Artificial neural networks activation function HDL coder

Conference Paper · July 2009

DOI: 10.1109/EIT.2009.5189648 · Source: IEEE Xplore

Karl Leboeuf
University of Windsor
12 PUBLICATIONS **179** CITATIONS
SEE PROFILE

Huapeng Wu
University of Windsor
69 PUBLICATIONS **1,157** CITATIONS
SEE PROFILE

Majid Ahmadi
University of Windsor
577 PUBLICATIONS **5,884** CITATIONS
SEE PROFILE

Artificial Neural Networks Activation Function HDL Coder

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

Abstract— **The sigmoid and hyperbolic tangent functions are usually used as the activation functions in Artificial Neural Networks (ANNs). The exponential nature of these functions make them difficult for hardware implementation. Hence, several different methods for approximating them in hardware are proposed. In this work, we present a MATLAB toolbox called the "SigTan HDL Coder", that generates synthesizable HDL Code which approximates these functions in hardware according to the specific user requirements. The HDL code is platform independent and can be used for FPGA as well as ASIC implementations. Input parameters to the system are the approximation error, input range, and the approximation method. Three different user-selectable methods for approximating the functions are programmed in the toolbox. All implemented approximation methods avoid the use of multipliers for their implementation, as multipliers are expensive hardware components in terms of area and speed.**

Index terms : Hardware implementation, Sigmoid function, Hyperbolic tangent function, Electronic Design Automation, MATLAB, Toolbox.

I. Introduction

Artificial Neural Networks (ANNs) are used for a wide range of applications including filter design, pattern recognition, nonlinear system identification, system control, and function approximation [1], [2]. In the past, ANNs were typically implemented in software. It is not until recently that their hardware implementation has become common [3], [4]; this is mainly due to the performance gains of hardware systems compared to software implementations.

One of the important components in designing ANNs in hardware is the nonlinear activation function, which is used at the output of every neuron. The most common used activation functions are the sigmoid and the hyperbolic tangent functions [1], [2]. Mathematically, the sigmoid function is defined as eq. 1, and the hyperbolic tangent function is defined as eq. 2 which are both shown in Fig. 1.

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

$$Tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

Straightforward implementation of these functions in hardware is not practical due to their exponential nature. Several different approaches exist for the hardware approximation of activation functions, including piecewise linear approximation, piecewise non-linear approximation, lookup tables (LUT), and hybrid methods [5], [6], [7], [9], [10], [11], [12].

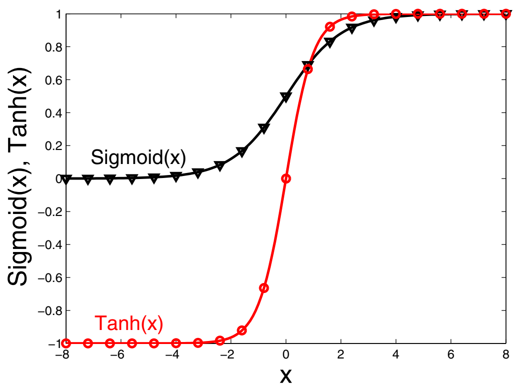


Fig. 1. The Hyperbolic Tangent and The Sigmoid Activation Functions

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 the slowest type of approximation, as they usually make use of multipliers in their architectures. Hybrid methods typically achieve good performance regarding area and delay as they employ a combination of approaches, such as the use of piecewise approximation with lookup tables.

In this work, we present a MATLAB toolbox called the SigTan HDL Coder, which generates synthesizable HDL code to approximate the hyperbolic tangent or the sigmoid function in hardware. This toolbox bridges the gap that exists between the system level design and hardware approximation of the activation functions. Inputs to the toolbox are the maximum acceptable error, the required input range, and the approximation method. Three approximation methods are programmed into the system, which are: straight forward lookup table approximation, range addressable lookup tables, and a hybrid method. The last method uses a piecewise linear approximation as the foundation, and then refines the results by subtracting a predetermined amount of error stored in a lookup table from the initial approximation results.

The rest of this paper is organized as follows. Section 2 briefly explores different methods for approximating the activation functions. Section 3 presents the SigTan HDL Coder toolbox for MATLAB. In section 4, some examples of the results obtained by the SigTan HDL Coder are presented. Finally, section 5 discusses some concluding remarks.

