Efficient implementation of activation functions for LSTM accelerators

Chong, Yi Sheng; Goh, Wang Ling; Ong, Yew-Soon; Nambiar, Vishnu P.; Do, Anh Tuan

2021

Chong, Y. S., Goh, W. L., Ong, Y., Nambiar, V. P. & Do, A. T. (2021). Efficient implementation of activation functions for LSTM accelerators. 2021 29th IFIP International Conference on Very Large Scale Integration (VLSI-SoC).
<https://dx.doi.org/10.1109/VLSI-SoC53125.2021.9606971>

<https://hdl.handle.net/10356/153121>

<https://doi.org/10.1109/VLSI-SoC53125.2021.9606971>

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at:
<https://doi.org/10.1109/VLSI-SoC53125.2021.9606971>.

Downloaded on 26 Apr 2023 16:28:55 SGT

Efficient Implementation of Activation Functions for LSTM accelerators

Yi Sheng Chong*, Wang Ling Goh[†], Yew Soon Ong[‡], Vishnu P. Nambiar[§], Anh Tuan Do[¶]

^{*†}School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore

^{*}Energy Research Institute, Interdisciplinary Graduate Programme, NTU, Singapore

[‡]School of Computer Science and Engineering, NTU, Singapore

^{§¶}Institute of Microelectronics, A*STAR, Singapore

^{*}yisheng002@e.ntu.edu.sg, [†]ewlgoh@ntu.edu.sg, [‡]asysong@ntu.edu.sg,

[§]vishnu_paramasivam@ime.a-star.edu.sg, [¶]doat@ime.a-star.edu.sg

Abstract—Activation functions such as hyperbolic tangent (*tanh*) and logistic sigmoid (*sigmoid*) are critical computing elements in a long short term memory (LSTM) cell and network. These activation functions are non-linear, leading to challenges in their hardware implementations. Area-efficient and high performance hardware implementation of these activation functions thus becomes crucial to allow high throughput in a LSTM accelerator. In this work, we propose an approximation scheme which is suitable for both *tanh* and sigmoid functions. The proposed hardware for sigmoid function is 8.3 times smaller than the state-of-the-art, while for *tanh* function, it is the second smallest design. When applying the approximated *tanh* and sigmoid of 2% error in a LSTM cell computation, its final hidden state and cell state record errors of 3.1% and 5.8% respectively. When the same approximated functions are applied to a single layer LSTM network of 64 hidden nodes, the accuracy drops by 2.8% only. This proposed small yet accurate activation function hardware is promising to be used in Internet of Things (IoT) applications where accuracy can be traded off for ultra-low power consumption.

I. INTRODUCTION

Long short term memory (LSTM) network has recently gained attention for its promising performance in processing time-series and continuous data such as sound classification [1] and speech recognition [2]. To deploy the compute intensive LSTM workload in resource constrained environment, various hardware implementations have been proposed [3], [4], [5].

One of the challenges in implementing LSTM in the ASIC hardware is the implementation of its activation functions. Hyperbolic tangent (*tanh*) and logistic sigmoid (*sigmoid*) are non-linear functions, as shown in Eq. 1 and 2, to enable the LSTM models to create complex mapping. The *tanh* and sigmoid functions are used twice and three times, respectively in one LSTM computation step, as shown in Figure 1, whereas, for example, [4] used a number of activation blocks to realize high computation throughput. Thus, dedicated area-efficient and high performance hardware implementation of these activation functions is highly sought after.

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

Fig. 1. Architecture and data flow of a LSTM cell.

$$\text{sigmoid}(x) = g(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

Traditionally, these non-linear functions are implemented by using look-up table (LUT) [6], range addressable LUT (RALUT) [6], piecewise linear (PWL) method [7] or a hybrid approach [8], [9]. LUT is the most straightforward method to approximate a non-linear function, mapping each input value to an output value. Yet, LUT is not area efficient, especially when large number of input values is required. Hence, [6] proposed RALUT to reduce the LUT size by grouping a set of input values that has similar output value. On the other hand, the PWL method divides a function into segments and these segments are estimated using linear approximation. However, the PWL may not be favourable for certain cases as it involves additions and multiplications that lengthen the data path. Lastly, the hybrid approach allows combining two or more methods in the approximation. For instance, [8] combined linear approximation and LUT for *tanh* implementation. It used linear approximation at small inputs, while using LUT at the inputs that have large output changes.

