IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

Efficient VLSI Implementation of Neural Networks With Hyperbolic Tangent Activation Function

Manivasagam M.
Department of ECE ,
Sengunthar Engineering College, .
Tiruchengod.

Mr. Rahul. A
Department of ECE & Guide
Sengunthar Engineering College
Tiruchengod.

***Abstract*— Nonlinear activation function is one of the main building blocks of artificial neural networks. Hyperbolic tangent and sigmoid are the most used nonlinear activation functions. Accurate implementation of these transfer functions in digital networks faces certain challenges. In this paper, an efficient approximation scheme for hyperbolic tangent function is proposed. The approximation is based on a mathematical analysis considering the maximum allowable error as design parameter. Hardware implementation of the proposed approximation scheme is presented, which shows that the proposed structure compares favorably with previous architectures in terms of area and delay. The proposed structure requires less output bits for the same maximum allowable error when compared to the state-of-the-art. The number of output bits of the activation function determines the bit width of multipliers and adders in the network. Therefore, the proposed activation function results in reduction in area, delay, and power in VLSI implementation of artificial neural networks with hyperbolic tangent activation function.**

***Index Terms*— Hyperbolic tangent, neural networks, nonlinear activation function, VLSI implementation.**

I. Introduction

NEURAL networks have a wide range of applications in analog and digital signal processing. Hardware implementation of neural networks has been used in applications such as pattern recognition [1], optical character recognition [2], test of analog circuits [3], real-time surface discrimination [4], smart sensing [5], and identification of heavy ions [6].

The main building blocks needed for hardware implementation of neural networks are multiplier, adder, and nonlinear activation function. A lot of research has been done in digital implementation of multipliers and adders, which can be readily used leaving the nonlinear activation function as the most complex building block.

To implement the neuron, various nonlinear activation functions, such as threshold, sigmoid, and hyperbolic tangent can be used. Hyperbolic tangent and sigmoid are mostly used because their differentiable nature makes them compatible with back propagation algorithm. Both activation functions have an s-shaped curve while their output range is different. Because of the exponentiation and division terms present in sigmoid and hyperbolic tangent activation function, it is hard to realize the hardware implementation of these functions directly.

To solve the implementation problem, approximation methods are generally applied. These methods are based on piecewise linear approximation (PWL), piecewise nonlinear approximation, lookup table (LUT), bit-level mapping, and hybrid methods. Generally, in PWL approximation methods, the function is divided into segments and linear approximation is used in each segment. This method is used in [7]–[9] for the hyperbolic tangent and sigmoid function implementation. Another PWL approximation method is introduced in [10], [11]. Unlike other PWL methods, the developed method is not based on input domain segmentation and exploits lattice algebra-based centered recursive interpolation (CRI) algorithm.

The piecewise nonlinear approximation is similar to the PWL method with the difference that nonlinear approximation is used in each segment. This method is used in [12] to approximate the sigmoid function and scheme 4 of [9] is proposed for approximating both sigmoid and hyperbolic tangent.

In the LUT-based methods, input range is divided to equal sub-ranges and each sub-range is approximated by a value stored in LUT. This method is used in [13] to implement the hyperbolic tangent.

The bit-level mapping method approximates output based on a direct bit-level mapping of input. This method can be implemented using purely combinational circuits, and is used in [14] to implement the sigmoid function.

Hybrid methods use a combination of the aforementioned methods. Examples include [15], [16], which have used a combination of PWL and LUT methods for hyperbolic tangent activation function implementation.

The approximation error present in all methods affects the neural network performance. Research performed in [17] and [18] shows that the nonlinear activation function implementation with higher accuracy improves the learning and generalization capabilities of neural networks. However, implementations with higher accuracy require more silicon area and decrease the network operation speed. Therefore, having nonlinear activation function hardware structures with lower area and higher speed for a specified accuracy becomes a key issue.

In general, PWL and piecewise nonlinear approximation methods need multiplications while the LUT and bit-level mapping methods use no multipliers. An exception is the CRI-based method, which is a PWL approximation while it

Manuscript received April 19, 2012; revised October 22, 2012; accepted November 21, 2012. Date of publication January 11, 2013; date of current version December 20, 2013.

The authors are with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: zamanlo@uwindsor.ca; mitramir@uwindsor.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.
Digital Object Identifier 10.1109/TVLSI.2012.2232321

requires no multipliers. However, it requires large number of registers [10]. The choice of approximation method depends on the target implementation technology. The hardware implementation of neural networks is mostly done in FPGA or ASIC. Because the current FPGAs provide a large number of multipliers, the PWL and piecewise nonlinear approximation-based methods are appropriate selection for FPGA implementation. However, due to high area requirements and delay of the multipliers in ASIC implementation, LUTs and bit-level mapping are more suitable.

The focus of this paper is on ASIC implementation of hyperbolic tangent function. Hyperbolic tangent has an output range of [−1, 1] and is defined as follows:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (1)$$

In the method proposed in [8], PWL is used to approximate the first derivative of hyperbolic tangent. Then, the first derivative approximation is integrated to obtain the hyperbolic tangent function. Lebouf *et al.* [13] proposed a new LUT-based structure for the hyperbolic tangent activation function. The proposed structure is based on range addressable lookup table (RALUT) [19]. A hybrid architecture is proposed by Namin *et al.* [15], which uses a simple PWL approximation in combination with RALUT. Another hybrid structure is proposed by Meher [16], which is based on a linear approximation in combination with LUT. The values stored in LUT are determined by the proposed boundary selection method.

In this paper, a new hybrid architecture, which is based on linear approximation in combination with bit-level mapping is proposed. The proposed architecture takes into account maximum allowable error as the design parameter.

The proposed approximation scheme divides the input range to three different regions using different strategy in each region. A mathematical analysis of the proposed approximation scheme in each region is provided.

The mathematical analysis shows that the proposed scheme requires less output bits for the same maximum error compared to the previous architectures. The hardware implementation of the proposed structure is realized in CMOS 0.18 μm to show the efficiency of proposed structure in terms of area, delay, and product of area and delay compared to the previous architectures.

The proposed structure is used for implementing a 4-3-2 network in CMOS 0.18 μm. Post-layout simulation results show that the proposed structure results in a neural network implementation with lower area, delay, and power.