II. A Review of Different Methods for the Approximation of the Activation Functions in Hardware

A. Piecewise Linear Approximations

Piecewise linear approximation uses a series of linear segments to approximate the activation function [5]. The number and locations of these segments are chosen such that error, processing time, and area utilization are minimized. The use of multipliers should be avoided for efficient hardware implementations employing this approximation method, as multipliers are expensive hardware components in terms of area and delay. A piecewise linear approximation of the hyperbolic tangent function with five segments is shown in Fig. 2.

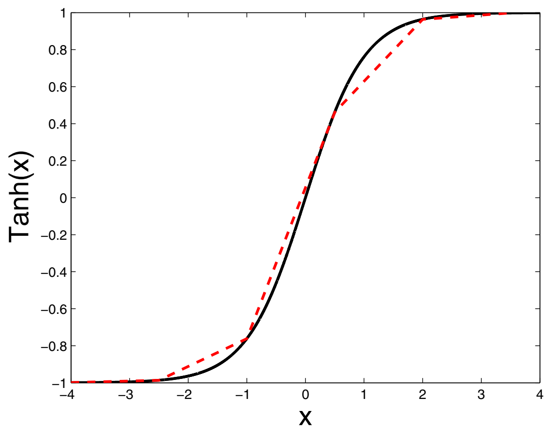


Fig. 2. Piecewise Linear Approximation of Tanh(x) with Five Segments

B. Lookup Table Approximation

With this approach, the function is approximated by a limited number of uniformly distributed points [9], [11]. This method results in a high performance design, however it will require a large amount of memory to store the lookup table (LUT) in hardware. Alternately, a hardware synthesizer can be used to implement the design entirely as a combinational logic circuit. There exist an exponential relationship between the area requirements and precision level using this method, rendering this approach impractical for large table sizes. A lookup table approximation of the sigmoid function with sixteen points is shown in Fig. 3.

C. Range Addressable Lookup Table Approximations

The concept of range addressable lookup tables (RALUTs) was originally proposed in [13] to approximate non-linear discontinuous functions. It shares many aspects with the classic

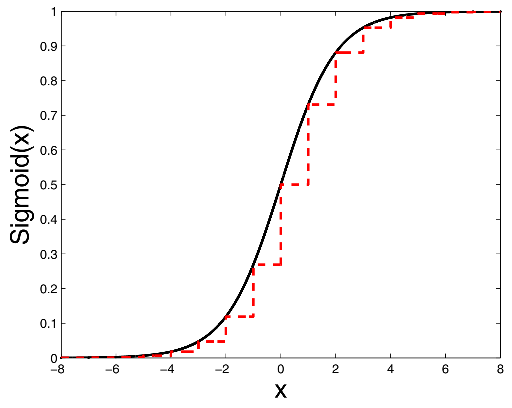


Fig. 3. The Lookup Table Approximation of the Sigmoid Function

LUT with a few notable differences. In LUTs, every data point stored in the table corresponds to a unique address. In RALUTs, every data point corresponds to a range of addresses. This alternate addressing allows for a large reduction in data points in situations where the output remains constant over a range. Examples of this are the hyperbolic tangent and the sigmoid functions, where the output changes only slightly outside the range of $(-2, 2)$. The VLSI approximation of the hyperbolic tangent function using this method is presented in [9]. A range addressable lookup table approximation of the hyperbolic tangent function with seven points is shown in Fig. 4.

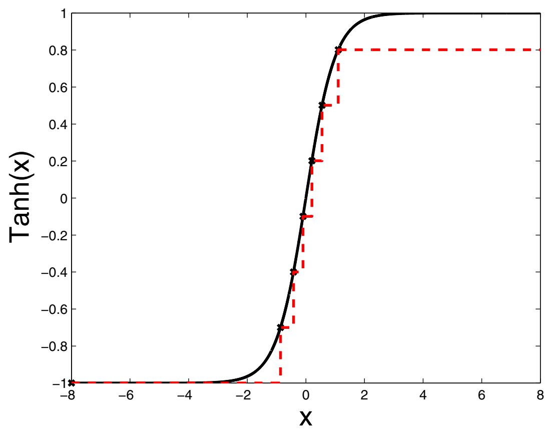


Fig. 4. Range Addressable Lookup Table Approximation of Tanh(x)

