A Digital Circuit Design of Hyperbolic Tangent Sigmoid Function for Neural Networks

Che-Wei Lin and Jeen-Shing Wang
Department of Electrical Engineering
National Cheng Kung University
Tainan 701, Taiwan, R.O.C.
jeenshin@mail.ncku.edu.tw

***Abstract*—This paper presents a digital circuit design approach for a commonly used activation function, hyperbolic tangent sigmoid functions, for neural networks. Our design concept for such a nonlinear function is to approximate the function of its first-order derivative by piece-wise linear functions first, then to obtain the estimate of the original function by integrating the approximated function of the first-order derivative by a digital circuit. The average error and maximum error of the proposed approximation approach are in the order of 10^{-3} and 10^{-2}, respectively in the software simulation. The hardware implementation of the proposed method consumes only one multiplication and one addition/subtraction ALU with the aid of resource sharing. The performance of our circuit has been validated by a neural network for a system identification problem in the software simulation.**

I. INTRODUCTION

Artificial neural networks have developed rapidly in the past decades both in theory and in hardware implementation. Current research on hardware implementation is working toward DSP, FPGA, and ASIC implementation [1]-[3]. Membership functions such as sigmoid function and hyperbolic tangent-sigmoid function are difficult to realize directly in circuits or low-end hardware implementation due to their exponential terms. Existing function approximation methods can be categorized into the following types: 1) piecewise linear approximation (PWL), 2) look up table (LUT) [4], 3) direct computation [5], and 4) hybrid methods [6]. PWL is widely used due to its simple and straightforward concept. The circuit using the LUT method is easy to implement and inexpensive when applications do not require high precision. The main drawback of the LUT method is that the table size increases exponentially when the precision requirements are higher. Direct computation such as polynomial power series evaluations provides high precision, but it needs a large amount of additions and multiplications. The hybrid approach usually includes the advantages of LUT, PWL, and direct computation.

Activation functions of neural networks can be categorized into two types: continuous functions, e.g. sigmoid functions and piece-wise linear functions, e.g. isosceles triangular functions. Continuous functions are usually chosen as the activation functions of neural networks because of their preferable nonlinearity and the requirement of derivative-based parameter learning methods such as backpropagation. However, using the aforementioned methods to approximate a continuous function may yield discontinuities in the function of its first-order derivative and extra hardware cost to approximate the function itself and its derivative. In this study, we focused on the implementation of an inexpensive function approximation method on hyperbolic tangent sigmoid functions. In order to preserve the continuity property of the first-order derivative, our proposed method first approximates the first-order derivative of the original function. The original function's approximation is obtained by integrating the first-order derivative approximation. Different from other direct computation methods, the proposed approximation method on the original function is a second-order polynomial at most in our design. Thus, the computation complexity of our approach is greatly reduced by the analysis of the approximate polynomial and substitutions of some arithmetic operations. In the circuit implementation, the function unit usage is reduced with the aid of resource sharing on multiplier. The average error and maximum error of the hyperbolic tangent sigmoid function approximation obtained by our approach are in the order of 10^{-3} and 10^{-2}, respectively. For dynamic system identification problems, the performance of the neural network with our approximated activation functions is close to the performance of the neural network with the original functions.

The rest of this paper is organized as follows. In section II, we present the motivation and design approach. In section III, hardware implementation procedures are explained in detail. In section IV, we compare the precision of our approximated function with and the original function and compare the performance of both two functions when applied to neural networks. Finally, the conclusions are presented in Section V.

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

II. DESIGN APPROACH

The design concept of the proposed method is to approximate the first-order derivative of the original function first, then to obtain the original function by integrating the first-order derivative approximation. Two different schemes are utilized to approximate the first-order derivative. The two schemes yield different precision, hardware resource usage, and performance when applied in a neural network. We illustrate our design through a hyperbolic tangent sigmoid function approximation. Our approach can be applied to other similar functions straightforwardly. The hyperbolic tangent sigmoid function and its first-order derivative are defined as (1) and (2).