The rest of this paper is organized as follows. In the next section, the proposed approximation scheme is discussed. The mathematical analysis for selection of the minimum number of input and output bits is provided in Section III. The domain boundaries of different regions are found in Section IV. The proposed structure based on the mathematical analysis done is explained in Section V. Hardware implementation of the hyperbolic tangent function and comparison with existing structures is done in Section IV. Neural network implementation using the proposed structure is discussed in Section VII. Finally, conclusions are drawn in Section VIII.

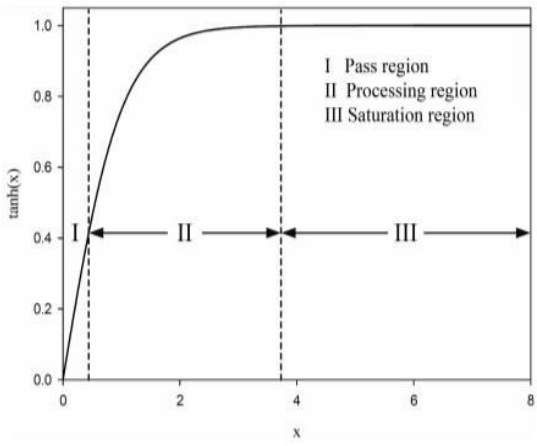

Fig. 1. Different regions of hyperbolic tangent function.

II. Proposed Approximation Scheme

In this section, mathematical analysis of the approximation scheme used for hardware implementation of hyperbolic tangent function is provided. The mathematical analysis in this and the following sections uses the basic properties of hyperbolic tangent function.

Hyperbolic tangent is an odd function

$$\tanh(-x) = -\tanh(x). \quad (2)$$

Using this property, only the absolute value of input is processed and the input sign is directly passed to the output.

The Taylor series expansion of hyperbolic tangent is as follows:

$$\tanh(x) = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \cdots \quad (3)$$

For small values of x, the higher order terms become small and can be ignored. Therefore, the hyperbolic tangent passes the small input values to output

$$\lim \tanh(x) = x \qquad x \to 0. \quad (4)$$

The output variation for large values of input is low

$$\frac{d\tanh(x)}{dx} = 0 \qquad x \to \infty. \quad (5)$$

Considering the two last properties, input range is divided to three regions. Region I in which the output is approximately equal to input is named pass region while because of low variation of output in region III, it is named saturation region. Region II includes the rest of input range, named processing region. Determining the boundary of each region is discussed later in Section IV. Different regions of hyperbolic tangent function are shown in Fig. 1.

A. Output Approximation in the Pass Region

The input and output of hyperbolic tangent function are represented as signed-magnitude notation. Therefore, considering the first basic property of the hyperbolic tangent function discussed in previous section, input sign bit is directly passed

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

to the output sign bit and only the absolute value of input is processed. Absolute value of input in binary format can be represented based on the following equation:

$$x = \sum_{k=-N_f}^{N_i-1} x_k \times 2^k = \sum_{k=0}^{N_i-1} x_k \times 2^k + \sum_{k=-N_f}^{-1} x_k \times 2^k \quad (6)$$

where N_i and N_f are the number of bits for integer and fractional part of the input and x_k is binary digit and can assume values 0 or 1.

In the pass region, output is approximated by passing the input to the output, which means that a linear approximation is used in this region. The inputs in the pass region include the values close to the origin, which are represented by fractional part of the input.

The absolute value of hyperbolic tangent function output is in the range of [0, 1], and can be shown as follows:

$$\tanh(x) = \sum_{k=-N_{\text{out}}}^{-1} y_k \times 2^k = \sum_{k=-N_f}^{-1} y_k \times 2^k + \sum_{k=-N_{\text{out}}}^{-(N_f+1)} y_k \times 2^k \quad (7)$$

where N_{out} is the number of bits used for representation of absolute value of output and y_k can assume one of the values 0 or 1.

Using (6) and (7), y_k is obtained as follows:

$$y_k = \begin{cases} x_k, & -N_f \le k \le -1 \\ 0, & -N_{\text{out}} \le k < -N_f. \end{cases} \quad (8)$$

Based on (8), the fractional part of input is shifted to left by $N_{\text{out}} - N_f$ bits and then is passed to the output.

B. Output Approximation in the Processing Region

Before going through the proposed approximation scheme in this region, a new parameter named N_{one} is introduced. This parameter is an indicator of position of the first occurrence of one in binary input, when scanned from left. Therefore, based on this parameter, the input range can be shown as follows:

$$2^{N_{\text{one}}} \le x < 2^{N_{\text{one}}+1}. \quad (9)$$

This input range is divided into equal sub-ranges. The number of these sub-ranges, N, is based on the equation shown as follows:

$$N = 2^{(N_{\text{one}}+N_f-i)} \qquad 0 \le i \le N_{\text{one}} + N_f. \quad (10)$$

Based on the value of N, sub-ranges within the input range are as follows:

$$2^{N_{\text{one}}}\left(1+\frac{j}{N}\right) \le x < 2^{N_{\text{one}}}\left(1+\frac{j+1}{N}\right) \qquad 0 \le j < N. \quad (11)$$

To have an approximation value close to all outputs corresponding to an input sub-range, the average value of outputs is considered as the approximation value as follows:

$$\sum_{k=0}^{\frac{2^{N_{\text{one}}+N_f}}{N}} \frac{\tanh\left(2^{N_{\text{one}}}\left(1+\frac{j}{N}\right)+k\times 2^{-N_f}\right)}{\frac{2^{N_{\text{one}}+N_f}}{N}}. \quad (12)$$

The total number of sub-ranges is found based on the fact that the difference between all values inside a sub-range and the approximation value found using (12) should be less than maximum allowable approximation error in this region, which results in the following equation:

$$\left|\tanh\left(2^{N_{\text{one}}}\left(1+\frac{j}{N}\right)\right) \le x < 2^{N_{\text{one}}}\left(1+\frac{j+1}{N}\right)\right) - \sum_{k=0}^{\frac{2^{N_{\text{one}}+N_f}}{N}} \frac{\tanh\left(2^{N_{\text{one}}}\left(1+\frac{j}{N}\right)+k\times 2^{-N_f}\right)}{\frac{2^{N_{\text{one}}+N_f}}{N}}\right| \le \epsilon_a \qquad 0 \le j < N \quad (13)$$

where ϵ_a is the maximum allowable approximation error in the processing region. ϵ_a depends on the maximum allowable error described in Section III.

