A Novel Approximation Methodology and Its Efficient VLSI Implementation for the Sigmoid Function

Zidi Qin, Yuou Qiu, Huaqing Sun, Zhonghai Lu, *Senior Member, IEEE*, Zhongfeng Wang, *Fellow, IEEE*, Qinghong Shen, and Hongbing Pan

***Abstract*—In this brief, a novel approximation method and its optimized hardware implementation are proposed for the sigmoid function used in Deep Neural Networks (DNNs). Based on piecewise approximation and truncated Taylor series expansion, the proposed method achieves very good approximation with low complexity while exploiting data representation with powers of two. In addition, by analyzing gradients of the sigmoid function, a small trick is introduced to improve the approximation precision. Furthermore, to reduce the hardware complexity and shorten the critical path, sampled values of the function are generated with simple logical-mapping. It is shown that the proposed approximation schemes can be implemented with purely combinational logic and the sigmoid function can be computed in one clock cycle. The experimental results demonstrate that the mean absolute errors are at the order of 1×10^{-3}. Compared with prior arts, the new design can obtain significant improvement in critical path with comparable performance.**

***Index Terms*—Sigmoid function, approximation method, VLSI architecture.**

I. Introduction

NONLINEAR functions are widely used in many fields including signal processing applications and deep neural networks (DNNs). As a typical nonlinear function, the sigmoid function is commonly used as the activation function in DNNs.

In emerging DNN accelerators, computing units of activation functions are important elements. The implementation of activation-function units can influence the performance, area and power of a DNN accelerator, especially when many units work in parallel. However, there are complex exponentiation and division operations in the computation of the sigmoid function, so it is inefficient to implement the function directly in digital hardware. Therefore, multiple approximation methods have been proposed for the efficient implementation of the sigmoid function.

The look-up-table (LUT)-based method is a straight-forward way to approximate nonlinear functions [1], [2]. The LUT method uses LUTs or ROMs to store the values which are sampled from the nonlinear functions. However, approximating a function with higher precision leads to significant increase in area, because large numbers of LUTs are required to store the sampled values.

Another commonly used approach is piecewise linear (PWL) approximations [3]–[8]. In PWL approximations, a nonlinear function is divided into several segments and a linear function is used to fit the curve in each segment. In the typical architecture for PWL, a multiplier, an adder and LUTs are required [3]. To simplify the hardware implementation of PWL approximation, the PLAN method [7], [8] uses six special segments to approximate the sigmoid function, and replaces the multiplications with shift operations. However, this kind of simplified methods bring too much precision loss for the approximation.

Other works employ piecewise second order approximation [9] and Taylor's theorem [10] for the implementation of sigmoid. However, the hardware implementations of these methods require more hardware resources and longer latency than PWL approximation.

In [11], the calculation of the sigmoid function is based on an approximation formula of the exponential function, requiring complex division operations. This results in large hardware complexity and long computing latency of the circuits in [11].

Bit-level mapping method is presented in [12] to implement the sigmoid function with simple combinational circuits. However, this method is suitable for cases when the input range or the input bit-width is small.

This brief presents a novel approximation method and corresponding hardware implementation for the sigmoid function. The contribution of this brief is twofold. First, a special segmentation scheme is applied for the proposed piecewise approximation method. An approximation equation based on Taylor series expansion is further derived for the sigmoid function. Second, taking advantage of special properties of the sigmoid function, the hardware implementation is optimized by replacing the multiplications with shift operations and generating the sampled values with simple logical-mapping. With

Manuscript received April 8, 2020; revised May 19, 2020; accepted May 27, 2020. Date of publication June 2, 2020; date of current version November 24, 2020. This work was supported by the National Nature Science Foundation of China under Grant 61376075 and Grant 41412020201. This brief was recommended by Associate Editor X. Li. *(Corresponding author: Hongbing Pan.)*

Zidi Qin, Yuou Qiu, Huaqing Sun, Zhongfeng Wang, Qinghong Shen, and Hongbing Pan are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China (e-mail: qinzidi@smail.nju.edu.cn; dorothy.q@foxmail.com; amshq@foxmail.com; zfwang@nju.edu.cn; qhshen@nju.edu.cn; phb@nju.edu.cn).