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

$$tanh'(x) = \frac{4}{(e^{x} + e^{-x})^{2}}. \quad (2)$$

A. Scheme-1: Isosceles Triangular Approximation

Scheme-1 approximates the first-order derivative through an isosceles triangular function since the shape of the first-order derivative of the hyperbolic tangent sigmoid function is similar to that of an isosceles triangular function. In scheme-1, the initial value and final value are set as 0 to match the condition in (2). The upper vertex of the isosceles triangular function is chosen to be the same as the maximum value of the first-order derivative. The maximum value of the $tanh'(x)$ occurs at 0 and the value is 1. The next issue is how to determine the lower vertex of the isosceles triangular function. Since the value of the original function is obtained by integrating the first-order derivative from negative infinity to a specific point, thus the different selection of the lower vertex of the isosceles triangular function affects the value of the original function. In (1), the values of the negative infinity, zero, and positive infinity are -1, 0, and 1 respectively. We want to maintain the values in the -1, 0, and 1 in our approximation. Thus, the region enclosed by $tanh'(x)$ and x-axis from negative infinity to zero and zero to infinity should be 1 in order to match the value of negative infinity, zero, and positive infinity. Thus, the two lower vertexes of the isosceles triangular function are chosen as ±2. The first-order derivative approximate function of Scheme-1 is defined as (3); the plots of the approximate function and the original function of (2) are shown in Fig. 1. The approximate $tanh(x)$, denoted as $tanh\_a(x)$, is defined as (4), which is obtained by integrating (3). The plots of the approximate function and the original function of (1) are shown in Fig. 2.

$$tanh\_a'(x) = \begin{cases} 1 - \frac{|x|}{2} & , 0 \le |x| \le 2; \\ 0 & , \text{otherwise.} \end{cases} \quad (3)$$

$$tanh\_a(x) = \begin{cases} x - 0.25 \times sign(x) \times x^{2} & , 0 \le |x| \le 2; \\ sign(x) & , \text{otherwise.} \end{cases} \quad (4)$$

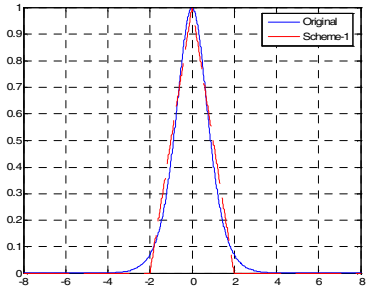

Figure 1. Plots of the first-order derivative of $tanh(x)$ and its approximation (Scheme-1).

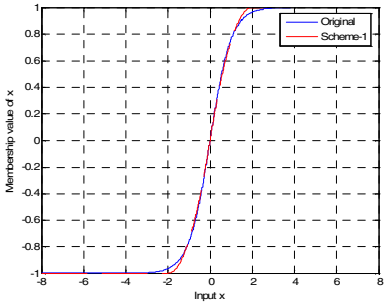

Figure 2. Plots of $tanh(x)$ and its approximation (Scheme-1).

B. Scheme-2: Piecewise Linear Approximation

Scheme-2 approximates the first-order derivative through three symmetric piecewise linear segments, as shown in Fig. 3. The maximum point is chosen to be the same as in scheme-1. The objective of the selections on the transition points is to minimize the absolute error between $tanh'(x)$ and the approximate $tanh'(x)$ as shown in (5). The approximate function of the first-order derivative is defined as (6), where m_1, m_2, c_1, c_2, a, b are -0.54324, -0.16957, 1, 0.42654, 1.52, and 2.57, respectively. The approximate function of $tanh(x)$ is obtained by integrating (6) and is shown as (7) where d_1 and d_2 are 0.016 and 0.4519, respectively. The plots approximation of $tanh(x)$ and the first-order derivative are shown in Fig. 3 and Fig. 4.

$$\min\left(|tanh'(x) - tanh\_a'(x)|\right). \quad (5)$$

$$tanh\_a'(x) = \begin{cases} m_1 \times sign(x) \times x + c_1 & , 0 \le |x| \le a; \\ m_2 \times sign(x) \times x + c_2 & , a \le |x| \le b; \\ 0 & , \text{otherwise.} \end{cases} \quad (6)$$

$$tanh\_a(x)=\begin{cases} sign(x)\times[0.5\times m_1\times|x|^2+c_1\times|x|+d_1] \\ \qquad\qquad ,0\le|x|\le a; \\ sign(x)\times[0.5\times m_2\times|x|^2+c_2\times|x|+d_2] \\ \qquad\qquad ,a\le|x|\le b; \\ sign(x) \qquad\qquad \text{, otherwise.} \end{cases} \quad (7)$$

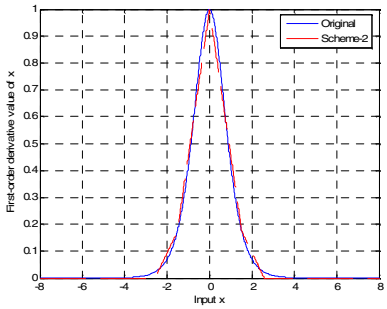

Figure 3. Plots of the first-order derivative of *tanh*(*x*) and its approximation (Scheme-2).

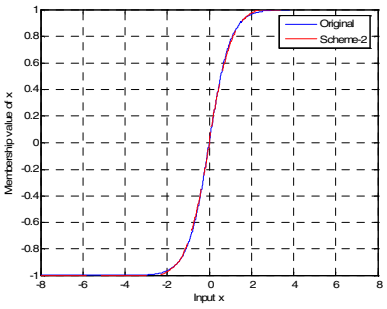

Figure 4. Plots of *tanh*(*x*) and its approximation (Scheme-2).