C. Output Approximation in the Saturation Region

The hyperbolic tangent function reaches its maximum value in the saturation region, while at the same time output variation in this region is low. Therefore, all output values in this region are approximated by the maximum value representable by the output bits. Using (7), this value is equal to $1 - 2^{-N_{\text{out}}}$.

III. Selection of Number of Input and Output Bits

In this section, a mathematical analysis is presented, which allows for optimal finding of the number of input and output bits, required for hardware implementation of the proposed approximation scheme.

A. Selection of Number of Input Bits

The representation of absolute value of input in binary format can be shown using (6). The number of bits needed for the integer part depends on the input range. Therefore, to cover the range, it is required to have

$$2^{N_i} \ge r_i \quad (14)$$

where r_i is the input range. Using (14), N_i can be written in the following form:

$$N_i \ge \left\lceil \frac{\ln r_i}{\ln 2} \right\rceil \quad (15)$$

in which $\lceil\ \rceil$ is the ceiling function, which rounds its input toward the next highest integer.

In comparison, the number of bits used for fractional part is determined by the maximum allowable error. It should be noted that input region between two consecutive points x_1 and x_2 can be approximated as $\tanh(x_1)$ having an error lower than maximum allowable error provided that the following equation is satisfied

$$\tanh(x_2) - \tanh(x_1) \le \epsilon \quad (16)$$

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

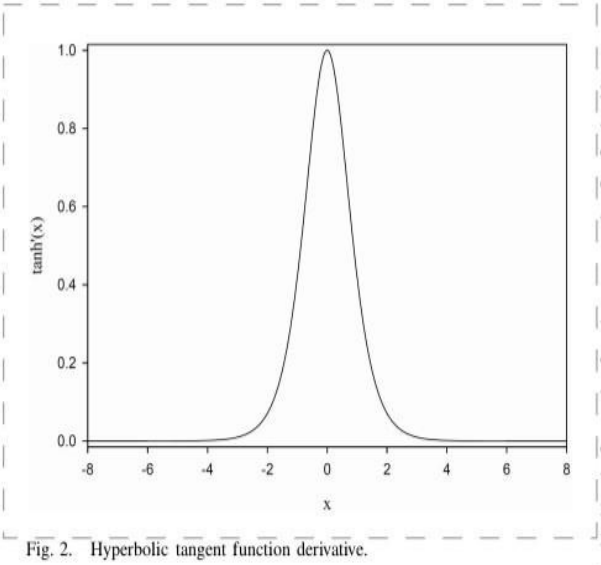

Fig. 2. Hyperbolic tangent function derivative.

where ϵ is the maximum allowable error.

The hyperbolic tangent change between two consecutive inputs is proportional to the hyperbolic tangent derivative shown in Fig. 2. Therefore, the maximum change of hyperbolic tangent function between two consecutive points occurs in the region, which is close to the origin. Based on (4), hyperbolic tangent output is approximately equal to its input in this region and therefore (16) can be simplified as follows:

$$x_2 - x_1 \leq \epsilon. \tag{17}$$

The difference between two consecutive points in the input is determined by the number of bits used for representing the fractional part of the input and is equal to 2^{-N_f}. Thus, (17) can be written as follows:

$$2^{-N_f} \leq \epsilon \tag{18}$$

which results in the following equation:

$$N_f \geq \left\lceil -\frac{\ln \epsilon}{\ln 2} \right\rceil. \tag{19}$$

Using (15) and (19), it can be written

$$N_{\text{inp}} = \left\lceil \frac{\ln ri}{\ln 2} \right\rceil + \left\lceil -\frac{\ln \epsilon}{\ln 2} \right\rceil \tag{20}$$

where N_{inp} is the minimum number of input bits required for representation of absolute value of input.

B. Selection of Number of Output Bits

As previously discussed, the hyperbolic tangent function is divided into three regions, including pass, processing, and saturation region. The number of bits required for output representation in these three regions, assuming a maximum allowable error of ϵ, depends on the properties of each region as will be discussed.

1) Pass Region: In the pass region, input is passed to the output. The pass region is where the inputs are close to the origin. These points are represented by the fractional part of the input. Therefore, minimum number of bits required in this region is equal to

$$N_{\text{out}} \geq N_f. \tag{21}$$

2) Processing Region: The output error in the processing region is composed of two elements. The first one is the approximation error while the second one is the quantization error of representing the approximated output. The total error caused by these sources should be less than maximum allowable error, which is shown as follows:

$$\epsilon_a + \epsilon_q \leq \epsilon \tag{22}$$

where ϵ_a is the maximum allowable approximation error and ϵ_q is the maximum quantization error of representing the approximated output.

The quantization error, ϵ_q, is proportional to the number of bits used for output representation. If rounding method is used for quantization of output, maximum quantization error is going to be equal to half of low significant bit, which is equal to $2^{-(N_{\text{out}}+1)}$. Therefore, the maximum allowable approximation error can be obtained as follows:

$$\epsilon_a = \epsilon - 2^{-(N_{\text{out}}+1)}. \tag{23}$$

It should be noted that the maximum allowable approximation error found through (23) was used in (13) in order to find the number of sub-ranges inside the processing region. Hence, the change in number of output bits changes the number of sub-ranges in this region.

3) Saturation Region: The approximated value of output in this region is equal to $1\text{-}2^{-N_{\text{out}}}$. This represents the maximum value of hyperbolic tangent function with an error, which is less than the maximum allowable error. This results in the following equation:

$$\left|2^{-N_{\text{out}}}\right| \leq \epsilon \tag{24}$$

which can be written in the following form:

$$N_{\text{out}} \geq \left\lceil -\frac{\ln \epsilon}{\ln 2} \right\rceil. \tag{25}$$

By comparing (25) and (19), the minimum number of bits needed for output representation in the saturation region is equal to N_f.

The minimum number of bits needed for output representation in the pass and saturation region is equal to N_f while there is no condition in the processing region. Therefore, the minimum number of bits needed for output representation of absolute value of hyperbolic tangent function using the proposed approximation scheme is equal to N_f.

