A MEMORY BASED APPROACH FOR DIGITAL IMPLEMENTATION OF TANH USING LUT AND RALUT

Samira Sorayassa and Majid Ahmadi

Department of Electrical and Computer Engineering
University of Windsor, Windsor, Ontario, N9B3P4, Canada

ABSTRACT

Tangent Hyperbolic (Tanh) has been used as a preferred activation function in implementing a multi-layer neural network. The differentiability of this function makes it suitable for derivative-based learning algorithm such as error back propagation technique. In this paper two different memory-based techniques for accurate approximation and digital implementation of the Tanh function using Look Up Table (LUT) and Range Addressable Look Up Table (RALUT) are given. A thorough comparative study of the two techniques in terms of their hardware resource usage on FPGA and their accuracies are explained. The schematic of the synthesized design for special cased are given as an example.

KEYWORDS

Tanh Activation function, Tanh Implementation on FPGA, Approximation methods, Lookup Tables (LUT) Range Addressable Lookup Tables (RALUT).

1. INTRODUCTION

Tangent Hyperbolic function has been used as a preferred activation function for multilayer neural network. Eqn. (1) shows the formula for the Tanh:

$$Tanh\ (x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (1)$$

Tanh has two attractive properties:

- Tanh is differentiable which makes it ideal for gradient-based learning algorithms.
- Tanh is varying between +1 and -1 which allows negative values to be generated by it.

Digital implementation of Tanh poses two problems, (a) presence of exponential function in the eqn. (1) and (b) requirement of division. There are many methods in the literature for the approximation and digital implementation of Tanh function. These include Memory-based approaches based on LUT and RALUT [1], [2], Piecewise linear approximation (PWL) [3], Power of 2 based method [4], CORDIC approximation method [5], and Taylor series expansion [6].

David C. Wyld et al. (Eds): SIGV, AI & FL, SESBC, MLAEDU, DSCC, NLPTT, SCOM, SCM, CEEE - 2022
pp. 45-52, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.122204

In this paper a comprehensive study of Memory based techniques for approximation and hardware implementations of Tanh using LUT and RALUT is given. The comparison between LUT and RALUT Techniques include their hardware resource utilizations, as well as their RMSE which is defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(i)_a - f(i)_0)^2} \qquad (2)$$

Where n is the total number of samples in the input range for LUT and RALUT [-4,4] and [-2,2] respectively. $f(i)_a$ and $f(i)_0$ are approximated output and original output of the ith sample respectively.

Organization of this paper is as follow: Section 2 presents approximation of Tanh using LUT and RALUT. Section 3 presents the simulation results using different breaking points and interpolation methods as well as resource usage for each technique. Section 4 represents concluding and remarks.

2. MEMORY-BASED APPROXIMATION OF TANH

In this approach Look Up Table (LUT) as well as Range Addressable Look Up Table (RALUT) have been used as means of approximating Tanh function [1-2].

In this method Tanh is approximated by several equally or non-equally spaced points and their values are stored in LUT or RALUT. It should be noted that the number of breaking points and word length chosen can affect the accuracy of our approximation. In this paper MATLAB toolbox has been used to define the spacing between samples using even spacing, explicit values and even power of 2 spacing. These methods are efficient in terms of speed while require large memory to provide acceptable accuracy. To reduce the size of the memory required in LUT, RALUT has been used where every output represents a range of input addresses. Fig. (1) shows the HDL Coder workflow while Fig. (2) presents the pseudo code for LUT and RALUT based designs.

