**Non-linear activation function approximation using a REMEZ algorithm**

**Samba Raju Chiluveru** | **Manoj Tripathy** | **Bibhudutta**

**Abstract**

Here a more accurate piecewise approximation (PWA) scheme for non-linear activation function is proposed. It utilizes a precision-controlled recursive algorithm to predict a sub-range; after that, the REMEZ algorithm is used to find the corresponding approximation function. The PWA realized in three ways: using first-order functions, that is, piecewise linear model, second-order functions (piecewise non-linear model), and hybrid-model (a mixture of first-order and second-order functions). The hybrid-order approximation employs the second-order derivative of non-linear activation function to decide the linear and non-linear sub-regions, correspondingly the first-order and second-order functions are predicted, respectively. The accuracy is compared to the present state-of-the-art approximation schemes. The multi-layer perceptron model is designed to implement XOR-gate, and it uses an approximate activation function. The hardware utilization is measured using the TSMC 0.18-μm library with the Synopsys Design Compiler. Result reveals that the proposed approximation scheme efficiently approximates the non-linear activation functions.

1 | INTRODUCTION

Neural networks (NNs) have a wide range of applications such as pattern recognition, speech recognition, testing of analog circuits, smart sensing, and forecasting, etc. Consequently, the NN hardware implementation has been attentive since the last decade [1] and can be implemented as either analog or digital systems. The analog implementation, however, has disadvantages such as no programmability, less computational accuracy, and thermal drift [2]. These drawbacks can be overcome by digital hardware implementation; subsequently, it often needs optimally built blocks.

The principal building blocks in NN are multiplier, adder, and non-linear activation functions. Among these, the most complex building block is the non-linear activation function, and its efficient approximation plays a crucial role in neural network performance. The conventional activation functions include threshold, Rectified Linear Unit (ReLU), leaky ReLU, sigmoid, and tanh, functions etc. The neural network generally uses a gradient descent algorithm for training in that it uses the first-order derivative of the activation function. Therefore, in hardware implementation, the derivative approximation is also needed. Additionally, research conducted in [3,4] shows that approximation of non-linear activation functions with higher accuracy increases the learning and generalization capabilities of neural networks. Therefore, non-linear activations and their derivatives with greater accuracy are required.

The multiplier and adder are well implemented in digital design. The non-linear activation functions such as sigmoid and tanh, are exponential functions. Thus, hardware implementation becomes a challenging task [5]. Therefore, anyone of the approximation schemes is utilized to approximate the non-linear function viz. truncated series expansion [6], look-up tables (LUTs) [7], and piecewise approximation (PWA) [3,8–10]. In truncated series expansion, the non-linear activation functions are approximated to the truncated Taylor series [6]. Moreover, higher precision requires more number of Taylor series terms, which means more hardware is required that ultimately increases total design cost.

In LUT-based approximation, the non-linear function is divided into equal parts, and each part is assigned an approximation value. However, for higher accuracy, this approximation requires adequate memory storage. Additionally, another drawback is that such estimated values are manually designed. The PWA scheme is indeed a good alternative in hardware implementation that saves silicon area and shows short-latency...
time \([9,11–13]\). From the literature, it is observed that first-order function piecewise linear (PWL) approximation implementation requires fewer hardware resources than higher order. But, it is also noted that a relatively large number of first-order functions are needed for a non-linear approximation that results in more storage requirements. While the second-order approximation requires more multiplier-related hardware resources than the first-order, it requires less coefficient memory than the first-order. Thus, a trade-off must be made between the storage requirements and the complexity of the hardware. Therefore, a hybrid-order model is also introduced, which uses a combination of first-order and second-order functions for approximation.

Recently in \([10]\), we have proposed an iterative algorithm for PWL approximation that approximates the sigmoid activation function. It was found that the accuracy of approximate sigmoid has greatly improved, and it also shows the hardware complexity optimization in terms of the number of PWL functions required. Nonetheless, the algorithm uses a fixed step size of \(\Delta\) value, and it approximates only the sigmoid activation function.

Here we extend the iterative algorithm with an adaptive value of \(\Delta\) derived from a mirror image of the derivative function concerning the \(x\)-axis. The resulting approximation now has an improved convergence rate, and decreased approximation function number for less maximum error \((\varepsilon < 0.001)\). The algorithm uses an acceptable error \((\varepsilon)\) as an input parameter. Additionally, the algorithm now used for PWA and approximation of any non-linear activation function. Furthermore, the proposed algorithm is used to approximate sigmoid and tanh and their first-order derivatives. They can be approximated in three ways: first-order, second-order, and hybrid-order. In the hybrid-order model, we use a mixture of first and second-order functions. This distinction is achieved with a double derivative approach.