III. HADEWARE IMPLEMENTATION

In this section, we illustrate the procedure for implementing the above (4) and (7) into digital circuits. A 24-bit data representation consisting ten integer bits and fourteen fractional bits is employed in our circuit. The objective of our design is to minimize the function unit usage and time consumption. We first try to reduce the operation time in the circuit. Based on the odd-function property of the *tanh*(*x*), the approximation can be simplified by (4) and (7). This simplification greatly reduces the computational time. Then we simplify the complicated operations in (4) and (7) by the following procedures. The operations that multiply by 0.25 or 0.5 are replaced by left-shifting 2 or 1 bit to substitute the corresponding multiplications. Multipliers are shared in our design since they are area expensive function units in the digital circuit. In (7), four multiplications are required if all multiplications need to be completed in a control step. Because we only use a multiplier in the design, we can save a lot of area with a reasonable increase in execution time.

The function units of the circuits are shown in Fig. 5 and Fig. 6. The two circuits both employ one multiplication and one arithmetic function unit (ALU) which can perform addition and subtraction through different control signal selections. All the multiplexers and de-multiplexers in Fig. 5 and Fig. 6 are controlled by a control circuit which is formulated by a finite state machine (FSM). Finally, the proposed schemes for the hyperbolic tangent sigmoid function are coded in Verilog and synthesized by Synopsys™ Design Complier in UMC 0.18μm technology. The area consumption of the proposed schemes is shown in Table I and the throughput rate is shown in Table II.

TABLE I. AREA CONSUMPTION OF THE PROPOSED SCHEMES

Method	Area consumption(μm^2)
Scheme-1	32069.837891
Scheme-2	83559.171875

TABLE II. THROUGHPUT RATE OF THE PROPOSED SCHEMES

Method	Throughput rate (MHz)
Scheme-1	1.107
Scheme-2	0.773

TABLE III. AVERAGE ERROR AND MAXIMUM ERROR OF PROPOSED SCHEMES

Method	Average error	Maximum error
Scheme-1	7.8×10^{-3}	4.3×10^{-2}
Scheme-2	4.1×10^{-3}	2.2×10^{-2}

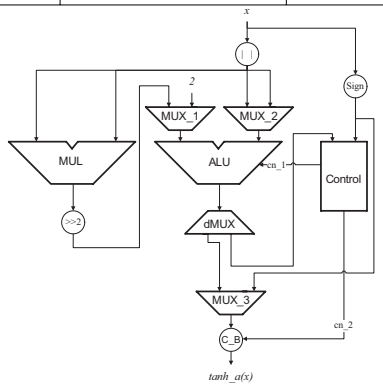

Figure 5. Function units of the hardware implementation for Scheme-1.

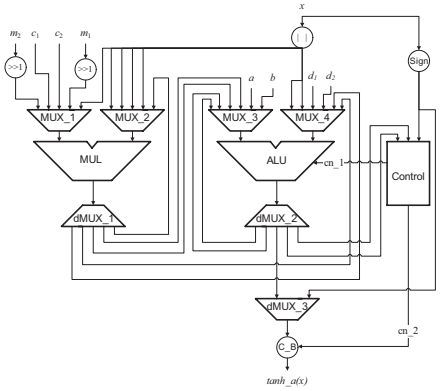

Figure 6. Function units of the hardware implementation for Scheme-2.

IV. SIMULATION

The performance of the proposed two schemes is evaluated in two ways. The first is precision comparison and the second is the performance comparison by applying the approximated function and original function in a neural network. The precision is measured by the average and maximum error of the proposed schemes. The average and maximum errors are obtained from the computer simulation with the sample interval from -8 to 8. The results are shown in Table III.

The neural network structures [7] with the original activation function and its approximation are applied to emulate the dynamics of a nonlinear dynamic system. The simulation is based on the same initial condition with different activation functions, the original function and its approximation obtained by the two proposed schemes. The approximate functions in (4) and (7) of the two schemes and their first-order derivatives in (3) and (6) are utilized in the computation and parameter learning phase of the neural network training. The system outputs obtained by neural networks that use the approximated functions and original functions are shown in Fig. 7. The performances of the original function and functions obtained from the two proposed schemes are evaluated by the mean square error (MSE) and listed in Table IV. The MSE of Scheme-1 is even better than that of the original *tanh*(*x*) function. The function approximation of Scheme-1 creates a better performance for the neural network in this dynamic system identification.

