use the amount of elements specified in the Table II and make the mapping directly to the output of AF, Fig. 4. The scenarios 1 to 5 have mapping to a memory address that contains the output of AF.

The mapping of scenarios 6 to 9 are done through purely combinational logic. To make mapping in the HPC hardware each interval contains a sum of products (SOP) to each output bit. SOP was made based in the amount of elements in interval and simplified with the Quine-McCluskey algorithm [12].

The amount of output bits of the Boolean expression is the multiple of elements quantity in the interval, the rest is constant. The choose of exponent of output corresponding is done with the data entry and the result of the fraction, always with multiple of two. The output contains 8 bits because a bit of the output fraction is used to differentiate the exponents of each interval, when necessary. The Fig. 4 presents the architecture to mapping of the combinational circuit in hardware.

IV. RESULTS

The nine scenarios developed in level high and low of abstraction. The architecture for computation of the neuron was the same with except for the AF. Thus, in all implemented architectures the runtime is the same, because all AF spend two clock cycles. The total runtime in the ANN system is directly related to PU. In each sample the runtime in the ANN system is $3.1\mu s$ and with 128 samples is achieved approximately $399\mu s$. In high-level language runtime achieved is approximately 109 ms, so that the hardware is 275 times more fast that software.

The Fig. 5 determines the area usage by each architecture for the activation function and ANN system. In the results of the scenarios present a similarity in the
area usage and accuracy. The results in the AF system present most values below 15 slices, just the scenarios 6 and 7 with approximately 135 slices, similar to recent literature [7], [13]. But the same scenarios have had proportional values in the full ANN system, except the Scenario 5 that uses more resources of the FPGA due the amount of elements for mapping. The synthesis results were performed three times at least. HPC and HPR scenarios used one and four dedicated memory (BRAM) respectively in the system of ANN. In the HPR scenarios the BRAM serve just for system inputs. In all scenarios were used 20 (100%) of dedicated multipliers available in Spartan-3e platform.

### A. Approximation of Activation Function

The execution of the ANN for nonlinear function approximation defined in (1) were analyzed in all scenarios. In Fig. 6 presents the AF approximation in all scenarios with their ideal behaviors in relation to HT function. The scenarios 1, 6 and 7 had inferior results in relation to the others, but yet with good precision. This can be verified in Fig. 7, that contains a reduced scale for visualize adequately the precision in all scenarios.

$$E_{abs} = |f(x_k) - \hat{f}(x_k)| \quad (3)$$

$$E_{rel} = \frac{f(x_k) - \hat{f}(x_k)}{f(x_k)} \quad (4)$$

In analysis of the errors contains a truncation according to (5), which corresponds to the maximum error possible for determined data representation [9]. Thus, as all scenarios have the same outputs fraction ($f$) for the mapping, the utilization of (5) is possible in the analysis of the errors. There are different amounts of elements in the set of data available in the LUT, thus the truncation have to be verified according with the implementation for a suitable analysis. Overflow and underflow exceptions were excluded in the analysis of the errors, when necessary.

$$E_{truncMax} = 2^{-(f+1)} \quad (5)$$

### Table III and IV present an analysis of the results obtained with the proposed architectures for AF hyperbolic tangent and ANN system respectively. In the two Tables are defined errors absolute and relative found, along with their average and maximum values associated to each architecture.

Some scenarios of literature do not contain all characteristic specified in the Tables III. The most scenarios have precision similar or better than literature, in the last line of the Table III presents the best accuracy achieved in [14], but precision similar to the our work. This work does not present area results. The area usage by each scenario is shown in Fig. 5.

Figure 5. FPGA Area Utilization in the AF and ANN system

Sections IV-A and IV-B present the analysis of the errors in the approximation of HT AF and ANN system. The errors absolute and relative are found by (3) and (4) respectively. The relative maximum error was not related for not having metric comparison between the scenarios. But relative average error presents the behavior of the approximations suitably.

In the Table III can be verified that the HPR scenarios (1 to 5) (RALUT methods) present the best accuracy compared with HPC scenarios (6 to 9) (combinational methods). But the HPR scenarios need of dedicated memory (BRAM) different of the HPC scenarios, that for mapping outputs of the AF system have Boolean expressions.