The iterative algorithm finds a proper sub-region for the given maximum error; then, the proposed algorithm uses the REMEZ optimization method to approximate the sub-region with a polynomial function \([14]\). Afterwards, the benchmark multi-layer perceptron (MLP) system is developed that implements the XOR-gate functionality \([15]\). The MLP architecture is trained separately, and its weights are converted to a fixed-point data type. In the evaluation, the MLP uses approximate sigmoid and tanh activation functions developed by the state-of-the-art and proposed algorithms. The hardware complexity is observed in two ways: the number of approximation functions required by a non-linear function and the hardware implementation results. The proposed PWA approximation is compared to the state-of-the-art approximation schemes. The Verilog HDL is used to code approximation activation function and MLP, afterwards, synthesis results are observed with Synopsys Design Compiler with TSMC 0.18-\(\mu\)m library.

The rest of the paper is organised as follows. The proposed algorithm is described in Section 2. In Section 3, the accuracy of approximate functions is described. Section 4 discusses the hardware implementation of the approximated activation function and XOR-gate. The conclusion is finally stated in Section 5.

\section{PROPOSED PWA ALGORITHM}

The proposed algorithm is developed for the PWA scheme, and now it can approximate any non-linear activation function \(g(x)\). It is designed as a recursive algorithm that works primarily for a targeted maximum error of \(\varepsilon\). The equal error distribution criteria have been met throughout the algorithm to find the broadest possible range. Afterwards, each range uses the REMEZ algorithm to find the PWA function for the corresponding non-linear activation function.

While finding a range, the algorithm uses an adaptive step size, \(\Delta\). It is calculated with a mirror image of the rate of change of non-linear activation function value, concerning the \(x\)-axis. Further, in the iterative process, \(\Delta\) value is multiplied by a factor of \((\text{length}(\text{Rec} \_der)/10)\), and this multiplication factor is such that it makes \(\Delta\) value to decrease gradually as the algorithm approaches the approximate function \(P_n(x)\). Therefore the approximation becomes more accurate. In addition, for higher accuracy (less maximum error \((\varepsilon < 0.001)\)), \(\Delta\) uses a user-defined typical value of \(\text{delta}_\text{lim}\) so that the algorithm can converge effectively. Otherwise, it will take a long time to converge. However, the approximation accuracy with fixed delta value and adaptive delta value is same. Therefore, a \(\text{delta}_\text{lim}\) is a minimum limit value, if the value of \(\Delta\) decreases further, its value is defined as \(\text{delta}_\text{lim}\), which makes the algorithm converge quicker to the estimated range.

\begin{equation}
 f(x) = \begin{cases} 
 \text{Linear} & \frac{d^2f(x)}{dx^2} \leq \delta \\
 \text{Non-linear} & \frac{d^2f(x)}{dx^2} > \delta 
 \end{cases}
\end{equation}

In the first-order-based approximation (PWL), the proposed algorithm uses a first-order polynomial function \(P_n(x) = a + bx\) for approximation, the \(a\) and \(b\) values are found by the REMEZ algorithm. The iterative algorithm divides the input function into sub-regions for a given maximum error of \(\varepsilon\); after that, the REMEZ algorithm determines the PWL function for each sub-region. Similarly, for a second-order based approximation, the algorithm uses a second-order polynomial function \(P_n(x) = a + bx + cx^2\), and the REMEZ algorithm estimates these coefficient values efficiently.

\begin{align}
 S(x) &= \frac{1}{1 + e^{-2x}} \\
 S'(x) &= S(x)(1 - S(x)) \\
 T(x) &= \frac{1 - e^{-2x}}{1 + e^{-2x}} \\
 T'(x) &= 1 - T^2(x)
\end{align}
In hybrid-order approximation, the proposed algorithm uses first-order and second-order polynomials. A second-order derivative is used to decide the degree of non-linearity; Equation (1) shows the required partitioning in mathematical form. The value of $\delta$ is a deciding factor that gives a proper partition, and it is determined by careful testing, and the final value is set to $\delta = 0.05$. After partitioning, the algorithm uses those regions to approximate the respective polynomial functions.

The REMEZ algorithm is a type of min-max algorithm that processes iteratively and operates on the Chebyshev min-max polynomials ($P_n(x)$) [16]. The REMEZ algorithm is designed to converge to a minimum of maximum error. It uses absolute error for convergence, and the algorithm stops if the maximum error is equal to the fraction of the minimum error ($E_M = \alpha E_n$). Otherwise, the algorithm searches for another pair of Chebyshev polynomials ($P_n(x)$).

Here sigmoid, tanh, and their first-order derivatives are approximated. The sigmoid function and its derivative defined in the input domain ($-8$, $8$), whereas tanh function and its derivative defined in the domain ($-4$, $4$). Beyond this range, these activation functions approximately saturate to a constant value. Moreover, the sigmoid and tanh, activation functions have a range of 0 to 1 and $-1$ to 1, respectively. Mathematical representations of sigmoid and its derivative are shown in Equations (2) and (3) respectively. Similarly, Equations (4) and (5) represents tanh and its derivative functions. Both sigmoid and tanh functions have got inherent symmetric properties. Hence only a half part is sufficient to represent the entire function. Therefore, the negative $x$ range of the sigmoid function can be expressed as $S(-x) = 1 - S(x)$, similarly tanh function can be expressed as $T(-x) = -T(x)$. The derivatives of both functions have even symmetry, consequently following the expressions $S'(x) = S'(-x)$ and $T'(x) = T'(-x)$. Therefore, we have used only a positive $x$ range for the approximation of the entire function.