The number of output bits required in the proposed approximation scheme is lower than the number of input bits while all previously developed architectures use the same number of input and output bits. Reduction in number of output bits may result in efficient hardware implementation. This will be investigated more in the next sections.

IV. Determining the Boundaries for Different Regions

In this section, using the maximum allowable error as design parameter, boundaries of each region are determined.

A. Pass Region

Based on the Taylor series expansion of hyperbolic tangent function for small values of x, higher order terms can be ignored. Therefore, the first three terms shown below are sufficient to present hyperbolic tangent function

$$\tanh(x) \sim x - \frac{x^3}{3} + \frac{2x^5}{15}. \tag{26}$$

Since in the pass region, input is passed to the output, boundary of pass region, x_{pa}, can be found using the following equation:

$$\left| \frac{x_{\text{pa}}^3}{3} - \frac{2x_{\text{pa}}^5}{15} \right| \leq \epsilon. \tag{27}$$

The x_{pa} obtained should be rounded to the nearest lower value representable by the input bits. Therefore, the quantized value of x_{pa} can be written in the following form:

$$x_{\text{paq}} = \frac{\left\lfloor x_{\text{pa}} \times \frac{2^{N_{\text{inp}}}}{r_i} \right\rfloor}{\frac{2^{N_{\text{inp}}}}{r_i}} \tag{28}$$

where $\lfloor\ \rfloor$ is the floor function and $0 \leq x \leq x_{\text{paq}}$ is considered as the pass region.

B. Saturation Region

The starting point of the saturation region is when the difference between hyperbolic tangent function and its approximation becomes equal to maximum allowable error. Therefore, the starting point of saturation region, x_s, is found as follows:

$$x_s = \tanh^{-1}\left(1 - 2^{-N_{\text{out}}} - \epsilon\right) = \frac{1}{2}\ln\left(\frac{2 - 2^{-N_{\text{out}}} - \epsilon}{2^{-N_{\text{out}}} + \epsilon}\right). \tag{29}$$

The x_s obtained should be rounded to the nearest higher value representable by input bits. Therefore, the quantized value of x_s can be written in the following form:

$$x_{s_q} = \frac{\left\lceil x_s \times \frac{2^{N_{\text{inp}}}}{r_i} \right\rceil}{\frac{2^{N_{\text{inp}}}}{r_i}} \tag{30}$$

where $x \geq x_{s_q}$ is considered as the saturation region.

C. Processing Region

The region between the pass and saturation region is considered the processing region, which can be shown as follows:

$$x_{\text{paq}} < x_{pr} < x_{s_q} \tag{31}$$

where x_{pr} is an input in the processing region.

V. Proposed Structure

Block diagram of the proposed structure is shown in Fig. 3. The hardware is composed of two main blocks, including hyperbolic tangent approximation and output assignment.

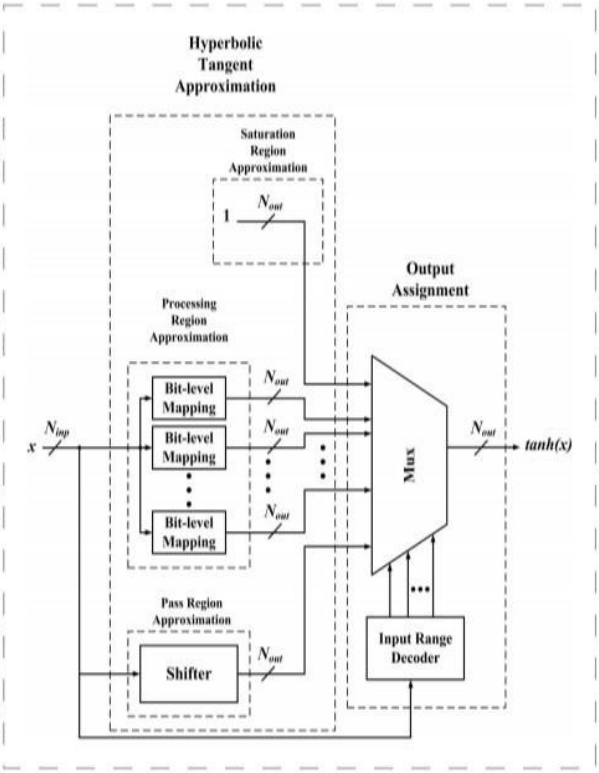

Fig. 3. Block diagram of the proposed structure.

A. Hyperbolic Tangent Approximation

This block is composed of three main blocks to approximate the hyperbolic tangent function in all three regions, including saturation, processing, and pass region. General arithmetic operations in each region can be described as follows.

1) Pass Region: In this region, fractional part of input is passed to the output. Based on (8), a shift to left by $N_{\text{out}} - N_f$ bits before passing the input to output is required.

2) Processing Region: For inputs in the processing region, a bit-level input mapping is required. The number of bit-level mapping blocks required is equal to the number of input ranges in this region. For each input range in the processing region, $\log_2^N$ bits after N_{one} bit of input should be mapped to output bits using the bit-level mapping. Using $\log_2^N$ bits after N_{one} bit covers all sub-ranges. The number of sub-ranges (N) is calculated using (13) while the output of each sub-range is found using (12). The bit-level mapping can be implemented using a combinational circuit.

3) Saturation Region Approximation: In this region, hyperbolic tangent function is approximated by the maximum value representable by output bits, and can be realized by setting all output bits to one.

B. Output Assignment

The input range decoder detects the N_{one} introduced previously, which is set by input range and region of operation, respectively. Depending on the input range, a multiplexer is used to obtain the appropriate output value.

To illustrate the proposed approximation scheme and structure, an example is presented. The example shows different steps of the design procedure.

