Efficient hardware implementation of the hyperbolic tangent sigmoid function

Conference Paper *in* Proceedings - IEEE International Symposium on Circuits and Systems · June 2009

DOI: 10.1109/ISCAS.2009.5118213 · Source: IEEE Xplore

Karl Leboeuf
University of Windsor
12 PUBLICATIONS **105** CITATIONS
SEE PROFILE

Huapeng Wu
University of Windsor
69 PUBLICATIONS **960** CITATIONS
SEE PROFILE

Majid Ahmadi
University of Windsor
565 PUBLICATIONS **4,616** CITATIONS
SEE PROFILE

Efficient Hardware Implementation of the Hyperbolic Tangent Sigmoid Function

Ashkan Hosseinzadeh Namin, Karl Leboeuf, Roberto Muscedere, Huapeng Wu, and Majid Ahmadi
Department of Electrical and Computer Engineering, University of Windsor
Windsor, Ontario N9B 3P4 Canada
Email: {hossei1,leboeu3,rmusced,hwu,ahmadi}@uwindsor.ca

Abstract— **Efficient implementation of the activation function is important in the hardware design of artificial neural networks. Sigmoid, and hyperbolic tangent sigmoid functions are the most widely used activation functions for this purpose. In this paper, we present a simple and efficient architecture for digital hardware implementation of the hyperbolic tangent sigmoid function. The proposed method employs a piecewise linear approximation as a foundation, and further improves the results using a lookup table. Our design proves to be more efficient considering area $\times$ delay as a performance metric when compared to similar proposals. VLSI implementation of the proposed design using a $0.18\,\mu m$ CMOS process is also presented, which shows a 35% improvement over similar recently published architectures.**

Index terms : Artificial neural networks, Sigmoid function, Hyperbolic tangent function, Hardware implementation, Piecewise linear approximation.

I. Introduction

Artificial neural networks (ANNs) have a wide range of applications including, but not limited to, signal processing, system control, function approximation, and optimization [1], [2]. ANNs were typically implemented only in software until recently, wherein their hardware implementation has become important [3], [4]; this is largely due to the performance gains of hardware systems compared to software implementations. Special care must be taken into account when designing every computational element in ANN hardware design. One of the important components is the nonlinear activation function, which is used at the output of every neuron. Several different activation functions are available today including the sigmoid, hyperbolic tangent (tanh), and step functions [1], [2]. The tanh and sigmoid functions both produce a curve with an "S" shape, where the tanh output varies between [-1,1] and the sigmoid output varies between [0,1]. Mathematically, the tanh function is defined as eq. 1, which is shown in Fig. 1.

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

Straightforward implementation of the tanh function in hardware is not practical because it will require exponentiation and division, both of which are expensive operations. Several different approaches exist for the hardware implementation of the activation functions, including piecewise linear approximation, piecewise non-linear approximation, and lookup tables (LUTs) [5], [6], [9], [10].

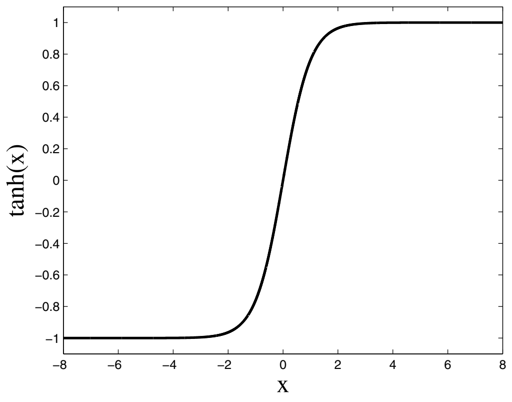

Fig. 1. The Hyperbolic Tangent Activation Function

Generally, lookup table implementations are the fastest, while they consume a large area of silicon. Piecewise linear approximations are slow, but consume less area; they are also the most common way of implementing the activation functions. Piecewise non-linear approximations achieve excellent approximation (an approximation with low maximum error) but are not fast, as they usually make use of multipliers in their architectures.

In this work, a simple architecture for the VLSI implementation of the tanh function is presented. The proposed design uses a piecewise linear approximation, and then improves the results by subtracting a pre-determined amount of error, which is stored in a lookup table. It is shown that the new design is more efficient compared to other similar proposals when considering the product of area and delay as a measure of performance. The hardware implementation of the proposed design is also presented, which compares favorably with similar proposals.

The rest of this paper is organized as follows. Section 2 briefly explores different methods for the VLSI implementation of the tanh function. Section 3 proposes a new approach for the approximation of the tanh function, while section 4 presents the hardware implementation in detail. In section 5, complexity comparisons between the proposed design and several different similar methods are presented. Finally, section 6 discusses some concluding remarks.