However, previous hardware implementations focused mainly on the single activation function, whereby [8], [10], [9] proposed solutions for *tanh* activation, while [11] focused on sigmoid implementation. To ease the design flow, this paper proposes a novel scheme to approximate both the sigmoid and *tanh* functions. The scheme exploits the properties in both activation functions, leading to a hybrid implementation of linear approximation and bit-level mapping. Quine–McCluskey algorithm [12] is adopted to minimize the logic usage at the bit-level mapping to reduce the design size.

Fig. 2. Tanh and sigmoid functions with their derivatives.

Fig. 3. Three different regions in the activation function.

II. PROPOSED APPROXIMATION SCHEME

A. Properties of tanh and sigmoid

We leverage on the symmetry of both functions to optimize their hardware implementations. While $\tanh$ is an odd function, sigmoid function has a symmetry point at point $(0, 0.5)$, as shown in Figure 2. Thus we only need to focus on the positive input range. For the negative input, the output values can be easily calculated using Eq. 3.

The positive input range is divided into three regions, namely pass, processing and saturation, as shown in Figure 3. In the pass region, input values are small and thus both functions are approximated linearly using Taylor expansion, as shown in Eq. 4 and 5. In the processing region, a set of input sub-ranges is mapped to a set of output values. Quine-McCluskey algorithm [12] is used to produce minimized Boolean logic that is equivalent to the mapping. In the saturation region, the output is approximated to a constant value which is close to 1.

$$\tanh: f(-x) = -f(x) \quad \text{sigmoid: } 1 - g(x) = g(-x) \quad (3)$$

$$\tanh(x) = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \dots \approx x \quad (4)$$

$$\text{sigmoid}(x) = \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + \frac{x^5}{480} - \dots \approx \frac{1}{2} + \frac{x}{4} \quad (5)$$

B. Data representation

The input and output values are represented in sign-magnitude notation. The magnitude is interpreted in fixed-point format. Given bit width of N , a number is composed of $\langle \text{sign}, N_i, N_f \rangle$. Since the output is very close to 1 at large inputs, the input range is fixed at $(-8, 8)$. Thus, $N_i = 3$ is chosen to represent the largest integer within the range. The

TABLE I
APPROXIMATION OUTCOME FOR TANH AT MAX ERROR, $\varepsilon = 0.02$

Input segment	Sub-range		Value	Region
	r_{begin}	r_{end}		
(0, 0.25)	0	0.25	x	Pass
(0.25, 0.5)	0.25	0.40625	x	Pass
	0.40625	0.4375	0.390625	Processing
	0.4375	0.46875	0.421875	Processing
	0.46875	0.5	0.4375	Processing
(0.5, 1)	0.5	0.53125	0.46875	Processing
	0.53125	0.5625	0.5	Processing
	0.5625	0.59375	0.515625	Processing
	...	...	...	...
(1, 2)	...	...	...	...
	1.9375	2.0	0.96875	Processing
(2, 4)	2.0	2.015625	0.96875	Processing
	2.015625	4	0.984375	Saturation
(4, 8)	4	8	0.984375	Saturation

data resolution depends on the user-specified maximum approximation error, ε , and needs to satisfy Eq. 6. For $\varepsilon = 0.02$, $N_f = 6$, data bit-width, N becomes 10, including the sign bit.

C. Determining region boundaries and output values

The boundaries of the three regions are determined with predefined maximum approximation error, ε . The proposed procedure to determine the boundaries is inspired by [9]. A new method is introduced to maximize the range of pass and saturation region.

The boundary of the pass region is where the maximum absolute difference between the ideal and its linear approximation does not exceed the maximum error ε , as shown in Eq. 7. The approximation equation depends on the target activation function. For $\tanh$, $F_{\text{pass}}(x) = x$, while for sigmoid, $F_{\text{pass}}(x) = 0.25x + 0.5$. Similarly, the boundary of the saturated region is where the maximum absolute difference between the ideal and the approximated constant does not exceed the maximum error ε , as shown in Eq. 8. Thus, the pass region is $[0, x_{\text{pass}}]$ while the saturation region is $[x_{\text{sat}}, 8]$. The processing region is $(x_{\text{pass}}, x_{\text{sat}})$.