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

TABLE I
OUTPUT VALUE FOR DIFFERENT INPUT RANGE AND SUB-RANGES

Input Range		Input Sub-Range		Output Value
x_{i1}	x_{i2}	x_{s1}	x_{s2}	
4	8	—	—	0.96875
2	4	—	—	0.96875
1	2	.875	2	0.96875
		1.75	.875	0.96875
		.625	1.75	0.9375
		1.5	.625	0.90625
		.375	1.5	0.90625
		1.25	.375	0.875
		.125	1.25	0.84375
		1	.125	0.78125
0.5	1	0.9375	1	0.75
		0.875	0.9375	0.71875
		0.8125	0.875	0.6875
		0.75	0.8125	0.65625
		0.6875	0.75	0.625
		0.625	0.6875	0.5625
		0.5625	0.625	0.53125
		0.5	0.5625	0.5
0	0.5	—	—	Input

Example: Design procedure for $\epsilon = 0.04$ considering an input range of $(-8, 8)$.

1) *Determining the Number of Input and Output Bits:* using (20), we have $N_{inp} = 8$. The minimum number of output bits required is equal to N_f. Using (19), we have $N_{\text{out}} = 5$.

2) *Determining the Boundaries of Pass, Processing, and Saturation Regions:* Using (27)–(31) the pass, saturation, and processing region boundaries are found equal to $x_{pa_q} = 0.5$, $x_{s_q} = 1.65625$, and $0.5 < x_{pr} < 1.65625$.

3) *Output Assignment in Pass Region:* In this region, the fractional part of input is shifted to left by $N_{\text{out}} - N_f$ bits and passed to the output. Therefore, no shift is required in this example and the fractional part of input is directly passed to the output.

4) *Output Assignment in Saturation Region:* In saturation region, the output value is equal to $1 - 2^{-5}$ or 0.96875.

5) *Output Assignment in Processing Region:* First, the maximum allowable approximation error is found using (23), which is equal to 0.024. Then, the number of sub-ranges, N, is found using (13). Finally, sub-ranges are found using (11) and the appropriate value of each sub-range is assigned using (12).

Table I summarizes these values for different input ranges and sub-ranges.

Also, quantization error, approximation error, and total error for the considered case are shown in Fig. 4(a)–(c). The approximated output and ideal output are shown in Fig. 4.

6) *Designing the Proposed Structure:* As can be seen from Table I, there are five different input ranges in the considered example. The input range is detected by the input range decoder.

The input range decoder detects the input range using N_{one}. Table II shows the input range decoder truth table, which can be implemented using a fully combinational circuit.

TABLE II
INPUT RANGE DECODER

Input Range	x_{i1}	x_{i2}	x_2	x_1	x_0	x_{-1}
r_1	4	8	1	X	X	X
r_2	2	4	0	1	X	X
r_3	1	2	0	0	1	X
r_4	0.5	1	0	0	0	1
r_5	0	0.5	0	0	0	0

TABLE III
BIT-LEVEL MAPPING FOR THE INPUT RANGE $1 < x < 2$

x_0	x_{-1}	x_{-2}	y_{-1}	y_{-2}	y_{-3}	y_{-4}	y_{-5}
0	0	0	1	1	0	0	1
0	0	1	1	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	1	1	1	0	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	1	0
1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1

TABLE IV
BIT-LEVEL MAPPING FOR THE INPUT RANGE $0.5 < x < 1$

x_{-1}	x_{-2}	x_{-3}	y_{-1}	y_{-2}	y_{-3}	y_{-4}	y_{-5}
0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	1	0	0
1	0	0	1	0	1	0	1
1	0	1	1	0	1	1	0
1	1	0	1	0	1	1	1
1	1	1	1	1	0	0	0

The input range $0 < x < 0.5$ (r_5) is in the pass region. Considering that N_{out} is equal to N_f, no shift is required and the hyperbolic tangent is approximated by passing the fractional part of input to output directly.

The input ranges $0.5 < x < 1$ and $1 < x < 2$ (r_4 and r_3) are in the processing region. Therefore, for hyperbolic tangent approximation, a bit-level mapping is required in these input ranges.

In the input range $1 < x < 2$, the N_{one} is equal to 1 and the number of sub-ranges (N) is equal to 8. Therefore, a bit-level mapping on the $\log_2^8 = 3$ bits after the N_{one} bit of the input is required to generate the output. Table III shows the required bit-level mapping.

For input range $0.5 < x < 1$, the N_{one} is equal to 0 and the number of sub-ranges (N) is equal to 8. Therefore, a bit-level mapping on the $\log_2^8 = 3$ bits after the N_{one} bit of the input is required to generate the output. Table IV shows the required bit-level mapping.

These tables can be implemented using a purely combinational circuit.

In the input ranges $2 < x < 4$ and $4 < x < 8$ (r_2 and r_1), input is in the saturation region and hyperbolic tangent is approximated by setting all output bits to 1.

