PWL Approximation of Hyperbolic Tangent and the First Derivative for VLSI Implementation

Ehsan Rasekh — University of Western Ontario, London, Canada

Iman Rasekh — Islamic Azad University, Arak, Iran

Mohammad Eshghi — Shahid Beheshti University, Tehran, Iran

ABSTRACT

Hyperbolic tangent function is approximated using piecewise linear approximation. This approximation can be used in any embedded hardware architecture where occupied chip space is a challenging factor. The presented recursive algorithm makes a trade-off between circuit delay and accuracy, where low memory consumption is required. In the presented centered linear approximation, hyperbolic tangent and its first derivative is approximated and optimized using maximum error and mean square error of the approximation. Hyperbolic tangent approximation using maximum error shows better results while the first derivative of hyperbolic tangent is better approximated using mean square error. It is demonstrated that a mean square error of 0.02 can be achieved after specific number of iterations in the approximation of hyperbolic tangent.

Index Terms**—** VLSI, tanh, CRI, PWL

1. INTRODUCTION

The hyperbolic tangent (tanh) function is a common nonlinear function. This function is widely used as an activation function in artificial neural networks.

Hyperbolic tangent yields the best overall properties as an activation function of layered neural networks [1]. Common learning algorithms of these networks, such as error back propagation, employ both tanh and its first derivative.

The efficient implementation of nonlinear functions is one of the main considerations in any system on chip. Considering the occupied space and the system delay, different methods are used to implement the functions. A function can be evaluated by summation of truncated series expansion. Another way is using a look-up table in which each output value is associated with an input value, stored in the memory [2]. It can also be implemented using the piecewise approximation [3]-[6].

In comparison, piecewise approximation seems to be a better choice. Piecewise Linear (PWL) approximation reduces the area and the delay [4],[5].

The accuracy of the implementation of the function affects the results directly. With more accurate approximated activation function in a neural network design, a smaller network with fewer interconnections is obtainable [7].

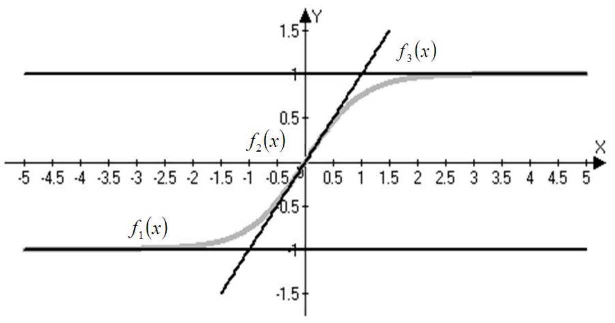

Figure 1. tanh function and a PWL approximation

Different PWL approximations are used to approximate the functions [5],[8]. The centered linear approximation (CRI) method is a PWL method used to efficiently present the nonlinear functions. CRI is a recursive method to produce PWL approximation of nonlinear functions [9],[10]. This method generates a good vertex smoothing. In the present paper, the CRI method is applied to approximate tanh.

In Sections 2 and 3 PWL approximation method of tanh and its first derivative are described. Results are discussed in Section 4. The conclusion is made in Section 5.

2. PIECEWISE LINEAR APPROXIMATION OF HYPERBOLIC TANGENT

The hyperbolic tangent of x is given by the equation:

$$f(x) = \tanh(x) = \frac{e^{x/2} - e^{-x/2}}{e^{x/2} + e^{-x/2}} \tag{1}$$

In this section a simple PWL approximation of tanh is presented using three functions, as shown in Figure 1. A recursive algorithm generates an approximation of tanh using these functions defined in equation (2).

$$f_1(x) = -1, \qquad f_2(x) = x, \qquad f_3(x) = 1 \tag{2}$$

Since tanh is an odd function, calculations can be perfumed once for either positive or negative numbers. Here

```
$f_1(x) = 1;$
$f_2(x) = x;$
for i = 0 to N
    $g(x) = \max[f_2(x), f_1(x)];$
    $f_2(x) = 1/2\,(f_1(x) + f_2(x) + \Delta);$
    $f_1(x) = g(x);$
    $\Delta = \Delta/4;$
end for;
$f_a(x) = \max[f_1(x), f_2(x)];$
```