The proposed algorithm is shown in Table 1, it is applied to a function $g(x)$, where $g(x)$ is any non-linear activation function or part of it. The recursive process starts by defining its range $x_{\min}$ and $x_{\max}$. Initially, $x_{\min} = Temp$ and $x_{\max} = x_{\text{lim}}$ are considered. Where $x_{\text{lim}}$ is 8 for sigmoid and 4 for tanh function, and $Temp$ is initialized with $x_{\text{lower}}$, after approximating one range, it is then assigned the next value as per the algorithm. After finding a sub-range, the algorithm uses the REMEZ method to obtain an optimized approximating function. Afterwards, the maximum error of the approximated function is compared with the acceptable error of $\epsilon$. If its value is less than or equal to $\epsilon$, then the corresponding function $P_n(x)$ is used to approximate that sub-range. Otherwise, $x_{\max}$ is decremented by $\Delta$ (i.e., $x_{\max} = x_{\max} - \Delta$). The algorithm cycle is repeated to get the final approximated polynomial ($P_n(x)$). This procedure is continued until the desired maximum error is reached while returning the first approximation polynomial function for first sub-region ($x_{\min}$, $x_{\max}$).

For the next sub-region, $x_{\min}$ is assigned with the last $x_{\max}$ value (i.e., $x_{\min} = x_{\max}$), and $x_{\max}$ value is assigned with $x_{\text{lim}}$ (i.e., $x_{\max} = x_{\text{lim}}$). For this range, the above-mentioned procedure is followed again. Likewise, the algorithm proceeds the entire range (i.e., $x_{\text{lower}}$ to $x_{\text{lim}}$).

### TABLE 1  Recursive algorithm to find approximate functions for a non-linear activation function ($g(x)$)

| Procedure: Automated technique |
|--------------------------------|
| Initialize: Temp = $x_{\text{lower}}$, delta_limit = 0.00001 |
| 1: $x_{\min} = Temp$, $x_{\max} = 8$
| 2: $f(x_i) = g(x_i)$
| \[ Der._g = g'(x_i) \]
| \[ x_{\min} < x_i < x_{\max} \]
| 3: Rev._der = max(Der._g) - Der._g |
| 4: $y = P_n(x)$ |
| 5: $g(x_i) - P_n(x_i) = (-1)^iE_i, i = 0, 1, 2$ |
| 6: $E_i = \max|g(x_i) - P_n(x_i)|$
| 7: $E_M = \max|E_i|, E_m = \min|E_i|$ |
| 8: $k = \max|g(x_i') - P_n(x_i')|$ |
| 9: if $E_M = aE_n$ then |
| 10: if $x_{\max} > \epsilon$ then |
| 11: if $x_{\max} = 8$ then |
| 12: Stop algorithm, $y = P_n(x)$ # |
| 13: else |
| 14: Temp = $x_{\max}$ |
| 15: $y = P_n(x)$ # Approximation function; |
| 16: go to 2 |
| 17: end if |
| 18: else |
| 19: if $\Delta > = delta_limit$ then |
| 20: \[ \Delta = Rev._der(\text{length}(Rev._der))^\frac{1}{p} \]
| 21: else |
| 22: $\Delta = delta_limit$ |
| 23: end if |
| 24: $x_{\max} = x_{\max} - \Delta$ |
| 25: go to 4 |
| 26: end if |
| 27: else |
| 28: go to 5 |
| 29: end if |

3 | RESULTS AND DISCUSSION

The proposed approximation scheme is applicable to any non-linear function. In this article, the sigmoid, tanh, and their first-order derivatives are approximated. These activation functions have symmetry, so the analysis is performed for only
a positive part, and a total of $10^6$ equidistance points are used. The approximation function accuracy is measured in terms of absolute average error of $E_{avg}$ shown in Equation (6), and absolute maximum error of $E_{max}$ shown in Equation (7). Where $f(x_i)$ is the sample value of the original function, and $y_i$ is the approximated function value.

$$E_{avg} = \frac{\sum_{i=0}^{10^6} |f(x_i) - y_i|}{10^6}$$

$$E_{max} = \max |f(x_i) - y_i|$$

For a hardware implementation, each of these PWA approximations requires adder, multiplier, and memory for coefficient storage. Therefore, reducing the approximation functions needed for a non-linear activation function, decreases the hardware requirements, leading to an optimized design. Thus, the hardware complexity is compared with the number of approximation functions needed by sigmoid and tanh activation functions for a given maximum error. Additionally, hardware implementation results for sigmoid and tanh activation functions are observed and compared to the state-of-the-art approximation. Subsequently, these approximation functions are used to implement an MLP, which implements XOR-gate. The Verilog HDL code is used to implement approximation functions and MLP, and the synthesis is performed with Synopsys Design Compiler using CMOS-0.18 μm library. Now the hardware complexity is measured with the area, delay, and power parameter, respectively.