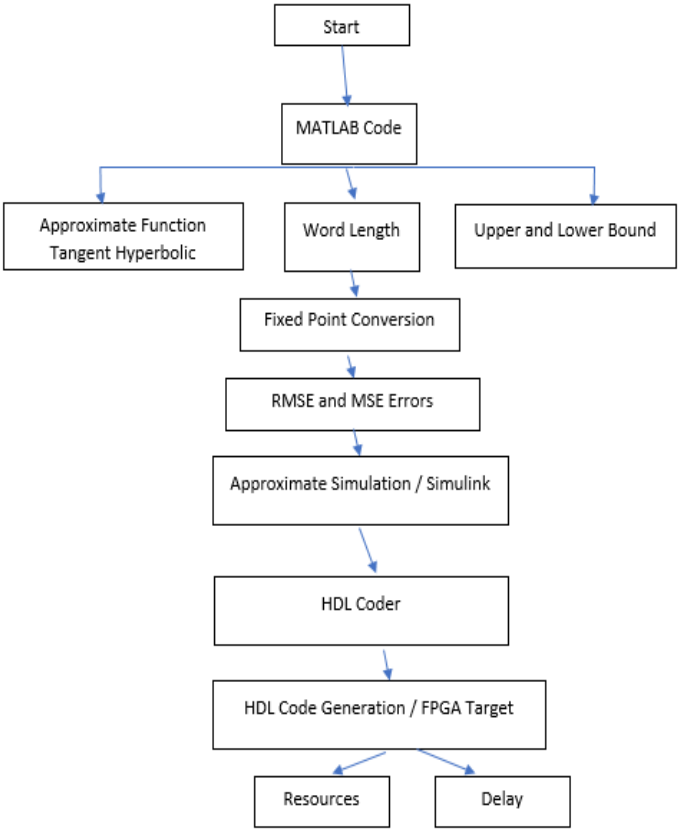

Figure 1. HDL coder workflow

Pseudo code for tangent Hyperbolic LUT & RALUT.

1. Function (CORDIC algorithm) determination.

   A. Function Approximation. Problem=Tangent Hyperbolic

2. Initialization of fixed-point arithmetic Data sample.

   B. Bound= [-4,4] & [-2,2] respectively.

   C. Word-length= [14]

   D. Break point specification= { Evenspacing, Explicitvalues, Evenpower2Spacing

   E. Interpolation method= { Linear interpolation metod, Flat interpolation method

3. Problem solving.

4. Problem comparing

5. Problem approximation.

Figure 2. Pseudo-code LUTs and RALUTs

3. SIMULATIONS AND RESULTS

In this section we present simulation results for approximation and implementation of Tanh function. Implementing MATLAB code requires conversion of floating-point into fixed-point with the optimized bit width suitable for efficient hardware generation. Furthermore, we have introduced the concept of clock and time to find the operation timeline in hardware and created resource-shared architectures to implement some costly operations such as multiplications and division for-loop in processing cycles. Also, relating and mapping procedural construct to the parallel area, for speed optimization and hardware operations is carried out.

The blocking RAM in hardware to map large arrays is also employed. Finally, from Simulink, the VHDL code was generated for the Artix device xa7a100t of the Artix family [7]. We have used Xilinx Vivado 2019 [8] and simulated with ModelSim-Mentor Edition [9] and the Simulation Waveform Editor included in ModelSim-Altra [9]. In our simulation to perform sampling, two types of interpolations have been considered.

The first one is linear interpolation where a line fits between adjacent breaking point and produces the point on the line corresponding to the input. The second one is flat interpolation where the output value corresponding to the breakpoint value that is immediately less than the input value is returned. If no breakpoint value exists below the input value, it returns the lower band value as the input value. Different ranges for the approximation of Tanh have been considered. These include $-4<x<4$ and $-2<x<2$. We have also used different word length and breaking points for our simulations. Table (1) & (2) show the RMSE error for LUT and RALUT approximation as a function of breaking points and number of iterations with linear and flat interpolations. Word length of 14 bits is reported in this study. Tables (3-6) show resource requirements for both LUT and RALUT methods. Figures (3-6) represent the schematic synthesized designed for LUT and RALUT for Linear and Flat interpolations with 10 breaking points for each technique.