Figure 2. A pseudo–algorithm for tanh approximation

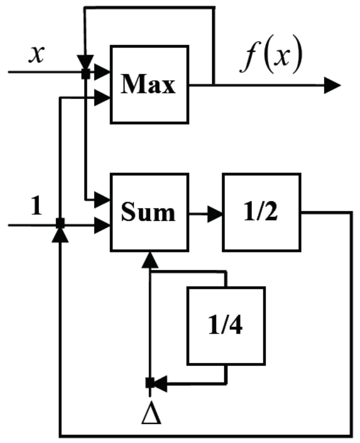

Figure 3. A block diagram of PWL tanh generator

an algorithm for computation of positive values is developed.

A pseudo–algorithm of implemented system is shown in Figure 2. In the presented pseudo-algorithm N is the interpolation level and Δ is the interpolation depth. Different interpolation levels need different optimum depth or Δ to obtain optimum results. *$f_a(x)$* is the approximated function.

Algorithm introduced in Figure 2 is implemented by the block diagram as shown in Figure 3. Both dividers in this figure are presented by simple bit shifters in hardware implementation. A 1–bit shifter stands for a by-2-divider and a 2–bit shifter stands for a by-4-divider.

For better approximation, the error of each interpolation level must be minimized. Two different norms of error are considered. First norm is Max Error (ME) described by equation (3):

$$ME_{N,\Delta} = \max|f(x) - f_a(x)| \tag{3}$$

Also Mean Square Error (MSE) is used as the other norm to optimize the system defined as follows:

$$MSE_{N,\Delta} = \sqrt{E(|f(x) - f_a(x)|^2)} \tag{4}$$

The approximation is optimized using each of these two norms. Optimization methods are utilized to achieve a Δ corresponding to the minimum error of each interpolation level [11].

Optimum approximated function along with hyperbolic tangent and approximation error is shown in Figure 4 for interpolation depths of 0, 2, 4 and 6.

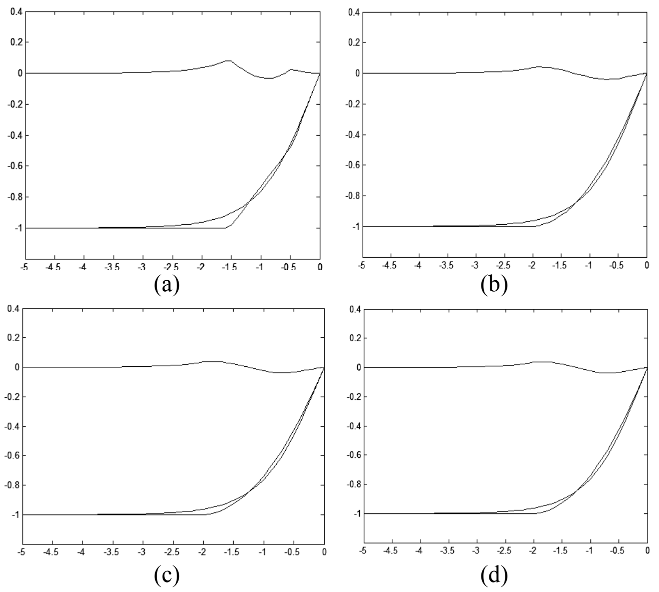

Figure 4. tanh, the approximated function and the error for 4 interpolation levels (a) N=0 (b) N=2 (c) N=4 (d) N=6

3. APPROXIMATION OF THE FIRST DERIVATIVE OF HYPERBOLIC TANGENT

The first derivative of tanh function ($\tanh'$) is given by

$$f'(x) = (\tanh(x))' = 1 - \tanh^2(x) \tag{5}$$

Equation (5) can be implemented by substituting the approximated value of tanh from Section 2. This approach needs a multiplier to produce the square function which impose much area and delay on the circuit. In this section a similar approach using CRI method is presented.