The sigmoid, tanh, and their derivatives are approximated with the proposed algorithm with $\epsilon = 5 \times 10^{-4}$, and approximation is performed using three PWA schemes. Table 2 depicts the number of approximation functions required for each function concerning the PWA style.

### 3.1 Sigmoid function

The sigmoid activation function is approximated with the proposed three PWA schemes for a maximum error of $\epsilon = 5 \times 10^{-4}$. Figure 1 shows the convergence of the proposed algorithm and [10], for the PWL-based sigmoid activation function approximation. The proposed algorithm generates the first sub-region $[0, 0.5051]$ for 316 iterations, while [10] generates the first sub-region $[0, 0.5000]$ for 7497 iterations. From this result, it is observed that the proposed algorithm requires fewer iterations compared to [10] and finds a wider range, resulting in fewer PWL function requirements for the low maximum error of $\epsilon$. The experiments are conducted to decide the maximum value of $\epsilon$, where the proposed algorithm can differentiate [10] in terms of the number of PWL functions requirement. It is found that the $\epsilon < 0.001$ approximately shows the difference between the proposed method and [10]. Therefore the maximum error ($\epsilon$) less than 0.001 gives a reduced number of PWL functions for sigmoid approximation. However, it is also verified for the tanh activation function.

For $\epsilon = 5 \times 10^{-4}$, the number of approximate functions obtained by each PWA scheme is shown in Table 2. The first-order approximation divides sigmoid function into 13 sub-regions, and each requires a PWL function, so it uses 13 PWL functions. The second-order approximation demands five sub-regions for the sigmoid activation function, resulting in five second-order functions. In the hybrid-order approximation, the sigmoid function uses a double derivative for the

| Function               | First order | Second order | Hybrid First order | Second order |
|------------------------|-------------|--------------|--------------------|--------------|
| Sigmoid                | 13          | 5            | 7                  | 1            |
| Derivative of sigmoid  | 11          | 5            | 8                  | 1            |
| Tanh                   | 19          | 6            | 3                  | 4            |
| Derivative of tanh     | 23          | 7            | 4                  | 5            |

**Figure 1** Convergence of PWL-based recursive algorithm for sigmoid activation function. PWL, piecewise linear
non-linearity detection with a threshold value of $\theta = 0.05$. Figure 2 represents the sectionalization of the sigmoid function into linear and non-linear regions. Correspondingly, the resulting sigmoid function has a linear region of $[0–0.45]$ and $[2.7–8]$, and a non-linear region of $[0.45–2.7]$. Afterwards, the linear region requires seven first-order functions, while the non-linear region requires a second-order function.

The performance of sigmoid approximation is measured with absolute average and maximum error. The approximated sigmoid function accuracy is compared with the state-of-the-art approximation schemes. The comparative results are shown in Table 3. From this result, it can be observed that the proposed PWA schemes give the lowest average and maximum errors as compared with the literature. Therefore, the proposed PWA schemes show better accuracy than other approximation schemes. The hybrid approximation shows more precision improvement than the approximation of the first and second orders. Furthermore, it shows less average error. Figure 3 shows the sigmoid approximation function for three PWA schemes with the absolute maximum error of $\epsilon$. From this figure, it can be easily observed the maximum error equal distribution.

3.2 | Tanh function

The proposed three approximation schemes are used to approximate the tanh activation function, for a maximum error of $\epsilon = 5 \times 10^{-4}$. The tanh function is more non-linear than the sigmoid function; therefore, it requires more sub-regions for approximation. Table 2 shows the number of approximation functions required by the tanh function. It needs 19 sub-regions for first-order approximation, resulting in 19 first-order PWL functions. The second-order approximation requires six sub-regions; thus, it requires six second-order functions. The hybrid approximation scheme uses the second-order derivative of the tanh function for partition. Figure 4 shows the separation of the tanh function into linear and non-linear regions. The linear region extends over $[0–0.05]$ and $[2.5–4]$, likewise the non-linear region extends within $[0.05–2.5]$. Thus, the hybrid approximation requires three first-order functions and four-second order functions, respectively.

Table 4 shows the absolute average and maximum error of the approximated tanh function, and previously published study. From this result, it is observed that the proposed PWA approximation schemes show less average and maximum error compared with the literature. Figure 5 shows the three approximation functions, the original tanh function, and the maximum error. In addition, the hybrid approximation in the linear region shows less cumulative average error than the non-linear region. From this figure, an equal distribution of maximum error can be observed.

3.3 | Derivatives

The first-order derivatives of sigmoid and tanh functions can be approximated in two ways, such as using original functions (mathematically derived) and using the proposed approximation schemes. From original functions, the derivatives are calculated by Equations (3) and (5), respectively. However, in this case, both original functions and their derivatives need the same approximation functions. In the proposed PWA scheme, the maximum error of $\epsilon = 5 \times 10^{-4}$ is used, and the sigmoid and tanh derivatives require 11 and 23 PWL functions for the first-order implementation. The second-order approximation requires five and seven second-order functions, respectively.