II. A Brief Review of Different Methods for the Approximation of the Hyperbolic Tangent Function

A. Piecewise Linear Approximation

The Piecewise linear approximation method approximates the function with a limited number of linear segments as shown in Fig. 2. More accurate approximations can be achieved by increasing the number of segments used, which will result in greater area utilization. This is the most widely used method to approximate the activation function [5], [6], [7]. It is preferable to avoid the use of multipliers with this method due to their negative effect on circuit throughput and area utilization.

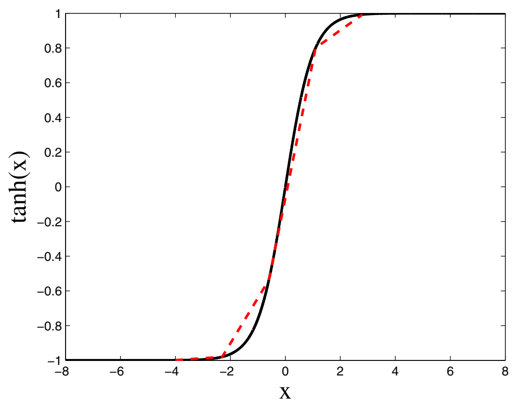

Fig. 2. The Piecewise Linear Approximation of the Hyperbolic Tangent Function with 4 segments

B. Lookup Table Approximation

With this approach, the function is approximated by a limited number of uniformly distributed points [8], [10]. There is a direct relation between the number of bits used, and the maximum approximation error. This method presents the fastest design, however it will require large memory use to store the lookup table. Alternately, a hardware synthesizer can be used to implement the design entirely as a combinational logic circuit. As a higher degree of accuracy is needed, the area requirements increase exponentially, rendering this approach impractical for large table sizes.

C. Piecewise Nonlinear Approximations

The activation function can also be implemented using piecewise approximations composed of non-linear segments. The simplest case uses second order approximations as follows in eq. 2.

$$y = a_0 + a_1 \times x + a_2 \times x^2 \tag{2}$$

The main drawback of this method is that it makes use of multipliers, which results in a low conversion speed, however accurate approximations can be achieved [12].

III. Proposed Method for the Approximation of the Hyperbolic Tangent Function

Our proposed system initially makes use of a piecewise linear approximation with two segments to roughly approximate the positive half of the tanh curve. The linear segments were selected to be $y = x$ for the interval $[0, 1]$, and $y = 1$ for $[1, 8]$, which will be referred to as segment-1 and segment-2, respectively. The odd symmetry of the tanh function was exploited to calculate the output for input range $[-8, 0]$.

Afterwards, the difference between the actual function's value and the approximated value will be adjusted to achieve a better approximation. The adjustment values are calculated offline and stored in a lookup table. These adjustments consist of subtracting the values stored in the lookup table from the initial piecewise linear approximation. This can be achieved with a simple combinational subtracter circuit.

The proposed method has two main advantages. First, segment-1 and segment-2 were chosen to avoid the use of a multiplier in the design. Since multipliers are expensive in terms of area and throughput, this is highly beneficial. Second, the lookup table used in our design stores the difference between the piecewise linear approximation and the actual function. This greatly minimizes the range of the output variance, which reduces the size of the lookup table.

It is also worth noting that some additional area savings are achieved in the subtracter module. Since the maximum value of the lookup table is relatively small, fewer bits are required for representation. This will reduce the width for one of the subtracter module inputs, which will result in area savings.

To further optimize the design of the lookup table, a range addressable lookup table (RALUT) was used instead of a classic lookup table. In an RALUT, every output corresponds to a range of input addresses. This can be highly efficient in situations where the output does not change over certain input ranges [11]. In this design, for example, the difference between segment-1 and the value of the tanh function remains small and relatively constant for the interval $[0, 0.3]$. A classic lookup table would possess a unique (and relatively constant) output for every address in this range, whereas a range addressable lookup table stores a single output.

The block diagram of the proposed system that achieves a maximum error of 0.02 is shown in Fig 3. Segment-1 and segment-2 of the piecewise linear approximation are shown in Fig. 4. The final system output of the tanh function approximator, along with the error is shown in Fig. 5. The system's inputs and outputs are registered. The registered input data will then enter the RALUT module, and the comparator module in parallel.

As mentioned earlier, the RALUT module contains the difference between the linear approximation $y = x$, and tanh for the range $[0, 1]$, and the difference between the linear approximation of $y = 1$ and tanh for the range $[1, 8]$.