To determine the output values of each region, the positive input range is divided into segments according to Eq. 9. An input segment is further divided into sub-ranges whose step size is $\frac{1}{2^s}$, where $0 \leq s < n$. The step size should not exceed the width of the input segment. The output value of a sub-range is the average of the output values of all input values in the sub-range.

For example, $\tanh$ approximation for $\varepsilon = 0.02$ is shown in Table I. It shows the region boundaries as well as the determined value for each segment and sub-range. The step size of segments (0.5, 1), (1, 2) and (2, 4) are $\frac{2}{64}$, $\frac{2}{64}$ and $\frac{2}{64}$ respectively.

$$\tanh: |f(x_1) - f(x_2)| < \varepsilon \quad \text{sigmoid: } |g(x_1) - g(x_2)| < \varepsilon \quad (6)$$

$$|F(x_{\text{pass}}) - F_{\text{pass}}(x_{\text{pass}})| \leq \varepsilon \quad (7)$$

$$|F(x_{\text{sat}}) - F_{\text{sat}}(x_{\text{sat}})| \leq \varepsilon \quad \text{where } F_{\text{sat}} = 1.0 - \varepsilon \quad (8)$$

$$2^n \leq X < 2^{n+1}, \text{ where } 0 \leq n < (N-2), X \text{ is integer} \quad (9)$$

Fig. 4. Proposed hardware architecture.

Fig. 5. Process block for $\tanh$ approximation with $\varepsilon = 0.02$.

Input, a a_2, a_1, a_0	Output, b b_1, b_0
0 0 0	0 1
0 0 1	0 1
0 1 0	1 0
0 1 1	1 0
1 0 0	1 1
1 0 1	1 1
1 1 0	0 0
1 1 1	0 0

Truth table

Fig. 6. An example of optimizing truth table and producing minimized Boolean function using Quine-McCluskey algorithm.

Note that x_1 and x_2 are two consecutive quantized inputs. $F(x)$ is any ideal activation function, $F_{\text{pass}}(x)$ is the approximated linear equation, x_{pass} is the largest possible input which satisfies (7), x_{sat} is the smallest possible input which satisfies (8).

III. HARDWARE IMPLEMENTATION

A. Proposed hardware design

Figure 4(a) shows the block diagram of the proposed hardware implementation. The input and output are of N -bit wide. It contains two main components, namely the approximator and the output selector. The approximator produces an output depending on which region an input falls into. The output bit width of the approximator equals to the fractional bit width, N_f as the activation function output is always within the range of (0, 1). The output selector chooses an output from the approximator and further processes the approximation result depending on the sign bit.

The approximator consists of three parts. Firstly, the pass block design depends on the target activation function. For $\tanh$, the fractional bits of the input is directly passed as output. For sigmoid, the input is multiplied by 0.25 and added 0.5. This is equivalent to right shifting the input by 2 then adding an one at bit ($N_{\text{frac}} - 1$), which saves significant resources by avoiding the expensive multiplier. Secondly, the process block consists of blocks of combinational logic, as shown in Figure 5. Each sub-block implements the truth table that maps an input segment to a set of output values. The truth table is represented in Boolean function. Quine-McCluskey algorithm

TABLE II
DECODER TABLE FOR TANH AT MAX ERROR, $\varepsilon = 0.02$

Region	Sub-range		Decoder output				
	r_{begin}	r_{end}	d_4	d_3	d_2	d_1	d_0
Pass	0	0.40625	0	0	0	0	0
Processing	0.40625	0.5	0	0	0	0	1
	0.5	1	0	0	0	1	0
	1	2	0	0	1	0	0
	2	2.015625	0	1	0	0	0
Saturation	2.015625	8	1	0	0	0	0

Fig. 7. Approximation result of $\tanh$ and sigmoid at $\varepsilon = 0.02$.

[12] is adopted to minimize the Boolean function, that in turn reducing the size of combinational logic. A simple example is shown in Figure 6. The algorithm is preferred over Karnaugh map especially when handling large input width. It also can be easily computed using computers. Thus, we use an open-source Python package called "logicmin" [13] to perform the logic minimization. Thirdly, a constant which represents the value of the saturation region is connected to the multiplexer (MUX).