For the hybrid-order approximation, a threshold value of $\theta = 0.05$ is used, for derivative of sigmoid and tanh, the segmentation is shown in Figures 6 and 7, respectively. Sigmoid

---

**FIGURE 2** Segmentation of sigmoid function to linear and non-linear parts

**TABLE 3** Accuracy of approximate sigmoid function in terms of absolute average and maximum error

| Method     | Input range | Avg. error | Max. error |
|------------|-------------|------------|------------|
| Proposed-first | $[-8,8]$    | 0.024      | 0.049      |
| Proposed-second  | $[-8,8]$    | 0.029      | 0.049      |
| Proposed-hybrid    | $[-8,8]$    | 0.027      | 0.049      |
| Recursive [10]     | $[-8,8]$    | 0.027      | 0.049      |
| Tommiska [2]       | $[-8,8]$    | 0.17       | 0.39       |
| Myers [5]          | $[-8,8]$    | 2.47       | 4.90       |
| Bajger [11]        | $[-8,8]$    | 0.1199     | 0.8616     |
| Area method [9]    | $[-8,8]$    | 0.1070     | 0.7880     |
| Tsai [17]          | $[-8,8]$    | 2.8        | 11.8       |
| Faiedh [18]        | $[-5,5]$    | 0.20       | 1.11       |
| Alippi [19]        | $[-8,8]$    | 0.87       | 1.89       |
| PLAN [20]          | $[-8,8]$    | 0.59       | 1.89       |
| CRI $q = 3$ [21]   | $[-8,8]$    | 0.85       | 2.06       |
| Zhang [22]         | $[-4,4]$    | 0.77       | 2.16       |
derivative requires eight first-order and one second-order function, while the \( \tanh \) derivative requires four first-order and five second-order functions. Table 5 shows the accuracy results of sigmoid and \( \tanh \) function derivatives, and these results reflect the accuracy of mathematically derived derivatives and the proposed approximation based derivatives, respectively.

### 3.4 Hardware complexity in terms of number of approximate functions

The reduced number of approximation functions required by an activation function results in a lower memory requirement. As a result, for a non-linear activation function, hardware complexity is directly proportional to the number of approximate functions required. So the hardware complexity is measured by the number of approximation functions needed by the approximation method.

Most of the state-of-the-art defined for PWL approximation; therefore, the number of first-order functions required by an activation function is used for comparison. The PWL functions needed by the sigmoid activation function are shown in Table 6, for a maximum error of \( \epsilon = 5 \times 10^{-4} \). However, the state-of-the-art is defined for different maximum errors; its accuracy is limited and can not be extended. Therefore, the proposed algorithm used these maximum errors to find corresponding approximate PWL functions. The required number of approximation functions are tabulated in Table 6. The proposed approximation uses only 26 first-order functions to approximate sigmoid function for an average error of \( 2.4 \times 10^{-4} \). However, no present method gives an average error less than or equal to the proposed method. Hence, we confirmed that the proposed PWL approximation is more accurate among the presented PWL methods. Subsequently, the number of PWL functions required by the proposed scheme is less than the previously published literature.

Furthermore, Zhang et al. [22] use 16 intervals for sigmoid approximation with an average error of \( 5.7 \times 10^{-4} \), and each interval approximated with a second-order function, whereas, the proposed second-order approximation requires five functions for \( 2.6 \times 10^{-4} \). Therefore, it can be concluded that the proposed algorithm is an optimized approximation for the sigmoid activation function.
The tanh activation function is approximated using a polynomial approximation scheme [23,25] and memory-based approach [12,24]. The tanh function approximated with the PWL method requires 16 approximation functions for a maximum error of $1.9 \times 10^{-2}$ [25]. At the same time, the proposed method requires 32 first-order functions for a maximum error of $5 \times 10^{-4}$. However, no method in literature gives equivalent accuracy as proposed approximation. Furthermore, the presented research is defined for different maximum errors; therefore, the proposed algorithm uses the same error for better comparison. The number of PWL functions requirement is shown in Table 7, from these results, it can be decided that the proposed approximation scheme yields fewer PWL functions compared with literature, therefore requiring less hardware.

In memory-based approximation, only approximate values are needed for storage, and these values are used with a Look-up Table. However, it is a manual process and requires large memory for high accuracy. Additionally, in [23], two second-order approximation schemes are used for a maximum error of $4.3 \times 10^{-2}$. However, it is not extendable to any other error.
4 | HARDWARE VERIFICATION OF XOR

For XOR-gate implementation, the approximate sigmoid and tanh activation functions are used with a multi-layer perceptron model (MLP). The hardware complexity of approximated sigmoid and tanh is compared with the area-based method [9] and recursive method [10]. Moreover, the proposed three approximation schemes (first, second, and hybrid-order) are used in MLP implementation. The floating-point data type hardware implementation requires extensive hardware circuitry, so a fixed-point data type is used for hardware realization.