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

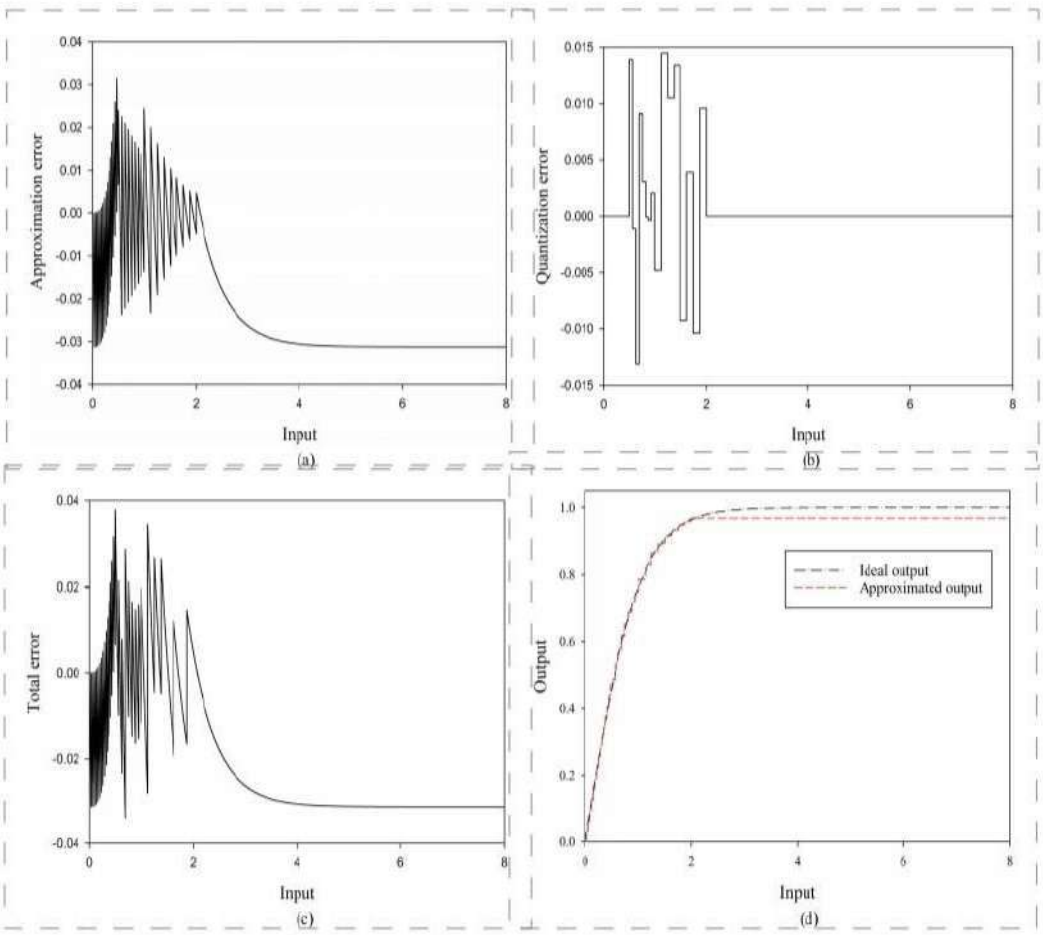

Fig. 4. (a) Approximation error. (b) Quantization error. (c) Total error. (d) Ideal and approximated output.

The multiplexer assigns the output value using the input range decoder. For each input range, the multiplexer transfers the approximation in that range to the output.

The hardware implementation of the considered example is shown in Fig. 5. It should be noted that the bit-level mapping needed for input ranges r_3 and r_4 is implemented using Tables III and IV while the input range decoder is implemented based on Table II.

VI. Hardware Implementation of the Hyperbolic Tangent Function and Comparison With Existing Structures

The proposed structure in the previous section is implemented with maximum allowable errors of 0.02 and 0.04. The proposed structure is coded using Verilog hardware description language and synthesized by Synopsys Design Compiler using TSMC 0.18-μm library.

Meher [16] has synthesized his proposed structure using TSMC 90-nm library while the synthesis of all other previously developed architectures is done using 0.18-μm library. To have a fair comparison between all architectures, we have coded the design in [16] in Verilog and synthesized using TSMC 0.18-μm library. It should be noted that signed-magnitude notation is used for input and output representation.

The comparison of different structures for $\epsilon = 0.04$ and $\epsilon = 0.02$ is summarized in Tables V and VI. These tables include the input range, number of input bits, number of output bits, maximum error after design and synthesis results, which are area and delay. The maximum error after design is evaluated for 10^6 points uniformly distributed in the input range [12]. Also, to have a comparison of number of cells required for different designs, the gate count measure, which is the design area normalized with respect to two input NAND gate area is included. Moreover, considering that both area and delay are important in hardware design, the area×delay is included in these tables too.

The method used by Lin and wang [8] is based on PWL approximation, which requires multiplication that has high area requirement and delay.

The other three previously proposed architectures use LUT. In [13], two LUT-based structures are proposed to implement the hyperbolic tangent function. In the first structure, 512 and 1024 points for errors of $\epsilon = 0.04$ and $\epsilon = 0.02$ are stored in LUT. The second structure is based on RALUT. Using RALUT, number of stored points for errors of $\epsilon = 0.04$ and $\epsilon = 0.02$ is reduced to 61 and 127. The reduction in number of stored points reduces the area consumption.

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

TABLE V

COMPARISON OF DIFFERENT STRUCTURES FOR $\epsilon = 0.04$

Structure	Input Range	N_{inp}	N_{out}	Maximum Error	Area (μm^2)	Gate Count	Delay (ns)	Area × Delay ($\mu m^2 \times ns$)
Scheme-1 [8]	(−8, 8)	24	24	0.0430	32069.83	3214	903	2.896×10^7
LUT [13]	(−8, 8)	8	8	0.0365	9045.94	907	2.15	1.944×10^4
RALUT [13]	(−8, 8)	8	8	0.0357	7090.40	711	1.85	1.311×10^4
Hybrid [15]	(−8, 8)	8	8	0.0361	3646.83	366	2.31	8.424×10^3
Optimized LUT [16]	(−8, 8)	8	8	0.0401	954.67	96	2.09	1.995×10^3
Proposed	(−8, 8)	8	5	0.0378	695.22	70	0.95	6.604×10^2

TABLE VI

COMPARISON OF DIFFERENT STRUCTURES FOR $\epsilon = 0.02$

Structure	Input Range	N_{inp}	N_{out}	Maximum Error	Area (μm^2)	Gate Count	Delay (ns)	Area × Delay ($\mu m^2 \times ns$)
Scheme-1 [8]	(−8, 8)	24	24	0.0220	83559.17	8374	1293	1.080×10^8
LUT [13]	(−8, 8)	9	9	0.0180	17864.24	1791	2.45	4.377×10^4
RALUT [13]	(−8, 8)	9	9	0.0178	11871.53	1190	2.12	2.517×10^4
Hybrid [15]	(−8, 8)	9	9	0.0189	5130.78	515	2.80	1.437×10^4
Optimized LUT [16]	(−8, 8)	9	9	0.0205	1603.32	161	2.82	4.521×10^3
Proposed	(−8, 8)	9	6	0.0196	1280.66	129	2.12	2.714×10^3

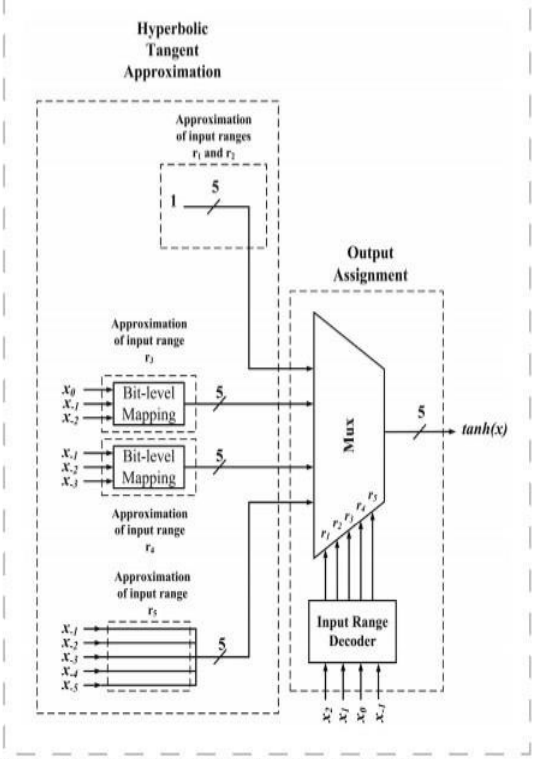

Fig. 5. Hardware implementation of the considered example.