The output selector consists of a MUX, a decoder and a decision block. The MUX collects all outputs from the approximator. First, the MUX decides its output using the signal from the decoder. Second, the decoder determines which range an input value falls into. The logic mapping in the decoder is also optimized using Quine-McCluskey algorithm [12]. Third, the decision block has different designs for different target activation functions. For $\tanh$, the approximation result is concatenated with the sign bit to produce the final result. For sigmoid, the final result is produced by adding an one to the negated approximation result.

For example, when $\tanh$ function is approximated at $\varepsilon = 0.02$, there are four sub-blocks in the process block. When developing truth table for the input segment (0.25, 0.5), unlike [9], sub-range (0.25, 0.40625) is excluded to reduce the table size. Similarly, for the input segment (2, 4), a truth table is built for the sub-range (2.0, 2.015625), while the rest of the input value in this segment is directly mapped to the saturation value. Finally, a decoder of 5-bit output is needed to cover six input segments, as shown in Table II.

IV. RESULTS AND DISCUSSIONS

The proposed design is synthesized at 65nm. The maximum approximation error, ε is set at 0.02. Therefore, the number representation for both $\tanh$ and sigmoid is 10-bit long with $\langle 1, 3, 6 \rangle$, where approximated $\tanh$ and sigmoid are shown in Figure 7. The approximated values of both functions are

TABLE III
COMPARISON OF TANH AND SIGMOID IMPLEMENTATION WHEN MAX ERROR, $\varepsilon = 0.02$

Function	Design	Tech (nm)	Input range	Max error	Area (μm^2)	Delay (ns)	Normalized area (μm^2)
<i>tanh</i>	Scheme-2 [10]	180	(-8, 8)	0.0220	83559.17	1293	10896.2
	LUT [6]	180	(-8, 8)	0.0180	17864.24	2.45	2329.5
	RALUT [6]	180	(-8, 8)	0.0178	11871.53	2.12	1548.1
	Optimized LUT [8]	90	(-8, 8)	0.0205	780.4	2.82	407.1
	Hybrid [9]	180	(-8, 8)	0.0196	1280.66	2.12	167.0
	Proposed design	65	(-8, 8)	0.0200	220.41	1.69	220.41
Sigmoid	PWL [7]	90	(-8, 8)	0.0076	2024.37	0.98	1055.9
	Proposed design	65	(-8, 8)	0.0198	126.53	2.25	126.5

TABLE IV
APPROXIMATION ERROR VS MSE OF LSTM COMPUTATION RESULT

	Mean squared error (MSE)	
ε	Hidden state	Cell state
0.04	0.051	0.115
0.02	0.031	0.058

Fig. 8. Approximation error vs LSTM network accuracy.

compared with the ideal outputs. The absolute approximation errors are always below $\varepsilon = 0.02$.

Table III compares the synthesized design for both the *tanh* and sigmoid functions with the existing implementations. The proposed *tanh* hardware is at least 1.87 times smaller than those implemented with LUT. Our design uses only combinational logic, avoiding Read-Only-Memory (ROM) or registers which are typically larger in size. Our implementation is 32% larger than [9], yet it is still the second best among the comparisons. On the other hand, the area of the proposed sigmoid hardware 8.3 times smaller than [11]. While, the approximation error of [11] is about 0.7%, our approximation error of 2% is still acceptable [14] when deploying to LSTM edge accelerators, where LSTM accuracy can be traded off with ultra low power and area.

When the ideal activation functions are replaced by the approximated version, there are some errors in the LSTM computation. We evaluate the impact of the approximation error on the LSTM computation. A LSTM cell with hidden size of 128 is fed with random input size of 128 and 64 time steps. This repeats for 32 times. The mean squared error (MSE) of the final hidden state and cell state for both cases are recorded in Table IV. By setting ε to 0.02, the LSTM computation result has errors of 3.1% and 5.8% at the final hidden state and cell state, respectively. This error is still acceptable, because when it works as a LSTM network the small error may cause negligible accuracy drop, due to the precision tolerant properties of the neural network algorithm [14].