Zhonghai Lu is with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden (e-mail: zhonghai@kth.se).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2020.2999458

above optimizations, high-precision approximation of the sigmoid function can be implemented with purely combinational logic circuits.

This brief is organized as follows. Section II introduces the proposed approximation method. The hardware architecture of the sigmoid calculator is described in Section III. Section IV provides experimental results and comparisons with previous works. Finally, we conclude this brief in Section V.

II. The Proposed Approximation Method

In this section, the approximation method and optimized implementation schemes for the sigmoid function are presented.

A. *Characteristics of the Sigmoid Function*

A general expression of the sigmoid function is as follows:

$$y = f(x) = \frac{1}{1+e^{-x}}. \tag{1}$$

When $x > 8$, the sigmoid function converges to 1. Similarly, when $x < -8$, it converges to 0. Thus, we approximate its value in the range $(-8, 8)$. As the sigmoid function has a symmetry point at $(0, 0.5)$, we only consider the positive half of the function, and the other part can be computed by

$$y_{x<0} = 1 - y_{x>0}. \tag{2}$$

When input values are integer multiples of ln2, Eq. (1) can be transformed to

$$f^*(n) = f(n\ln 2) = \frac{1}{1+2^{-n}}, \tag{3}$$

where n is an integer. Notably, as proved in [13], for $\forall n \in \mathbb{Z}\backslash\{0\}$, the value of $f^*(n)$ can be represented by a simple periodic binary number:

$$0.(\bar{s_1}\bar{s_2}\cdots\bar{s_n}s_{n+1}s_{n+2}s_{2n}), \tag{4}$$

where $s = s_1 = s_2 = \cdots = s_{2n}$ is the sign of n, and the round parentheses are used to show that a periodic number is dealt with. For example, if $n = 2$, then $s = 0$ and $f^*(n) = 0.110011001100\cdots$; If $n = -2$, then $s = 1$ and $f^*(n) = 0.001100110011\cdots$.

B. *The Basic Approximation Method*

As shown in Fig. 1, we propose a novel segmentation scheme, where the input range $[0, 8)$ is partitioned into several segments by the length of ln2. Thus, the interval $[0, 8)$ can be divided into 12 segments. Each sub-interval in $[0, 11\ln 2)$ is denoted by $[n\ln 2, (n+1)\ln 2)$, where n is a positive integer and $n \in [0, 12)$.

According to the Taylor series expansion, $f(x)$ can be expanded at $x = x_0$ as follows:

$$\begin{aligned} f(x) &= f(x_0 + \Delta x) \\ &= f(x_0) + f'(x_0)\Delta x + \frac{1}{2!}f''(x_0)(\Delta x)^2 + \cdots \\ &\quad + \frac{1}{n!}f^n(x_0)(\Delta x)^n. \end{aligned} \tag{5}$$

When Δx is small, the high order terms in Eq. (5) can be ignored. Thus, we only reserve the first-order term in Eq. (5) and set x_0 as $n\ln 2$. Then $f(x)$ can be expressed by

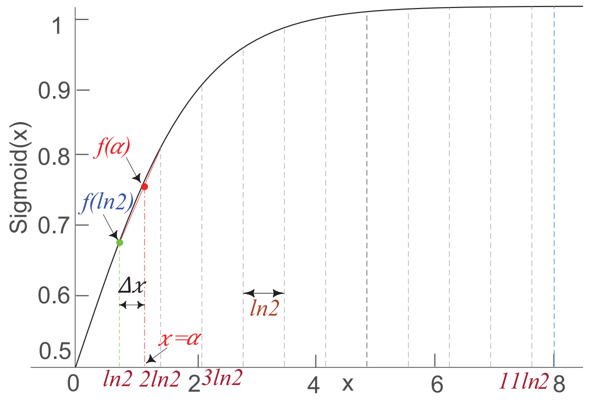

Fig. 1. Proposed segmentation scheme for the sigmoid function.

$$f(x) = f(n\ln 2) + f'(n\ln 2)\Delta x. \tag{6}$$

According to Eq. (6), in each sub-interval, the function can be approximated with a linear segment.