In the architecture proposed in [15] a simple PWL approximation in combination with RALUT is used. The RALUT stores the difference between PWL approximation and the hyperbolic tangent function, which results in a reduction in number of stored points compared to the RALUT used in [13]. This reduction lowers the area required while because of the subtraction present in the proposed architecture, it has more delay.

Meher [16] has proposed an optimized LUT-based architecture. The proposed architecture is based on linear approximation in combination with LUT, and requires 7 and 15 stored points for errors of $\epsilon = 0.04$ and $\epsilon = 0.02$. The reduction in the number of stored points reduces the area required for hardware implementation.

Our proposed structure is based on a linear approximation in combination with bit-level mapping, which removes the need to store points and can be implemented using a purely combinational circuit. Also, the number of output bits required in the proposed structure is lower than all previously proposed architectures, which reduces the area consumption.

On the other hand, the simple input range decoding method, which only uses the position of the first occurrence of one in binary input in combination with bit-level mapping provides a high speed structure. This is confirmed by synthesis results shown in Tables V and VI, which show that the proposed structure in both cases compares favorably to the previously proposed structures in terms of area, delay, and area×delay.

VII. NEURAL NETWORK IMPLEMENTATION USING THE PROPOSED STRUCTURE FOR HYPERBOLIC TANGENT ACTIVATION FUNCTION

The general configuration of a Madaline is shown in Fig. 6. The Madaline has $L+1$ layers with N_0 inputs, and N_i Adalines in each layer i. Multiplication and addition are the arithmetic operations required in neural network implementation.

The output of the activation function should be multiplied by weights. Therefore, the multiplier size, S_m, in hidden layers

IJIRAE::International Journal of Innovative Research in Advanced Engineering ISSN:2349-2163
Volume 13, Issue 03, March 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1303.34

TABLE VII
COMPARISON OF NETWORK IMPLEMENTATION

Activation Function	Maximum Error	Area (μm^2)	Gate Count	Delay (ns)	Power (mw)
Proposed	0.04	21 284.94	12 154	7.72	8.98
Optimized LUT [16]	0.04	56 957.42	15 729	11.22	11.21
Proposed	0.02	44 451.59	14 476	8.68	9.34
Optimized LUT [16]	0.02	81 124.80	18 151	11.81	11.93

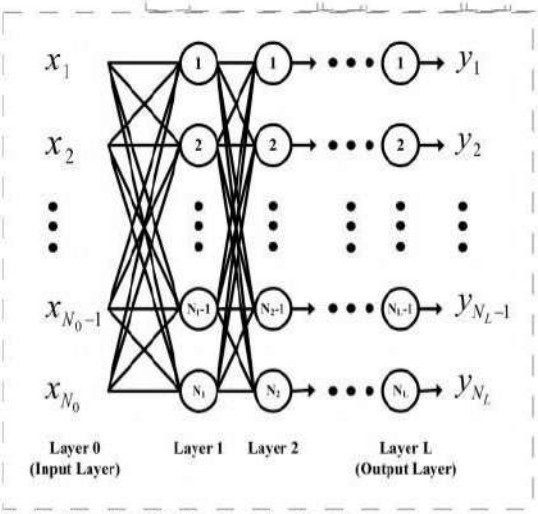

Fig. 6. Three layer Madaline general configuration.

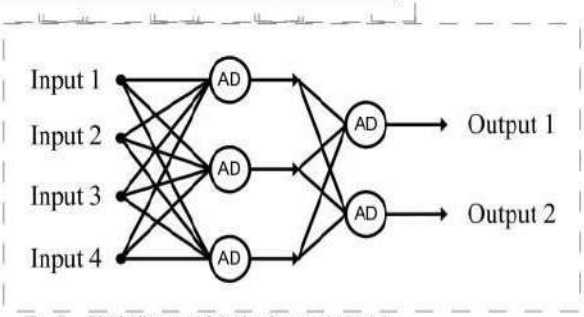

Fig. 7. Block diagram of the implemented network.

Template	Direction	Equivalent Bits (Input)	Class (Output)
	Horizontal	1100	01
	Vertical	0101	00
	Horizontal	0011	01
	Vertical	1010	00
	Left Diagonal	1001	10
	Right Diagonal	0110	11

Fig. 8. Optical input patterns and their related class.

of neural network can be written in the following form:

$$S_m = N_w \times N_{\text{out}} \tag{32}$$

where N_w is the number of bits used for synaptic weight storage while N_{out} is the number of output bits of the activation function.

The multiplication results of synapses connected to an Adaline in the next layer should be added before passing through the activation function of that layer. Considering (32), the number of output bits of multipliers is equal to $N_w + N_{\text{out}}$. Therefore, the size of adders, S_a, required for addition of multiplication results of N_i synapses between layer i and $i+1$ can be written in the following form:

$$S_a = (N_w + N_{\text{out}}) \times N_i. \tag{33}$$

Therefore, the size of multipliers and adders in the hidden layers of neural network depends on the number of output bits of the activation function.

For an specific maximum allowable error, the proposed structure requires less number of output bits compared to the previously developed architectures. Therefore, bit width of multipliers and adders in the hidden layers of the network using proposed structure as its activation function is lower. Multipliers and adders with lower bit width have lower area, delay, and power consumption. Therefore, using proposed structure results in efficient VLSI implementation of neural networks with hyperbolic tangent activation function.