$\tanh'$ and basic functions for approximation of the function is shown in Figure 5. A recursive algorithm generates an approximation of the first derivative of tanh function using four linear functions shown in Figure 5. The four basic functions in Figure 5 are listed in equation (6).

$$\begin{aligned} f_1'(x) &= 0, & f_2'(x) &= 0.7698x + 1.1736 \\ f_3'(x) &= 1, & f_4'(x) &= -0.7698x + 1.1736 \end{aligned} \tag{6}$$

$f_2'(x)$ and *$f_4'(x)$* are the tangents to the equation (5) in its reflection points. $\tanh'$ is an even function so the values for negative inputs and positive inputs are the same. In this approximation functions *$f_1'(x)$, $f_2'(x)$* and *$f_3'(x)$* are used to estimate the output for negative values. Using *$f_3'(x)$* instead of *$f_4'(x)$* in approximation improves the results especially around zero point.

A pseudo–algorithm of implemented system is shown in Figure 6. Note that here N is the interpolation level and Δ_1 and Δ_2 are the interpolation depths. *$f_a'(x)$* is the approximated function. The algorithm in Figure 6 can be implemented in a similar way as implemented Figure 3.

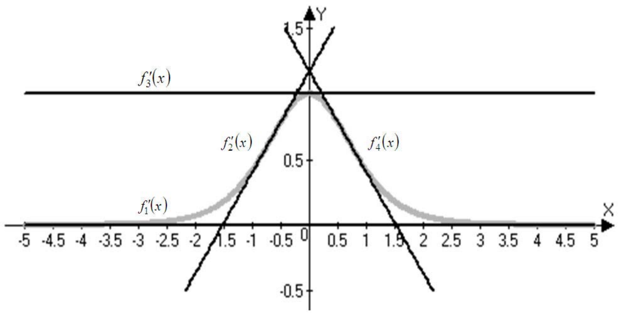

Figure 5. $\tanh'$ Function and a Simple PWL Approximation

```
$f_1' = g' = 0;$
$f_2' = 0.7698x + 1.1736;$
$f_3' = 1;$
for i = 0 to N
    $h = \min[f_2', f_3'];$
    $f_3' = 1/2(f' + f_3' - \Delta_1);$
    $f_2' = \max[g', f_3'];$
    $g' = 1/2(h + g' + \Delta_2);$
    $\Delta_1 = \Delta_1/4;$
    $\Delta_2 = \Delta_2/4;$
end for;
$h = \min[f_2', f_3'];$
$f_o' = \max[h, f_1'];$
```

Figure 6. A pseudo–algorithm for approximation of $\tanh'$

For each N, Δ values are optimized to minimize the function maximum error and mean square error (3), (4) [11]. The approximated and original first derivative of hyperbolic tangent function are shown in Figure 7 for interpolation depths of 0, 2, 4 and 6 using the optimized depths. The error curve is also shown in the same figure.

4. ANALYSIS AND RESULTS

In the previous sections, the approximated functions are optimized. The error has been assessed for 106 input data. This data uniformly spaced between -8 and 8, as it is in [8] and [10].

In approximation of hyperbolic tangent, Different interpolation levels between 1 and 6 are assumed, for each one the optimum depths with minimum $ME_{N,\Delta}$ and $MSE_{N,\Delta}$ are computed separately from equations (3) and (4). Five point decimal numbers between 0 and 1 are chosen as interpolation depths. Optimized depth for each interpolation level, after $ME_{N,\Delta}$ optimization is shown in Table I. Also optimized depths, after $MSE_{N,\Delta}$ optimization is shown in Table II. $ME_{N,\Delta}$ optimization leads to better results in compare to $MSE_{N,\Delta}$ optimization. In the approximation of the first derivative of hyperbolic tangent two interpolation