The difference of the behavior between the approximations can be analyzed by relative average error of clear form through of the Figs. 8 and 9. These figures can be verified the differences of the intervals in each approximation. Analysis of the relative error was divided in two groups (HPR and HPC) with one figure for each group with all samples. The samples quantity between 0.002 e 0.004 in the Fig. 9 is higher than in Fig. 8, this demonstrates the superior accuracy in the HPR scenarios presented in the Table III.

Table III.

<table>
<thead>
<tr>
<th>Approx.</th>
<th>Error_Ave</th>
<th>Error_Ave</th>
<th>Rel_Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sc. 1 (180)</td>
<td>4.18 x 10^-2</td>
<td>1.45 x 10^-2</td>
<td>1.24 x 10^-3</td>
</tr>
<tr>
<td>Sc. 2 (212)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 3 (314)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 4 (340)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 5 (404)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 6 (180)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 7 (212)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 8 (314)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>Sc. 9 (404)</td>
<td>3.30 x 10^-2</td>
<td>6.80 x 10^-2</td>
<td>8.43 x 10^-4</td>
</tr>
<tr>
<td>PLAN [13]</td>
<td>4.30 x 10^-2</td>
<td>7.60 x 10^-2</td>
<td>-</td>
</tr>
<tr>
<td>Alippi [13]</td>
<td>4.30 x 10^-2</td>
<td>7.60 x 10^-2</td>
<td>-</td>
</tr>
<tr>
<td>Comb. [9]</td>
<td>3.90 x 10^-3</td>
<td>1.70 x 10^-3</td>
<td>-</td>
</tr>
<tr>
<td>Hybrid [6]</td>
<td>2.00 x 10^-3</td>
<td>1.70 x 10^-3</td>
<td>-</td>
</tr>
<tr>
<td>Hybrid [14]</td>
<td>4.61 x 10^-4</td>
<td>1.35 x 10^-4</td>
<td>1.38 x 10^-3</td>
</tr>
</tbody>
</table>

Analyzing the Figs. 8 and 9, and the Table II can be defined equality between the scenarios in each group respecting the characteristics discussed in Section II. Samples will be analyzed only in the first half of the graphs, by symmetry of the HT. The Intervals between [0;125] and [235:250] have same response for all scenarios of each group, due the equality between the last and the first three lines of the Table II.

The relationship between the increase in accuracy and the number of elements in the interval can be observed at various moments in Figs. 8 and 9. In HPR group the scenarios 1 and 2 have the same response, except the interval between samples 220 and 235 which scenario 2 has obtained improvement by greater amount of elements. This situation occurs similarly in the scenarios 4 and 5 by changing the samples in the same interval. Already the difference in accuracy between the scenarios (1 and 2) and (4 and 5) is visible in all intervals changed. In HPC group, the scenarios (6 and 7) and (8 and 9) can be compared of similar form, how made in the discussion with HPR group in the scenarios (1 and 2) and (4 and 5) respectively.

B. Approximation of Artificial Neural Network

The approximations made in the HT function achieved satisfactory results bring high data precision in ANN, when compared to the ideal result in (2). The comparison between the implementation on FPGA with the optimal value contains adequate validation with low error rate. The implementations results in the ANN system present a suitable hardware with floating point. In Fig. 10 present the approximation of ANN system with 128 samples and precision in approximately 10^-3. In Fig. 11 present the same results, but with reduced scale only 30 points and the approximations with the same results with line and point for facilitate the visualization. The Scenario 1 was the only with behavior unstable, all the others scenarios follow the ideal curve with suitable precision.

In the execution of ANN system the scenarios 8 and 9 have the same results for ANN system. Analyzing the tradeoff between area and accuracy the scenario 8 is the best choice between all. Scenario 2 is a good option by have a little table and great precision in the execution of AF and ANN system.
The analysis of the relative average error for ANN system are presented in the Figs. 12 and 13, as made in the Section IV-A for AF system. In the Fig. 12 presents the scenario 1 as the worst scenario in the implementation of the ANN system, mainly in the extremities of the function approximated. In all the others scenarios were obtained similar behaviors, with the biggest error near of zero. The Figs. 12 and 13 demonstrate the superior accuracy in the HPR scenarios presented in the Table IV.