To evaluate the efficiency of proposed structure, it is used to implement a 4-3-2 network for an optical template matching application. The general neural network block diagram is shown in Fig. 7. It is capable of recognizing six different input patterns and classifying them as four different classes. The optical input patterns and their related class are shown in Fig. 8.

The network is coded using Verilog hardware description language and synthesized by Synopsys Design Compiler using TSMC 0.18-μm library. The signal processing in the implemented network is based on fixed point arithmetic and the

proposed structure implemented in the previous section, which had maximum allowable errors of $\epsilon = 0.02$ and $\epsilon = 0.04$ is used as activation function. The network training is done off-chip and the calculated weights are stored on the registers inside the chip.

The same network is also coded and synthesized using the structure proposed by Meher [16] with maximum allowable errors of $\epsilon = 0.02$ and $\epsilon = 0.04$.

Post-layout simulation results show that the implemented network using the proposed structure and the one proposed by Meher [16] for both cases of $\epsilon = 0.02$ and $\epsilon = 0.04$ performs well.

The post-layout simulation results of the implemented network are summarized in Table VII, which show that the implementation of the network using proposed structure results in efficient VLSI implementation in terms of area, delay, and power for both maximum allowable errors of $\epsilon = 0.02$ and $\epsilon = 0.04$.

VIII. Conclusion

A new approximation scheme for hyperbolic tangent was proposed in this paper. The proposed approximation scheme is based on a mathematical analysis considering maximum allowable error as a design parameter.

Based on the proposed approximation scheme, a hybrid architecture for hardware implementation of hyperbolic tangent activation function was presented. The synthesis results showed that the proposed structure compares favorably to the previously developed architectures in terms of area, delay, and area×delay.

The proposed structure required less output bits for the same maximum allowable error compared to the previously developed architectures. Reduction in number of activation function output bits resulted in multipliers and adders with lower bit width which in turn reduces the area, power, and delay in VLSI implementation of neural networks. The proposed structure is used for implementing a 4-3-2 network, which is capable of recognizing six different input patterns. Post-layout simulation results showed that the proposed structure results in an efficient neural network VLSI implementation in terms of area, delay, and power.

References

[1] V. Koosh and R. Goodman, "Analog VLSI neural network with digital perturbative learning," *IEEE Trans. Circuits Syst. II, Exp. Briefs* vol. 49, no. 5, pp. 359–368, May 2002.

[2] D. Kim, H. Kim, H. Kim, G. Han, and D. Chung, "A SIMD neural network processor for image processing," *Advan. Neural Netw.*, vol. 3497, pp. 665–672, Jun. 2005.

[3] D. Maliuk, H.-G. Stratigopoulos, and Y. Makris, "An analog VLSI multilayer perceptron and its application toward built-in self-test in analog circuits," in *Proc. IEEE 16th Int. Online Test. Symp. Conf.*, Jul. 2010, pp. 71–76.

[4] L. Gatet, H. Tap-Beteille, and M. Lescure, "Real-time surface discrimination using an analog neural network implemented in a phase-shift laser rangefinder," *IEEE Sensors J.*, vol. 7, no. 10, pp. 1381–1387, Oct. 2007.

[5] G. Zatorre, N. Medrano, M. Sanz, B. Calvo, P. Martinez, and S. Celma, "Designing adaptive conditioning electronics for smart sensing," *IEEE Sensors J.*, vol. 10, no. 4, pp. 831–838, Apr. 2010.

[6] R. Jimenez, M. Sanchez-Raya, J. Gomez-Galan, J. Flores, J. Duenas, and I. Martel, "Implementation of a neural network for digital pulse shape analysis on a FPGA for on-line identification of heavy ions," *Nucl. Instr. Meth. Phys. Res. A*, vol. 674, pp. 99–104, May 2012.

[7] A. Armato, L. Fanucci, E. Scilingo, and D. D. Rossi, "Low-error digital hardware implementation of artificial neuron activation functions and their derivative," *Microprocess. Microsyst.*, vol. 35, no. 6, pp. 557–567, 2011.

[8] C. W. Lin and J. S. Wang, "A digital circuit design of hyperbolic tangent sigmoid function for neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. Conf.*, May 2008, pp. 856–859.

[9] S. Vassiliadis, M. Zhang, and J. Delgado-Frias, "Elementary function generators for neural-network emulators," *IEEE Trans. Neural Netw.*, vol. 11, no. 6, pp. 1438–1449, Nov. 2000.

[10] K. Basterretxea, J. Tarela, and I. D. Campo, "Digital design of sigmoid approximator for artificial neural networks," *Electron. Lett.*, vol. 38, no. 1, pp. 35–37, Jan. 2002.

[11] K. Basterretxea, J. Tarela, and I. D. Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," *IEEE Syst. Devices*, vol. 151, no. 1, pp. 18–24, Feb. 2004.

[12] M. Zhang, S. Vassiliadis, and J. Delgado-Frias, "Sigmoid generators for neural computing using piecewise approximations," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 1045–1049, Sep. 1996.

[13] K. Leboeuf, A. Namin, R. Muscedere, H. Wu, and M. Ahmadi, "High speed VLSI implementation of the hyperbolic tangent sigmoid function," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.* Nov. 2008, pp. 1070–1073.

[14] M. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," *IEEE Comput. Digital Tech.*, vol. 150, no. 6, pp. 403–411, Nov. 2003.

[15] A. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient hardware implementation of the hyperbolic tangent sigmoid function," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2009, pp. 2117–2120.

[16] P. Meher, "An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks," in *Proc. 18th IEEE/IFIP VLSI Syst. Chip Conf.*, Sep. 2010, pp. 91–95.

[17] K. Basterretxea, J. Tarela, I. D. Campo, and G. Bosque, "An experimental study on nonlinear function computation for neural/fuzzy hardware design," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 266–283, Jan. 2007.

[18] K. Basterretxea, "Recursive sigmoidal neurons for adaptive accuracy neural network implementations," in *Proc. Adapt. Hardw. Syst. NASA/ESA Conf.*, Jun. 2012, pp. 152–158.

[19] R. Muscedere, V. Dimitrov, G. Jullien, and W. Miller, "Efficient techniques for binary-to-multidigit multidimensional logarithmic number system conversion using range-addressable look-up tables," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 257–271, Mar. 2005.