D. Hybrid Approximation Methods

Hybrid methods usually combine two or more of the previously mentioned methods to achieve better performance. The hybrid system that is being programmed through the SigTan HDL Coder makes use of the piecewise linear approximation along with the use of RALUT [12]. It initially makes use of a piecewise linear approximation with two segments to roughly approximate the activation function. The linear segments were selected to be $y = x$ for the interval $[0, 1]$, and $y = 1$ outside of that range for the hyperbolic tangent function, while they are $y = \frac{1}{4}x$ for the interval $[0, 2]$, and $y = 1$ outside of that range for the sigmoid function. These segments will be referred to as segment-1 and segment-2 respectively. Note that the symmetry property of the functions was exploited to calculate the output for the negative range. The segments for the piecewise linear approximation of a hyperbolic tangent function are shown in Fig. 5.

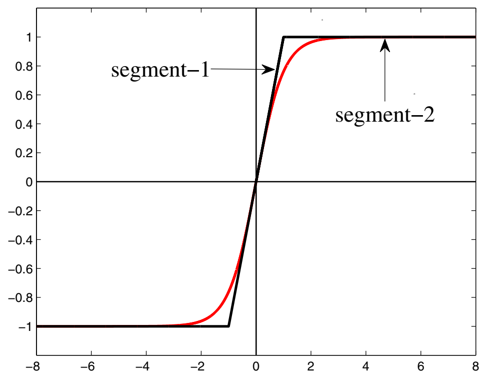


Fig. 5. Segment-1 and Segment-2 of the Piecewise Linear Approximation for Tanh(x)

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.

This 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. Note that for the sigmoid function, calculating x divided by four is done through shifting twice to the right. 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. To further optimize the design a range addressable lookup table was used instead of a classic lookup table. The block diagram of the hybrid system is shown in Fig 6.

III. The SigTan HDL Coder Toolbox for MATLAB

The Graphical User Interface (GUI) for the SigTan HDL Coder is shown in Fig. 7. Regarding the implementation, this program has been developed in MATLAB R2007a on a LINUX operating system. It also supports similar platforms, such as UNIX.

The interface allows the user to enter the maximum allowable approximation error, as well as the input range of interest.

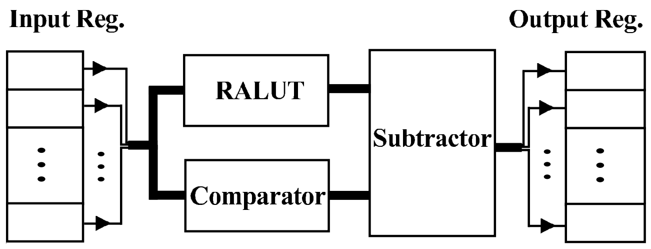
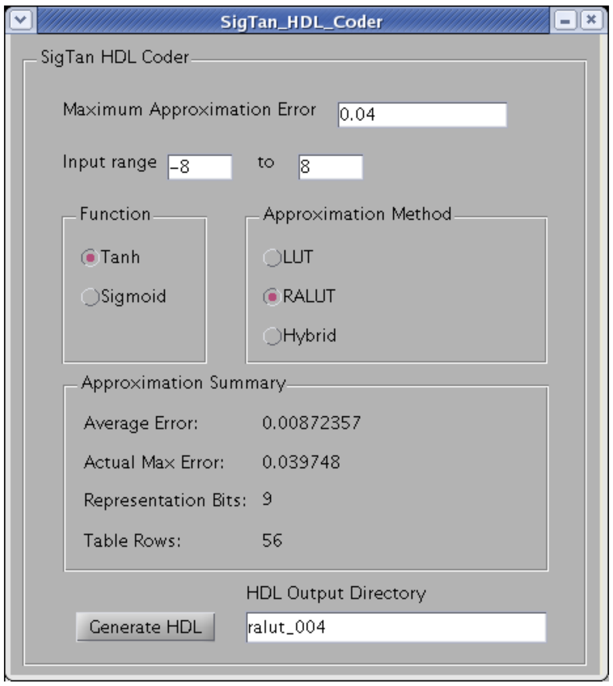


Fig. 6. System Block Diagram for a Hybrid Method

Next, the function and approximation method are chosen. Finally, once the user clicks "Generate HDL", MATLAB code executes, and all the necessary files are created in the chosen directory, ready for a logic compiler tool.