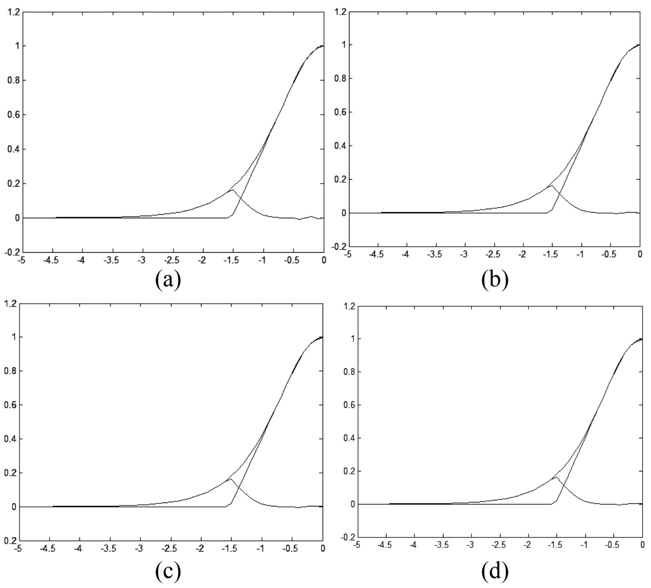

Figure 7. $\tanh'$, the approximated function and error for 4 interpolation level (a) N=0 (b) N=2 (c) N=4 (d) N=6

depths are considered. But during the optimization the best results are achieved by considering the second interpolation depth equal to zero. $MSE_{N,\Delta}$ and $ME_{N,\Delta}$ optimizations are tested to improve the approximation. For both optimization method, maximum error for each interpolation level is same and increasing interpolation depth does not improve the results. Better results are achieved using $MSE_{N,\Delta}$ optimization. An optimized depth for each interpolation level is shown in Table III.

In both approximations the approximation error is decreasing by interpolation level increment. Considering the interpolation level N, the circuit has a latency of N+1 clock cycles. Therefore in each implementation the accuracy of approximation can be changed by the latency. However using more than four interpolation levels does not make considerable improvement in accuracy.

Both approximations show good resemblance of the original and the generated synthesized function in their critical points. Resemblance of the original and the generated synthesized function in its critical point and surroundings improve the results of any implemented neural network, especially where the networks are trained offline [3].

In the presented CRI method no multiplier is needed to implement both nonlinear functions. Many other methods such as truncated expansion series need multipliers which impose considerable occupied space and more delay than the presented method. Also look up tables need much more memory to achieve same accuracy.

In [3] seven linear function are used to present the sigmoid function. The authors in [8] report a second-order PWL approximation with 2.2×10^{-3} ME. However, this second order approximation requires a multiplier. In [10] CRI structure is used to extract sigmoid function. Results

TABLE I
Optimized depths of tanh for different interpolation levels (ME)

N	Optimized Δ	$ME_{N,\Delta}$	$MSE_{N,\Delta}$
0	0.56577	0.07833	0.02067
1	0.54651	0.04825	0.01633
2	0.53003	0.04031	0.01531
3	0.52747	0.03912	0.01521
4	0.52582	0.03859	0.01513
5	0.52569	0.03861	0.01512
6	0.52565	0.03862	0.01512

TABLE II
Optimized Depth of tanh for Different Interpolation Levels (MSE)

N	Optimized Δ	$MSE_{N,\Delta}$	$ME_{N,\Delta}$
0	0.53451	0.02015	0.08621
1	0.51069	0.01503	0.05462
2	0.50086	0.01443	0.04494
3	0.49880	0.01434	0.04408
4	0.49850	0.01432	0.04349
5	0.49809	0.01431	0.04356
6	0.49808	0.01431	0.04355

TABLE III
Optimized Depth of $\tanh'$ for Different Interpolation Levels (MSE)

N	Optimized Δ_1	$MSE_{N,\Delta}$	$ME_{N,\Delta}$
0	0.11004	0.03927	0.16500
1	0.10217	0.03926	0.16500
2	0.10068	0.03925	0.16500
3	0.10001	0.03925	0.16500
4	0.09989	0.03925	0.16500
5	0.09983	0.03925	0.16500
6	0.09989	0.03925	0.16500

show good approximation error in lower approximation levels but increasing interpolation level does not lead to any improvement in approximation.