Table 1. LUTs –linear& flat interpolation different breakpoints

N of points	N of iterations	RMSE Linear interpolation	RMSE Flat interpolation
50	150	0.0010	0.0210
40	120	0.0016	0.0262
30	90	0.0029	0.0350
20	60	0.0067	0.0528
10	30	0.0275	0.1070

Table 2. RALUTs –linear& flat interpolation different breakpoints

N of points	N of iterations	RMSE Linear interpolation	RMSE Flat interpolation
50	150	4.3285e-04	0.0184
40	120	6.8217e-04	0.0230
30	90	0.0012	0.0308
20	60	0.0028	0.0467
10	30	0.0124	0.0948

Table 3. LUT-50 &10 points-Flat

LUT-50 points-Flat		LUT-10 points-Flat	
Multipliers	0	Multipliers	0
Adders/Subtractors	9	Adders/Subtractors	5
Registers	1	Registers	0
Total 1 Bit Registers	14	Total 1 Bit Registers	0
RAMs	0	RAMs	0
Multiplexers	7	Multiplexers	2
I/O Bits	31	I/O Bits	32
Shifters	0	Shifters	0

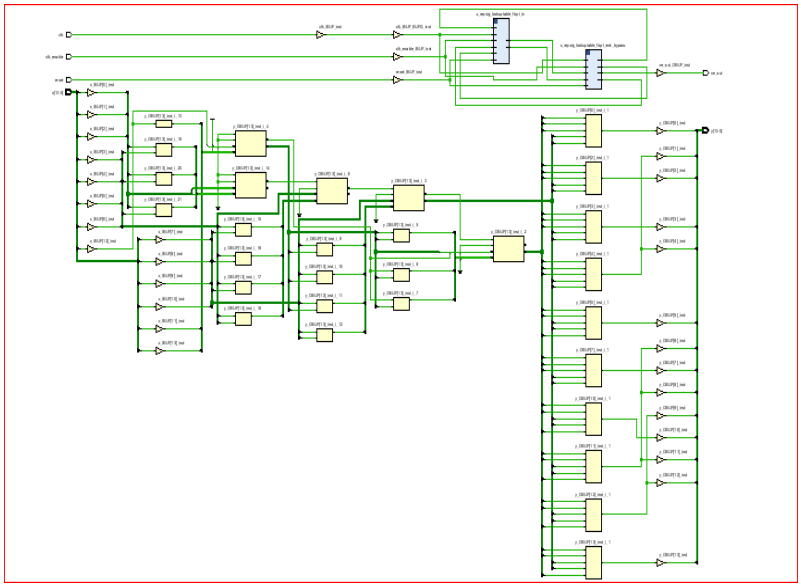

Figure 3. Schematic of synthesized design LUT-10 points-Flat

Table 4. LUT-50 & 10 points-Linear

LUT-50 points-Linear		LUT-10 points-Linear	
Multipliers	2	Multipliers	2
Adders/Subtractors	14	Adders/Subtractors	13
Registers	0	Registers	0
Total 1 Bit Registers	0	Total 1 Bit Registers	0
RAMs	0	RAMs	0
Multiplexers	3	Multiplexers	3
I/O Bits	32	I/O Bits	32
Shifters	0	Shifters	0