The input-output relationship of the comparator module can

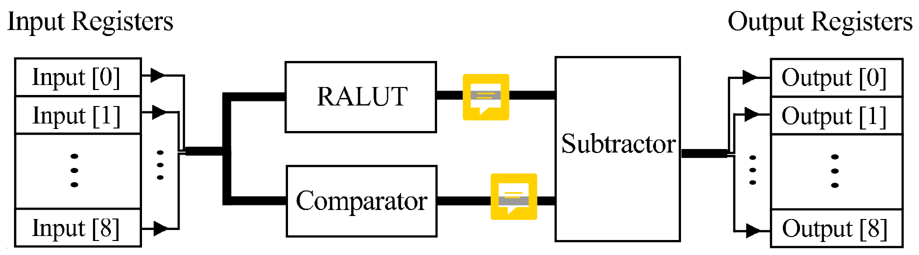

Fig. 3. Proposed System Block Diagram for a Maximum Error of 0.02

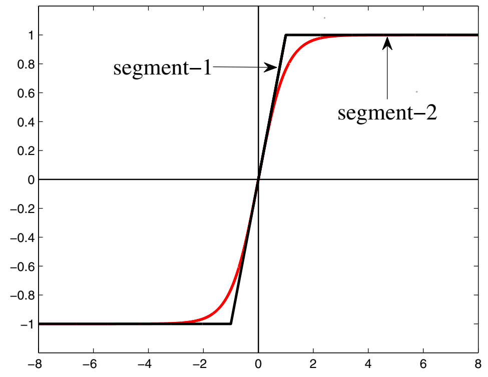

Fig. 4. Segment-1 and Segment-2 of the Piecewise Linear Approximation

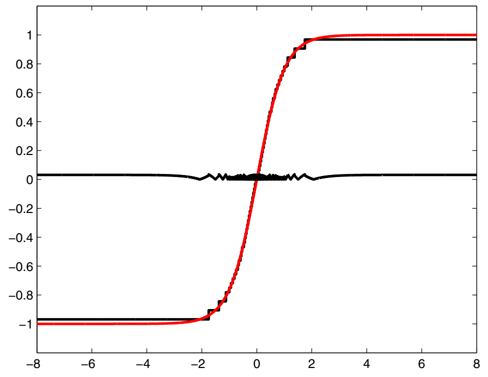

Fig. 5. Hyperbolic Tangent Function Approximation and Error for a Maximum Error of 0.04

be expressed as follows:

$$y = \begin{cases} x & 0 \leq x \leq 1, \\ 1 & 1 \leq x \leq 8. \end{cases}$$

In other words, the comparator output is equal to the input over $[0, 1]$, and is equal to the constant value 1 for the rest. The subtracter module subtracts the output of the comparator module from the output of the RALUT module. The result of the subtracter module will then enter the registers to create the system's output.

IV. Hardware Implementation

Hardware implementation began with system-level design in MATLAB to determine system parameters, including the required number of bits for an accurate representation, and the size of the RALUT for a specified amount of maximum error.

For a maximum error of 0.04, a 9-bit representation was used for both the input and output, whereas a 10-bit representation was required to keep the maximum error below 0.02. RALUT modules were synthesized as a combinational circuit for our design; this allows for a fair comparison with other architectures.

After the main system parameters were determined, hardware description language (HDL) code was written, defining the system in hardware. Next, the HDL code was synthesized using Synopsys' Design Compiler to library gates. Virtual Silicon standard library cells for a TSMC $0.18\mu m$ CMOS process were selected for this step. Synthesis parameters were chosen to maximize operating speed of the system. Hardware implementation results with a maximum error of 0.04 and 0.02 are summarized in the fourth row of tables I and II respectively.

V. Comparison Between Similar Hardware Implementations

Comparison between different hardware implementation results for 0.02 and 0.04 maximum error are summarized in tables I and II, respectively. All hardware implementations use a CMOS $0.18\mu m$ process.

In the tables, the first line presents scheme-1 (table I) and scheme-2 (table II) proposed by C.W. Lin and J.S. Wang. Scheme-1 approximates the first order derivative of tanh with an isosceles triangular function, and scheme-2 approximates the first order derivative of the tanh through three symmetric piecewise linear segments. These approximated derivatives are then integrated to yield an approximation of tanh. The second row of the tables present the lookup table approximations of the tanh function, while the third row presents the range addressable lookup table approximations [8]. The last row of each table presents our proposed designs.