Authors in [2] implemented hyperbolic tangent by using classic lookup tables and range addressable lookup tables. A large size of memory is used in [2] to implement the same error as achieved in Section 4.

5. CONCLUSION

In the present work, a piecewise linear approximation of hyperbolic tangent and its first derivative is introduced. These approximations can be used in any hardware implementation especially where artificial neural networks is needed to be implemented.

The introduced method requires low hardware resources. The recursive algorithm makes a trade-off between circuit delay and accuracy. The approximation is optimized using maximum error and mean square error.

Hyperbolic tangent approximation shows better results through maximum error optimization; while the first derivative of hyperbolic tangent is more efficiently approximated using the mean square error.

Achieved mean square error is less than 0.02 dependent on interpolation level in hyperbolic tangent approximation. Mean square error of 0.04 is achieved in first derivative approximation. Both optimizations lead to good results around zero point which improve the VLSI implementation of neural networks.

The presented method for approximation of hyperbolic tangent and its first derivative provides high accuracy with low memory requirement.

6. REFERENCES

[1] L.B. Kalman, "*Why Tanh Choosing a Sigmoidal Function*", IEEE Electron. Lett., vol 25, IV, pp. 578–581, 1992.

[2] K. Leboeuf, A.H. Namin, R. Muscedere, Huapeng Wu, M. Ahmadi, "*High Speed VLSI Implementation of the Hyperbolic Tangent Sigmoid Function*" ICCIT '08. Third International Conference, vol.1, pp.1070-1073, 11-13 Nov. 2008.

[3] D.J. Myers, and R.A. Hutchinson, "*Efficient implementation of piecewise linear activation function for digital VLSI neural network"*, IEEE Electron. Lett., vol 25, 24, pp. 1662–1663, 1989.

[4] R. Batruni, "*A multilayer Neural Network with Piecewise–Linear Structure and Back–Propagation Learning*", IEEE Transactions on Neural Networks, , vol 2, pp. 395-403, 1991.

[5] P. Murtagh, and A.C. Tsoi, "*Implementation issues of sigmoid function and its derivative for VLSI digital neural networks*", IEE Proc.-E, Comput. Digit. Tech., vol 139, 3, pp. 207–214, 1992.

[6] C. Wen, X. Ma "*A max-piecewise-linear neural network for function approximation*" Neurocomputing, Vol 71 , 4-6, pp. 843-852, 2008.

[7] K. Basterretxea, J.M. Tarela, and I. Del Campo, "*Consequences of Gaussian function approximation in the performance of neuro-fuzzy system"*. Proc. 4th Int. Conf. on Recent advances in soft computing, Nottingham, UK, vol 1, pp 313–318, 2002.

[8] M. Zhang, V. Vassiliadis, and J.G. Delgado-Frias, "*Sigmoid generators for neural computing using piecewise approximations*", IEEE Trans. Comput., vol 45, (9), pp 1045–1049., 1996.

[9] K. Basterretxea, J.M. Tarela, I. Del Campo, M.V. Martiınez, and E. Alonso, "*Optimised PWL recursive approximation and its application to neuro-fuzzy systems*", Math. Comput. Model, vol 35, (7–8), pp 867–883, 2002.

[10] K. Basterretxea, J.M. Tarela, and I. Del Campo, "*Approximation of Sigmoid Function and The Derivative for Hardware Implementation of Artificial Neurons*" IEE Proc.–Circuits Devices Syst., vol 151, pp 18–24, 2004.

[11] G. N. Vanderplaats, *Numerical Optimization Techniques for Engineering Design*, New York: McGraw-Hill, 1984.