The input data width plays a crucial role in hardware implementation. Here more priority is given to the accuracy of the non-linear activation function. Therefore, the data width is selected such that the resulting data-type conversion error should be minimum. From [10], it is observed that the 8-bit data width shows a significant deviation in accuracy, while the 16-bit data width representation shows almost the same accuracy. Hence, the hardware implementation uses a 16-bit fixed-point data type. These results can also be found for second and hybrid-order implementations, and they are approximately shown the same results. Therefore, the MLP weights, XOR-gate input and output values use a 16-bit data width.

In the PWA scheme, the first-order, second-order, and hybrid-order approximations are realized using an adder, multiplier, and storage memory for coefficients. Figure 8 shows the block diagram representation of these three approximations. The PWL function requires one multiplier, one adder,

| Method        | Avg. error | Max. error | No. of PWL functions required by original function | Proposed method |
|---------------|------------|------------|-------------------------------------------------|-----------------|
|               |            |            |                                                 |                 |
| Basterretxea  | 0.84       | 1.97       | 33                                              |                 |
| Myers         | 2.47       | 4.90       | 7                                               |                 |
| Area method   | 0.1070     | 0.7880     | 16                                              |                 |
| Baiger        | 0.1199     | 0.8616     | 16                                              |                 |
| Faiedh        | 0.20       | 1.11       | 10                                              |                 |
| PLAN          | 0.59       | 1.89       | 8                                               |                 |
| CRI q = 3     | 0.85       | 2.06       | 24                                              |                 |
| Recursive     | 0.027      | 0.049      | 28                                              |                 |
| Proposed method | 0.024   | 0.049      | 26                                              |                 |

| Method        | Avg. error | Max. error | No. of PWL Functions required by original function | Proposed method |
|---------------|------------|------------|-------------------------------------------------|-----------------|
|               |            |            |                                                 |                 |
| Area method   | 0.1100     | 1.58       | 18                                              |                 |
| Vassiliadis   | 0.50       | 5.7        | 24                                              |                 |
| Recursive     | 0.0175     | 0.049      | 34                                              |                 |
| Proposed method | 0.0165 | 0.049      | 32                                              |                 |

**TABLE 6** Hardware complexity of approximate sigmoid function in terms of number of PWL functions

**TABLE 7** Hardware complexity of approximate tanh function in terms of number of PWL functions

F I G U R E 8  Hardware realization of proposed approximation schemes
and storage memory for coefficients like RAM/ROM. At the same time, the second-order function requires three multipliers, two adders, and storage memory for coefficients. In hybrid-order approximation, the first-order and second-order are decided with the second-order derivative. Therefore, it needs three multipliers, two adders, and storage memory. The second-order derivative controls the lower part of the block diagram and the adder circuit.

Figure 9 shows the MLP architecture where \( W_{11}, W_{12}, W_{21}, W_{22}, V_1, \) and \( V_2 \) are weights and \( b_0 \) and \( b_1 \) are bias values. It has a three-layer structure, that is, one input layer, hidden layer, and output layer, respectively. The input and hidden layer have two neurons, whereas the output layer has a single neuron. The XOR-gate functionality is implemented with MLP [26]. The Matlab code is used to implement MLP, and it is trained to XOR-gate functionality. The final weights are considered for hardware implementation. However, Matlab uses a floating-point number system, therefore, every weight is converted to a 16-bit fixed-point format.

The activation function of the trained MLP is replaced by approximate sigmoid and tanh activation functions. Afterwards, MLP uses the area-based and recursive-based activation functions to compare hardware complexity. The area-based method has a fixed maximum error, and it cannot be extendable. Furthermore, it already shows good accuracy and proved to be an optimized design compared with the other PWL approximation schemes [9]. Therefore here we used the area-based method’s maximum error for better hardware comparison. It has a maximum error value of \( 7.88 \times 10^{-3} \) for sigmoid function and \( 1.9 \times 10^{-2} \) for tanh function. Therefore, when developing sigmoid and tanh approximation functions using the proposed approximation scheme, these maximum error values are considered.

The proposed PWA scheme requires eight PWL approximation functions for both sigmoid and tanh functions. The second-order approximation requires two second-order functions, and the hybrid-order requires one second-order and one first-order function for both sigmoid and tanh functions, respectively. Furthermore, the second-order approximation use six registers. The hybrid-order approximation use five registers, and first-order approximation use 16 registers to store coefficient values.