After code execution, important statistics regarding the approximation are displayed: the average and maximum approximation error, as well as the number of representation bits and lookup table rows. The "actual" maximum error is presented because it generally differs slightly from the entered amount due to the fixed-point nature of the table values. This, and the other presented data are made available to the user so that they may quickly adjust design parameters to achieve desirable results.



Maximum Approximation Error 0.04

Input range −8 to 8

Function: [✓ Tanh] Sigmoid

Approximation Method: LUT [✓ RALUT] Hybrid

Average Error:	0.00872357
Actual Max Error:	0.039748
Representation Bits:	9
Table Rows:	56

HDL Output Directory: ralut_004

Fig. 7. Graphical User Interface

IV. Hardware Implementation Examples

Hardware implementation results for different maximum approximation errors are tabulated in table I. Note that the approximation range was selected to be (-8,8) for both functions. All designs were implemented using CMOS $0.18\mu m$ technology from TSMC. The VHDL code was simulated

Architectures	Function	Max-Error	AVG-Error	Area	Delay	Area × Delay
LUT	Tanh	0.5%	0.050%	37647.4 μm^2	2.61 ns	98.26×10^{-12}
RALUT			0.171%	27194.7 μm^2	2.26 ns	61.46×10^{-12}
Hybrid			0.203%	12400.1 μm^2	3.03 ns	37.57×10^{-12}
LUT		2%	0.200%	9045.9 μm^2	2.15 ns	73.44×10^{-12}
RALUT			0.446%	7090.4 μm^2	1.85 ns	13.11×10^{-12}
Hybrid			1.230%	3646.8 μm^2	2.31 ns	8.42×10^{-12}
LUT	Sigmoid	0.5%	0.098%	34159.1 μm^2	2.43 ns	83.01×10^{-12}
RALUT			0.485%	24385.5 μm^2	2.26 ns	55.11×10^{-12}
Hybrid			0.260%	9273.6 μm^2	3.01 ns	27.91×10^{-12}
LUT		2%	0.392%	8911.8 μm^2	2.15 ns	19.16×10^{-12}
RALUT			1.846%	4480.3 μm^2	2.12 ns	9.498×10^{-12}
Hybrid			1.060%	3004.5 μm^2	2.36 ns	7.091×10^{-12}

TABLE I
COMPLEXITY COMPARISON OF DIFFERENT IMPLEMENTATIONS

using Cadence's NCSim to confirm that the architectural code was functioning correctly. Afterwards, the VHDL code was synthesized to a gate-level netlist using Synopsys' Design Compiler. Compilation parameters were always chosen to maximize the operating frequency of the designs.

As can be seen in the table, different area-delay trade-offs can be achieved using the different approximation errors. In this table, lookup table approximations (LUT and RALUT) present faster designs compared to the hybrid solutions at the cost of increased area utilization. Range addressable lookup table implementations are even faster than the regular lookup tables while consuming less area. Note that as the input range becomes smaller and gets closer to the origin, the RALUT implementations become less efficient compared to the lookup table approach. Note that the LUT approach has less average error compared to the RALUT, and is more suitable for FPGA implementations. The hybrid approach always presents the best trade-off for area and delay; the design is much smaller than the lookup table approaches, while possessing slightly increased delay.

V. Conclusions

In this work, a new MATLAB toolbox that improves the hardware implementation of artificial neural networks is proposed. The toolbox generates synthesizable platform-independent HDL code that approximates the hyperbolic tangent or the sigmoid functions in hardware according to user needs. Three different approximation methods are programmed in the toolbox, all of which avoid the use of hardware multipliers for high performance. Different speed/area trade-off designs are easily realized using the SigTan HDL Coder.

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] M.T. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic", IEE Proceedings on Computers and Digital Techniques, vol. 150, no. 6, pp.403 - 411, Nov. 2003.
[8] 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.
[9] K. Leboeuf, A.H. Namin, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient Hardware Implementation of the Hyperbolic Tangent Sigmoid Function", Third International Conference on Convergence and hybrid Information Technology, accepted, November 2008.
[10] H.K. Kwan, "Simple sigmoid-like activation function suitable for digital hardware implementation" Electronic Letters,vol. 28, no. 15, pp. 1379-1380, July 1992 .
[11] 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.
[12] A.H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "High Speed VLSI Implementation of the Hyperbolic Tangent Sigmoid Function ", IEEE International Symposium on Circuits and Systems (ISCAS), accepted, May 2009.
[13] 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.
[14] 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.