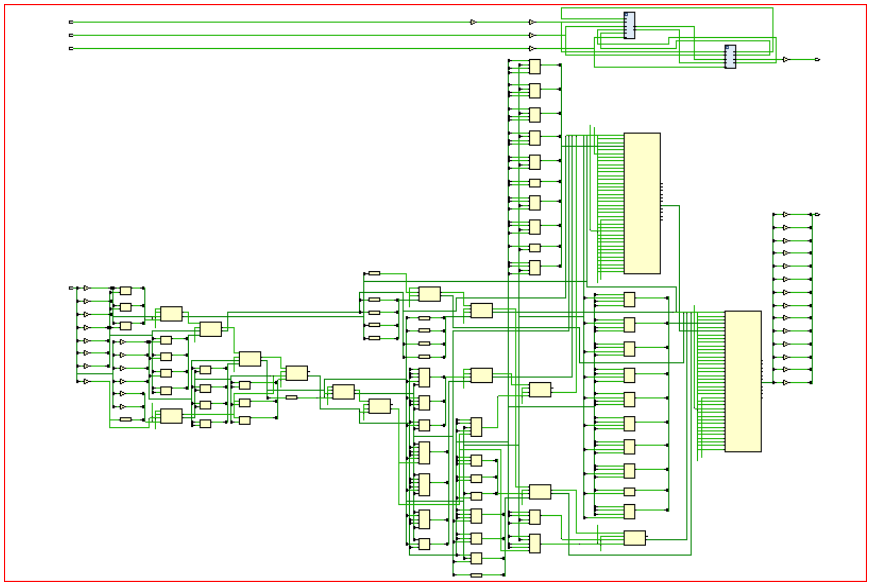

Figure 4. Schematic of synthesized design LUT-10 points- Linear

Table 5. RALUT-50 &10 points-Flat

RALUT-50 points-Flat		RALUT-10 points- Flat	
Multipliers	0	Multipliers	0
Adders/Subtractors	9	Adders/Subtractors	5
Registers	1	Registers	0
Total 1 Bit Registers	14	Total 1 Bit Registers	0
RAMs	0	RAMs	0
Multiplexers	7	Multiplexers	2
I/O Bits	31	I/O Bits	32
Shifters	0	Shifters	0

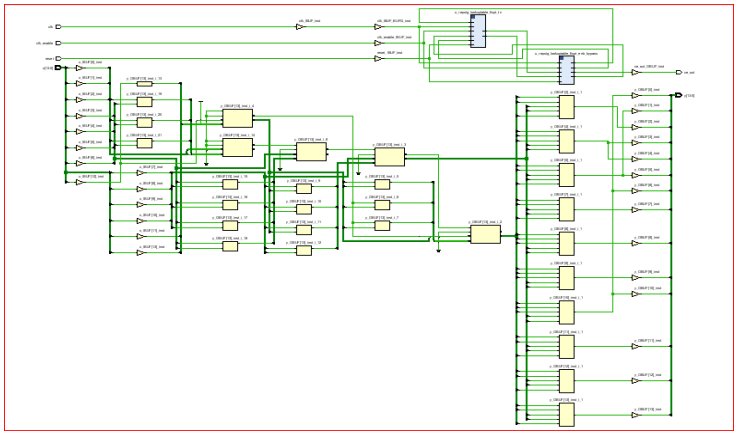

Figure 5. Schematic of synthesized design- RALUT-10 points-Flat

Table 6. RALUT_50 &10 points-Linear

RALUT_50 points-Linear		RALUT-10 points-linear	
Multipliers	1	Multipliers	1
Adders/Subtractors	15	Adders/Subtractors	14
Registers	6	Registers	6
Total 1 Bit Registers	92	Total 1 Bit Registers	92
RAMs	0	RAMs	0
Multiplexers	12	Multiplexers	12
I/O Bits	32	I/O Bits	32
Shifters	0	Shifters	0

4. CONCLUSION

In this paper a memory-based approximation and implementation schemes for Tanh using LUT and RALUT are presented. The choices of breaking points were based both in flat and linear interpolation techniques.

A comparative study of both interpolation techniques for the first time is presented in this paper. It should be noted that nearest interpolation has been used in the literature which requires additional circuits such as comparators to generate output while using flat interpolation is more hardware efficient. Our experiment shows that linear interpolation provides more accurate approximation. Experimental results were conducted with different word length though due to the space limitation only word length of 14 is reported. Resource requirements on FPGA for both LUT and RALUT were reported along with the RMSE of the approximations for both approaches. Furthermore, the effect of iteration number as well the breaking points for the error produced in approximation have been described. Overall, the presented approaches are shown to enjoy high accuracy with ease of implementation on FPGA.