The second-order and hybrid-order approximations use more multipliers and adders compared with first-order approximation. Therefore, they require more hardware for approximation. Table 8 shows the hardware utilization of approximation function and the corresponding MLP implementation. It displays the power, delay, and area required for approximation function and XOR-gate; the piece-wise approximation scheme shows good results among the three proposed methods. It requires a power of 5.02 mW and a delay of 5.571 ns for sigmoid implementation, while it uses a power of 4.98 mW and a delay of 3.264 ns for tanh implementation. The XOR-gate implementation needs 202.5 mW of power, and 22.833 ns of delay for sigmoid function, likewise, the tanh function requires 226.47 mW of power and 21.917 ns delay. Furthermore, it is observed that the proposed second-order and the hybrid-order approximations require more area and power compared with the PWL scheme. The sigmoid function uses an extra adder to find negative value range, while tanh requires a complement circuit, therefore, tanh comparatively has less delay. The second-order and hybrid-order models require the same area, but the power requirement and latency is less for hybrid-order.

The area-based method and recursive methods are PWL approaches, so for a fair comparison, the proposed PWL.

![Figure 9](image-url)  
**Figure 9** Multi-layer perceptron model for XOR-gate implementation

| Method        | Sigmoid | Tanh     | XOR with Sigmoid | XOR with Tanh |
|---------------|---------|----------|-----------------|---------------|
|               | Delay (ns) | Area (mm²) | Power (mW) | Delay (ns) | Area (mm²) | Power (mW) | Delay (ns) | Area (mm²) | Power (mW) |
| Area method   | 5.571   | 0.0218   | 7.16        | 3.264       | 0.0219   | 7.43       | 22.833     | 0.1882  | 218.9      | 21.917     | 0.1884     | 241.47     |
| [9]           |         |          |             |             |          |            |            |          |            |            |            |
| Recursive [10]| 5.571   | 0.0208   | 5.02        | 3.264       | 0.0209   | 4.98       | 22.833     | 0.185   | 202.5      | 21.917     | 0.185      | 226.47     |
| PW-linear     | 5.571   | 0.0208   | 5.02        | 3.264       | 0.0209   | 4.98       | 22.833     | 0.185   | 202.5      | 21.917     | 0.185      | 226.47     |
| PW-Nonlinear  | 8.749   | 0.0573   | 36.88       | 4.229       | 0.0573   | 36.66      | 28.498     | 0.2905  | 435.84     | 28.444     | 0.2905     | 429.62     |
| PW-hybrid     | 6.512   | 0.0573   | 35.99       | 4.210       | 0.0573   | 35.80      | 24.210     | 0.2906  | 430.56     | 24.104     | 0.2906     | 429.97     |
approximation is compared. However, the recursive model is developed for sigmoid function, and here we used it for tanh activation function approximation. For the sigmoid function, a maximum error value of $7.88 \times 10^{-3}$ is used, and the tanh function uses a maximum error of $1.9 \times 10^{-2}$. However, for these errors, the proposed algorithm and recursive algorithm requires the same number of approximate functions from the Table 8, it is observed that the PWL approaches require equal latency because they differ in the number of approximation functions only. The proposed PWL scheme shows a small difference in area requirement, but the power requirement shows a notified improvement. The XOR-gate hardware implementation values are shown separately; here, the approximation effect on neural network is observed clearly. Results demonstrated that the proposed PWL scheme requires less area and less power compared with the area-based method. However, the recursive method shows equal hardware complexity like the proposed PWL approximation. If accuracy increases ($\epsilon < 0.001$), it shows deviation from the proposed PWL method. For a sigmoid function with a maximum error of $\epsilon = 5 \times 10^{-4}$, the recursive process requires 28 PWL functions, while the proposed approach requires 26 PWL functions only. Likewise, the tanh function requires 34 PWL functions for recursive function and 32 PWL functions for the proposed PWL method.

5 | CONCLUSION

Here we have proposed PWA schemes to approximate a non-linear activation function and its first-order derivatives more accurately. The algorithm uses a recursive technique to find the correct sub-region and then uses the Remez algorithm to find the sub-regions optimized polynomial function. The non-linear activation function approximation is performed in three ways, that is, with first-order, second-order, and hybrid-order. A second-order derivative approach is used for the segmentation of non-linear functions into linear and non-linear parts. Afterwards, a PWL function is used for the linear part, and second-order function is used for the non-linear part. The XOR-gate is implemented with an MLP architecture, and it uses the approximated sigmoid and tanh activation. The accuracy comparison is performed with the present literature. The hardware comparison is performed in two ways, that is, using the number of approximation functions needed by non-linear function and hardware results in the application-specific integrated circuit platform. Results successfully proved the advantage of the proposed approximation scheme. Another advantage of the proposed algorithm is that it can approximate any non-linear activation function for every maximum error.