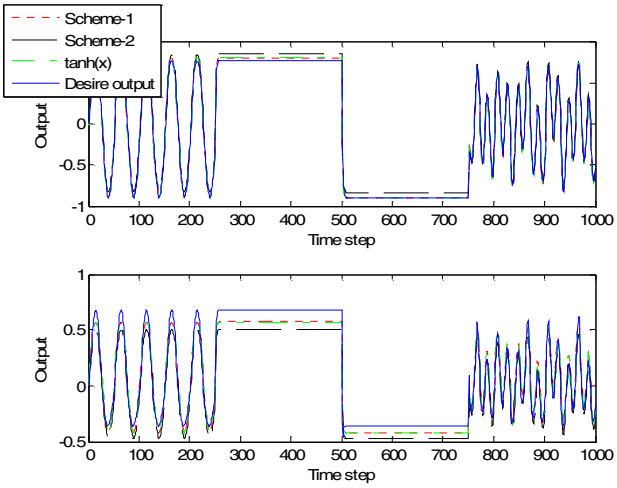

Figure 7. The system outputs obtained by neural networks that use the approximated functions obtained by Scheme-I and Scheme-II, and the original functions.

TABLE IV. PERFORMANCE COMPARISON BETWEEN ORIGINAL FUNCRION AND FUNCTIONS OBTAINED BY TWO PRPOSED SCHEMES

Method	Mean Square Error (MSE)	Time Consumption(Sec.)
Scheme-1	4.2×10^{-3}	7.0630
Scheme-2	5.5×10^{-3}	7.1090
tanh(*x*)	4.5×10^{-3}	9.6090

V. CONCLUTION

Nonlinear activation functions, such as hyperbolic tangent sigmoid and sigmoid functions conventionally applied in neural networks, are difficult to implement in low-end hardware implementation. The two proposed schemes provide efficient approximation methods for hyperbolic tangent sigmoid functions and can also be applied to other nonlinear function approximation. In our simulations, the Scheme-2 provides better approximation accuracy but obtains a worse performance in the system identification problem. The Scheme-1 provides a lower accuracy in the function approximation but outperforms the neural network with the original functions in the same problem. Due to the lower computation complexity of the proposed method than that of the original *tanh*(*x*), the time consumption in the computer simulation requires only 73% (Scheme-1) and 74% (Scheme-2) of the computational time using the original *tanh*(*x*) function. With the aid of resource sharing, Scheme-1 without a look-up table utilizes the same function units as the one in [6] does. Scheme-2 performs better in approximation accuracy but worse in the identification performance of neural networks. In conclusion, we have presents an efficient digital circuit design of a hyperbolic tangent sigmoid function for neural networks. The elimination of multipliers in the proposed schemes may be considered as a future research work since they may consume a large area in digital circuit realization. However, since multiplication is a necessary operation that exists in the computation of neural networks, one can employing our design approach for activation functions by utilizing the existing multipliers of his/her neural network hardware. Under this design concept, our proposed schemes will not cause any area cost on multiplication.

REFERENCES

[1] F. Yang, and M. Paindavoine, "Implementation of an RBF Neural Network on Embedded Systems: Real-Time Face Tracking and Identity Verification," *IEEE Trans. on Neural Networks*, vol. 14, no. 5, pp. 1162-1175, Sep. 2003.

[2] C.-F. Juang, and J.-S. Chen, "Water Bath Temperature Control by a Recurrent Fuzzy Controller and Its FPGA Implementation," *IEEE Trans. on Industrial Electronics*, vol. 53, no. 3, pp. 941-949, Jun. 2006.

[3] B. Burton, F. Kamran, R.G. Harley, T.G. Habetler, M.A. Brooke., and R. Poddar, "Identification and Control of Induction Motor Stator Currents Using Fast On-line Random Training of a Neural Network." *IEEE Trans. on Industry Applications*, vol. 33, no. 3, pp. 697-704, Jun. 1997.

[4] P. Treleaven, M. Pacheco, and M. Vellasco, "VLSI Architectures for Neural Networks," *IEEE Micro*., vol. 9, no. 6, pp. 8-27, Dec. 1989.

[5] B. Lee, and N. Burgess, "Some Results on Taylor-Series Function Approximation on FPGA," in *Proc. of Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 2198-2202, Nov. 2003.

[6] S. Vassiliadis, M. Zhang, J. G. Delgado-Frias, "Elementary Function Generators for Neural-Network Emulators," *IEEE Trans. on Neural Networks*, vol. 11, no. 6, pp. 1438-1449, Nov. 2000.

[7] J.-S. Wang and Y.-P. Chen, "A Fully Automated Recurrent Neural Network for Unknown Dynamic System Identification and Control," *IEEE Trans. on Circuit and Systems-I*, vol. 53, no. 6, pp. 1363-1372, Jun. 2006.