We also analyze how the approximated functions affect the LSTM network accuracy. A single layer LSTM network

with 64 hidden nodes for keyword spotting task is trained to classify 12 classes [15]. The network uses the publicly available Google Speech Commands dataset [16], attaining accuracy of 92.4%. When the exact activation functions in the LSTM network are replaced by the approximated version, a small accuracy drop of 2.8% is shown in Figure 8 when the maximum approximation error is 0.02. The network accuracy drops by 14% when the approximation error is 0.04. This analysis suggests that to maintain high accuracy, activation functions with small approximation error are required.

V. CONCLUSION

This work proposes an approximation scheme for both the sigmoid and *tanh* function. The scheme utilizes the properties found in both functions and divides the positive input range into three different regions. The hardware is specially designed to fit the needs from different activation functions. Using Quine-McCluskey algorithm, we minimize the logic complexity when designing combinational logic in the proposed design, thus reducing the overall design area. The proposed sigmoid hardware has 8.3 times smaller area than the state-of-the-art, while for the proposed *tanh* hardware is the second smallest design. Despite that there is an error when approximating the activation function, the LSTM computation result has negligible MSE errors of 3.1-5.8%. Besides, when introducing the approximated activation functions in a single layer LSTM network, only small accuracy drop of 2.8% is observed given that the maximum approximation error of 0.02. Therefore, our proposed small and accurate activation hardware is promising for the edge LSTM accelerators where accuracy can be traded off for ultra-low power consumption and area.

ACKNOWLEDGMENT

We thank the Programmatic grant no. A1687b0033, Singapore RIE 2020, AME domain.

REFERENCES

- [1] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks," in *ICASSP 2015*, vol. 2015-August, aug 2015, pp. 4580–4584.
- [2] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *31st International Conference on Machine Learning, ICML 2014*, vol. 5, 2014, pp. 3771–3779.
- [3] N. M. Rezk, M. Purnaprajna, T. Nordstrom, and Z. Ul-Abdin, "Recurrent Neural Networks: An Embedded Computing Perspective," *IEEE Access*, vol. 8, pp. 57 967–57 996, 2020.
- [4] M. Wang *et al.*, "E-LSTM: An Efficient Hardware Architecture for Long Short-Term Memory," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 280–291, jun 2019.

- [5] E. Azari and S. Vrudhula, "ELSA: A throughput-optimized design of an LSTM accelerator for energy-constrained devices," *ACM Transactions on Embedded Computing Systems*, vol. 19, no. 1, 2020.
- [6] K. Leboeuf *et al.*, "High speed VLSI implementation of the hyperbolic tangent sigmoid function," in *Proceedings-3rd International Conference on Convergence and Hybrid Information Technology, ICCIT 2008*, vol. 1, 2008, pp. 1070–1073.
- [7] H. Sun *et al.*, "A Universal Method of Linear Approximation with Controllable Error for the Efficient Implementation of Transcendental Functions," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 1, pp. 177–188, jan 2020.
- [8] P. K. Meher, "An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks," in *Proceedings of the 2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip, VLSI-SoC 2010*, 2010, pp. 91–95.
- [9] B. Zamanlooy and M. Mirhassani, "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 39–48, jan 2014.
- [10] C. W. Lin and J. S. Wang, "A digital circuit design of hyperbolic tangent sigmoid function for neural networks," in *Proceedings - IEEE International Symposium on Circuits and Systems*, 2008, pp. 856–859.
- [11] Z. Qin *et al.*, "A Novel Approximation Methodology and Its Efficient VLSI Implementation for the Sigmoid Function," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3422–3426, jun 2020.
- [12] E. J. McCluskey, "Minimization of boolean functions," *The Bell System Technical Journal*, vol. 35, no. 6, pp. 1417–1444, 1956.
- [13] "dreylago/logicmin." [Online]. Available: <https://github.com/dreylago/logicmin>
- [14] T. Yang *et al.*, "Design Space Exploration of Neural Network Activation Function Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1974–1978, 2019.
- [15] M. Sun *et al.*, "Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting," in *2016 IEEE Workshop on Spoken Language Technology, SLT 2016 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., feb 2016, pp. 474–480.
- [16] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv*, apr 2018. [Online]. Available: <http://arxiv.org/abs/1804.03209>