ORCID

Samba Raju Chiluveru https://orcid.org/0000-0002-1399-3297

REFERENCES

1. Mista, J., Saha, I.: Artificial neural networks in hardware: a survey of two decades of progress. Neurocomputing. 74(1-3), 239–255 (2010)
2. Tommiska, M.: Efficient digital implementation of the sigmoid function for reprogrammable logic. IEE Proc. Comput. Digit. Tech. 150(6), 403–411 (2003)
3. Basterretxea, K., et al.: An experimental study on nonlinear function computation for neural/fuzzy hardware design. IEEE Trans. Neural Netw. 18(1), 266–283 (2007)
4. Basterretxea, K.: Recursive sigmoidal neurons for adaptive accuracy neural network implementations. In: Proceedins of NASA/ESA conference on adaptive hardware and systems (AHS), pp. 152–158, Erlangen, 25–28 June. 2012
5. Myers, D., Hutchinson, R.: Efficient implementation of piecewise linear activation function for digital VLSI neural networks. Electron. Lett. 25, 1662 (1989)
6. Al.Nour, M., Abdel-Aty-Zohdy, H.S.: Implementation of programmable digital sigmoid function circuit for neuro-computing. In: Proceedins of Midwest symposium on circuits and systems, Cat. No. 98CB36268, pp. 571–574, Notre Dame, IN, 9–12 August. 1998
7. Piazza, F., Uncini, M., Zenobi, M.: Neural networks with digital LUT activation functions. In: Proceedins of 1993 international conference on neural networks, vol. 2, pp. 1401–1404, Nagoya, 25–29 October. 1993
8. Kim, Y., et al.: A 4.9 mw neural network task scheduler for congestion-minimised network-on-chip in multi-core systems. In: Proceedins of IEEE Asian solid-state circuits conference (A-SSCC), pp. 213–216, KaoHsiung, 10–12 November. 2014
9. Armato, A., et al.: Low-error digital hardware implementation of artificial neuron activation functions and their derivative. Microprocess. Microsyst. 35(6), 557–567 (2011)
10. Chiluveru, S.R., Tripathy, M., Mohapatra, B.: Accuracy controlled iterative method for efficient sigmoid function approximation. Electron. Lett. 56(18), 914–916 (2020)
11. Buiger, M., Omondi, A.: Low-error, high-speed approximation of the sigmoid function for large FPGA implementations. J. Signal Process. Syst. 52(2), 137–151 (2008)
12. Leboeuf, K., et al.: High speed VLSI implementation of the hyperbolic tangent sigmoid function. In: Third international conference on convergence and hybrid information technology, vol. 1, pp. 1070–1073, Busan, 11–13 November. 2008
13. Del Campo, I., et al.: Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons. Electron. Lett. 49(25), 1598–1600 (2013)
14. Dunham, C.B.: A fortran program for discrete nonlinear Chebyshev approximation. J. Comput. Appl. Math. 6(3), 241–245 (1980)
15. Abrol, S., Mahajan, R.: Implementation of single artificial neuron using various activation functions and XOR gate on FPGA chip. In: Proceedins ofSecond international conference on advances in computing and communication engineering, pp. 118–123, Dehradun, 1–5 May. 2015
16. Tawfik, S.A.: Minimax approximation and remez algorithm, MathWorks.. http://www.mathworks.com/mathcentral/fileexchange/8094-remez-algorithm (2005). Accessed 10 June 2020
17. Tsai, C.H., et al.: A hardware-efficient sigmoid function with adjustable precision for a neural network system. IEEE Trans. Circuits Syst. II Express Briefs. 62(11), 1073–1077 (2015)
18. Faeidi, H., et al.: Digital hardware implementation of a neural system used for nonlinear adaptive prediction. J. Comput. Sci. 2(4), 355–362 (2006)
19. Alippi, C., Storti.Gajani, G.: Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning. In: Proceedins of IEEE international symposium on circuits and systems, pp. 1505–1508, Singapore, 11–14 June. 1991
20. Amin, H., Curtis, K.M., Hayes.Gill, B.R.: Piecewise linear approximation applied to nonlinear function of a neural network. IEEE Proc. Circuits Device Syst. 144(6), 315–317 (1997)
21. Basterretxea, K., Taret, J., Del Campo, I.: Digital design of sigmoid approximator for artificial neural networks. Electron. Lett. 38(1), 35–37 (2002)
22. Zhang, M., Vassiliadis, S., Delgado-Frias, J.G.: Sigmoid generators for neural computing using piecewise approximations. IEEE Trans. Comput. 45(9), 1045–1049 (1996)

23. Lin, C.W., Wang, J.S.: A digital circuit design of hyperbolic tangent sigmoid function for neural networks. In: IEEE international symposium on circuits and systems, pp. 856–859, Seattle, CA, 18–21 June. 2008

24. Vassiliadis, S., Zhang, M., Delgado-Frias, J.G.: Elementary function generators for neural-network emulators. IEEE Trans. Neural Netw. 11(6), 1438–1449 (2000)

25. Armato, A., et al.: Low-error approximation of artificial neuron sigmoid function and its derivative. Electron. Lett. 45(21), 1082–1084 (2009)

26. Murugan, S., et al.: Design and implementation of multilayer perceptron with on-chip learning in Virtex-E. AASRI Procedia. 6, 82–88 (2014)

How to cite this article: Chiluvuru SR, Tripathy M. Non-linear activation function approximation using a REMEZ algorithm. IET Circuits Devices Syst. 2021;1–11. https://doi.org/10.1049/cds2.12058