V. CONCLUSION

This work presented the implementation of a system in float point for approximating a function using nonlinear ANN. Nine implementations were developed in low level and one in high level of abstraction for comparison. The ANN consists of four neurons and hyperbolic tangent function in all neurons, all implementations into reconfigurable device were adequate to provide accuracy high in the system. Comparison of error and area usage with different techniques for the approximation of hyperbolic tangent AF was discussed.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Error Max</th>
<th>Error Ave</th>
<th>Rel Ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sc. 1 (180)</td>
<td>6.25 \times 10^{-2}</td>
<td>4.75 \times 10^{-3}</td>
<td>2.86 \times 10^{-1}</td>
</tr>
<tr>
<td>Sc. 2 (212)</td>
<td>9.63 \times 10^{-3}</td>
<td>2.99 \times 10^{-3}</td>
<td>4.21 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 3 (314)</td>
<td>9.17 \times 10^{-3}</td>
<td>1.75 \times 10^{-3}</td>
<td>3.00 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 4 (340)</td>
<td>9.17 \times 10^{-3}</td>
<td>2.10 \times 10^{-3}</td>
<td>3.47 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 5 (404)</td>
<td>9.17 \times 10^{-3}</td>
<td>2.69 \times 10^{-3}</td>
<td>3.39 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 6 (180)</td>
<td>1.20 \times 10^{-2}</td>
<td>5.98 \times 10^{-3}</td>
<td>7.04 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 7 (212)</td>
<td>1.20 \times 10^{-2}</td>
<td>5.84 \times 10^{-3}</td>
<td>7.34 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 8 (340)</td>
<td>1.31 \times 10^{-2}</td>
<td>5.01 \times 10^{-3}</td>
<td>6.26 \times 10^{-2}</td>
</tr>
<tr>
<td>Sc. 9 (404)</td>
<td>1.31 \times 10^{-2}</td>
<td>5.01 \times 10^{-3}</td>
<td>6.26 \times 10^{-2}</td>
</tr>
</tbody>
</table>

The approximations made in the HT function with satisfactory results and high data accuracy, when compared to the ideal result. The implementation on FPGA contains adequate validation with low error rate, as can be seen in Tables III and IV, providing a suitable system for implementations with ANN.

The high precision with low use of logical resources in approximating HT AF were satisfactory for all scenarios, but the scenarios 6 to 9 contain a slight advantage regarding the usage of combinational circuit and the best tradeoff between area and precision. The approximations demonstrated ability to act with quality in implementations of ANN.

The use of FPGA for applications ANN is suitable for the level of parallelism provided by approaches, the hardware runtime was 275 times more fast than software. The prototyping of system on another platform can be made just with integrating the modules and respecting their specific features, for bring portability and expanding to the system.

ACKNOWLEDGMENT

The researchers would like to thank the support of the Universidade do Estado de Mato Grosso (UNEMAT), Universidade Estadual Paulista (UNESP) and Laboratório de Processamento de Sinais e Sistemas Digitais (LPSSD).
Figure 13. Relative Error of the HPC Scenarios with 128 Points for ANN System.

REFERENCES


Maicon A. Sartin received the B.S. degree in Computer Engineering from UNORP, São José do Rio Preto, Brazil in 2004 and his M.S. degree in Computer Science from the Pontifical Catholic University of Rio Grande do Sul, Brazil in 2009. He is currently pursuing the Ph. D. degree at electrical engineering in UNESP, Ilha Solteira, Brazil. He is an assistant professor at the UNEMAT, Colider, Brazil. His research interests include digital systems, reconfigurable devices, parallel processing and intelligent systems.

Alexandre C. R. da Silva received the B. Sc. degree in Electrical Engineering from the Universidade de Mogi das Cruzes, Mogi das Cruzes, Brazil, in 1984, the M. Sc. and Ph. D. degrees in Electrical Engineering from the Universidade de Campinas (UNICAMP), Campinas, Brazil, in 1989. He received the degree of Associated Lecture from the Universidade Estadual Paulista (UNESP), Ilha Solteira, Brazil, in 2003. In 2007, he developed postgraduate stage at the University of Limerick, Limerick, Ireland. He is currently full Professor - MS6 in the Universidade Estadual Paulista. He has experience in Electrical Engineering with emphasis in Electronic Circuits and Development Tools for Synthesis and Mixed Systems Digital System (Digital - Analog), acting on the following topics: Smart Sensor Networks IEEE 1451, system (FPGA - DSP - Microcontrollers) and Synthesis of Digital Circuits. Fellow Research Productivity - PQ - Level 2 - CPq.