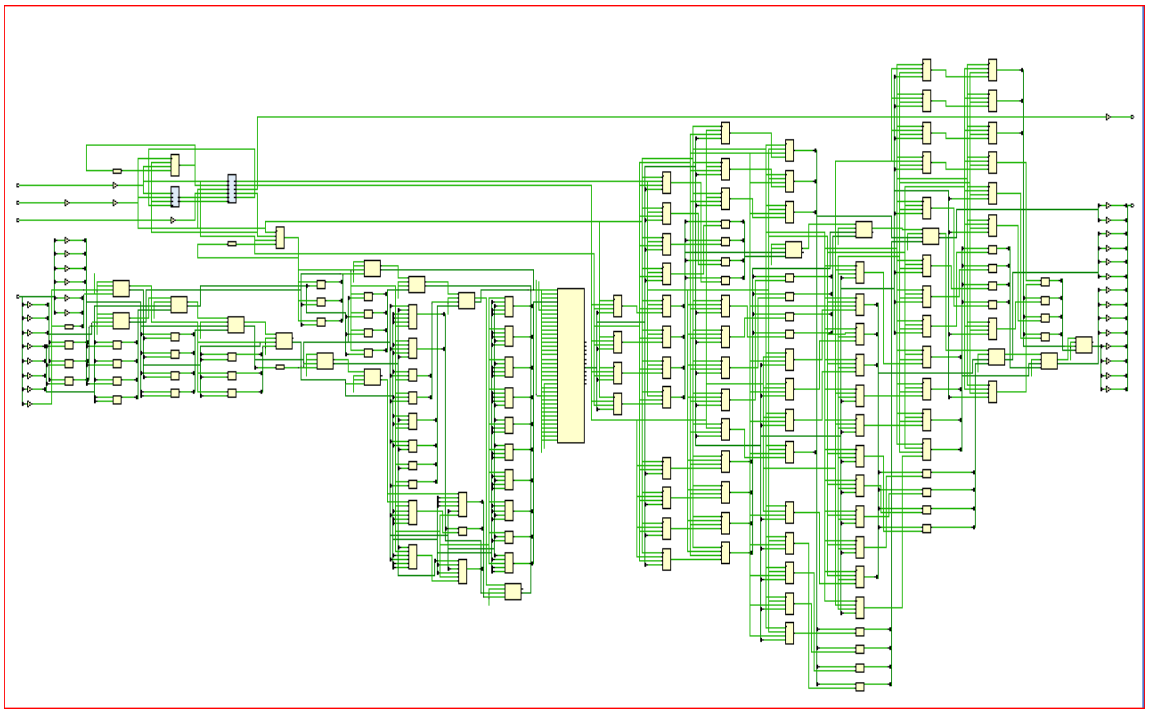

Figure 6. Schematic of synthesized design- RALUT-10 points-linear

REFERENCES

[1] K. Leboeuf, A. H. Namin, H. Wu, M. Ahmadi "High speed VLSI implementation of the hyperbolic tangent sigmoid function" Proceedings of Third International Conference on Convergence and Hybrid Information Technology, 2008, Vol. 1, pp. 1070-1073.

[2] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu and M. Ahmadi "Efficient Hardware Implementation of the Hyperbolic Tangent Sigmoid Function"Proc. of IEEE – ISCAS, May 2009, pp 2117-2120.

[3] A. Armato, L. Fanucci, E.P. Scilingo, D. DeRossi "Low-errn digital hardware implementation of artificial neuro activation functions and their derivatives" Microprocessor and Microsystems, 30 (6), 2011, pp 557-567

[4] S. Gomar, M. Mirhassani, M. Ahmadi "Precise digital implementation of hyperbolic tanh and sigmoid function" Proc of 50th Asidomar Conference on Signals, Systems and Computers, 2016, pp 1586-1589

[5] V. Tiwari, N. Khane "Hardware Implementation of Neural Network with Sigmoid activation functions using CORDIC" Microprocessors and Microsystems 39(b), 2015, pp 373-381

[6] F. Temurtas, A. Gulbug, N Yumusak "A study on neural networks using taylor series expansion of sigmoid activation function" Proc. of Inter. Conf. on Computational Science and its applications, May 2004, pp 389-397.

[7] Artix-7 fpgas data sheet: DC and AC switching….-xilinx.(n.d.). Retrieved November 5,2021 from https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf.

[8] *Vivado ML Overview*. Xilinx. (n.d.). Retrieved November 5, 2021, from https://www.xilinx.com/products/design-tools/vivado.html.

[9] *Intel® FPGA simulation - modelsim*-intel® FPGA*. Intel. (n.d.). Retrieved November 5, 2021, from https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/model-sim.ht

AUTHORS

Bio of Samira Sorayaasa

Received BSc from Jahrom University in 2008, MSc in Management from Shiraz University in 2012, M.A.Sc. from University of Windsor 2022.she is IEEE member.

Bio of Majid Ahmadi

Received BSc from Sharif University in 1970, PhD from Imperial College in 1977.
He has been with the Department of ECE, University of Windsor since 1980.
He holds the rank of Distinguished University Professor. He is FIET and FIEEE.