As can be seen from the tables, lookup table implementations have the highest speed between different proposals, where RALUT implementation is even faster than the classic lookup table implementation. From an area utilization point

Architectures	Max-Error	AVG-Error	Area	Delay	Area $\times$ Delay
Scheme-1 [7]	0.0430	0.0078	32069.83 μm^2	903 ns	2.895×10^{-5}
LUT [8]	0.0365	0.0040	9045.94 μm^2	2.15 ns	1.944×10^{-11}
RALUT [8]	0.0357	0.0089	7090.40 μm^2	1.85 ns	1.311×10^{-11}
Proposed	0.0361	0.0246	3646.83 μm^2	2.31 ns	0.842×10^{-11}

TABLE I

COMPLEXITY COMPARISON OF DIFFERENT IMPLEMENTATIONS FOR 0.04 MAXIMUM ERROR

Architectures	Max-Error	AVG-Error	Area	Delay	Area $\times$ Delay
Scheme-2 [7]	0.0220	0.0041	83559.17μm^2	1293 ns	1.080×10^{-4}
LUT [8]	0.0180	0.0020	17864.24 μm^2	2.45 ns	4.376×10^{-11}
RALUT [8]	0.0178	0.0057	11871.53 μm^2	2.12 ns	2.516×10^{-11}
Proposed	0.0189	0.0121	5130.78 μm^2	2.80 ns	1.436×10^{-11}

TABLE II

COMPLEXITY COMPARISON OF DIFFERENT IMPLEMENTATIONS FOR 0.02 MAXIMUM ERROR

of view, our proposed design outperforms all other proposals in the table.

Since minimizing the critical path delay and area utilization are both important goals in hardware design, we have defined a new cost function, area $\times$ delay as a performance metric, which is presented in the last column of the comparison tables. As it is shown in the tables our proposed architecture performs 35% and 43% better than the next proposal (RALUT) for 0.04 and 0.02 maximum error respectively.

VI. CONCLUSIONS

An efficient architecture for the digital hardware implementation of the hyperbolic tangent sigmoid function was proposed. The proposed method employs a piecewise linear approximation as a foundation, and further improves the results by using a look up table and a subtracter. VLSI implementation of the proposed design using the $0.18\,\mu m$ CMOS process from TSMC was presented. It was shown that the proposed design outperforms all other similar proposals when considering area $\times$ delay as a measure of performance.

VII. ACKNOWLEDGEMENT

The authors would like to acknowledge financial support from the Natural Sciences and Engineering Council (NSERC) of Canada. The authors also acknowledge the important contributions of the Canadian Microelectronics Corporation (CMC) for their equipment, software loan, and fabrication services.

REFERENCES

[1] S. Haykin, ”Neural networks : a comprehensive foundation“, Prentice Hall, July, 1998.
[2] D.E. Rumelhart and J.L. McClelland and the PDP Research Group,“Parallel Distributed Processing, Vol. 1: Foundations”, The MIT Press, July, 1987.
[3] M. Krips, T. Lammert, and A. Kummert ,“FPGA implementation of a neural network for a real-time hand tracking system”, Proc. 1st IEEE Int. Workshop on Electronic Design, Test and Applications (DELTA), Christchurch, New Zealand, 29-31 January 2002, pp. 313-317.
[4] A.R. Omondi, and J.C. Rajapakse, “Neural networks in FPGAs”. Proc. 9th Int. Conf. on Neural Information Processing (ICONIP), Singapore, 18-22 November 2002, vol. 2, pp. 954-959.
[5] K. Basterretxea and J.M. Tarela and I. Del Campo, “Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons”,IEE Proceedings - Circuits, Devices and Systems, vol. 151, no. 1, pp. 18-24, February 2004.
[6] K. Basterretxea and J.M. Tarela, “Approximation of sigmoid function and the derivative for artificial neurons”, advances in neural networks and applications, WSES Press, Athens, pp.397-401, 2001.
[7] C. Lin and J. Wang, “A digital circuit design of hyperbolic tangent sigmoid function for neural networks”, IEEE International Symposium on Circuits and Systems (ISCAS), pp. 856-859, May 2008.
[8] K. Leboeuf, A.H. Namin, R. Muscedere, H. Wu, and M. Ahmadi, “High Speed VLSI Implementation of the Hyperbolic Tangent Sigmoid Function ”, Third International Conference on Convergence and hybrid Information Technology, accepted, November 2008.
[9] H.K. Kwan, “Simple sigmoid-like activation function suitable for digital hardware implementation” Electronic Letters,vol. 28, no. 15, pp. 1379-1380, July 1992 .
[10] F. Piazza, A. Uncini, and M. Zenobi, “Neural networks with digital LUT activation functions”, Proceedings of 1993 International Joint Conference on Neural Networks, IJCNN, pp. 1401-1404, October 1993.
[11] R. Muscedere, V. Dimitrov, G.A. Jullien, and W.C. Miller,“Efficient techniques for binary-to-multidigit multidimensional logarithmic number system conversion using range-addressable look-up tables”, IEEE Transactions on Computers, vol. 54, no. 3, pp. 257-271, March 2005.
[12] M. Zhang, S. Vassiliadis, and J.G. Delgago-Frias, “Sigmoid generators for neural computing using piecewise approximations” IEEE Trans. Comput. vol. 45, no. 9, pp. 1045-1049, 1996.