As shown in Fig. 1, for an arbitrary input, α, the value of n denotes the sub-interval in which α is located. The corresponding value of n can be determined by the following equation:

$$n = \lfloor \alpha/\ln 2 \rfloor, \tag{7}$$

where n is the integer part of $\alpha/\ln 2$. Then the value of $f(\alpha)$ can be approximated by

$$f(\alpha) = f(n\ln 2) + f'(n\ln 2)\Delta x. \tag{8}$$

To compute $f(\alpha)$, the value of Δx and $f'(n\ln 2)$ should be further determined.

In the range $[n\ln 2, (n+1)\ln 2)$, we approximate the value of $f'(n\ln 2)$ by the following equation:

$$f'(n\ln 2) = \frac{f((n+1)\ln 2) - f(n\ln 2)}{\ln 2}. \tag{9}$$

Furthermore, as $\Delta x = \alpha - n\ln 2$, Eq. (8) can be transformed into

$$\begin{aligned} f(\alpha) &= f(n\ln 2) + \frac{f((n+1)\ln 2) - f(n\ln 2)}{\ln 2} \times (\alpha - n\ln 2) \\ &= f(n\ln 2) + (f((n+1)\ln 2) - f(n\ln 2))(\frac{\alpha}{\ln 2} - n). \end{aligned} \tag{10}$$

According to Eq. (10), we denote that

$$\lambda = f(n\ln 2), \tag{11}$$

$$\mu = f((n+1)\ln 2) - f(n\ln 2), \tag{12}$$

and

$$\phi = \frac{\alpha}{\ln 2} - n, \tag{13}$$

where ϕ is the decimal part of $\alpha/\ln 2$. Thus, we can get

$$f(\alpha) = \lambda + \mu \times \phi. \tag{14}$$

Eq. (14) is the proposed basic approximation equation for the sigmoid function.

C. *Scheme I: Hardware Complexity Optimization*

In this section, an optimized approximation scheme denoted by Scheme I is introduced.

TABLE I
THE VALUES OF λ, μ, m_1 AND m_2 IN DIFFERENT SUB-INTERVALS

sub-intervals	λ**(binary)**	μ**(binary)**	m_1, m_2
$(0, \ln 2]$	0.100000000000	0.001010101010	$-3, -5$
$(\ln 2, 2\ln 2]$	0.101010101010	0.001000100010	$-3, -7$
$(2\ln 2, 3\ln 2]$	0.110011001100	0.000101101100	$-4, -5$
$(3\ln 2, 4\ln 2]$	0.111000111000	0.000011010111	$-5, -6$
$(4\ln 2, 5\ln 2]$	0.11110000111	0.000001110100	$-5, 0$
$(5\ln 2, 6\ln 2]$	0.111110000011	0.000001000011	$-6, 0$
$(6\ln 2, 7\ln 2]$	0.111111000000	0.000000100000	$-7, 0$
$(7\ln 2, 8\ln 2]$	0.111111100000	0.000000010000	$-8, 0$
$(8\ln 2, 9\ln 2]$	0.111111110000	0.000000001000	$-9, 0$
$(9\ln 2, 10\ln 2]$	0.111111111000	0.000000000100	$-10, 0$
$(10\ln 2, 11\ln 2]$	0.111111111100	0.000000000010	$-11, 0$
$(11\ln 2, 8)$	0.111111111110	0.000000000001	$-12, 0$

The implementation of Eq. (14) requires a multiplier, which has large hardware complexity and long latency. Plus, λ and μ in different subintervals are constant numbers, which are shown in Table I. A direct way is to store λ and μ in LUTs or ROMs. Because of the special segmentation scheme, the value of λ has special regularity, which is introduced in Eq. (4). As we found, the value of μ also has special properties. As shown in Table I, the value of μ is the power of two, when $n > 5$. For other sub-intervals, the values of μ cannot be approximated directly with powers of two. To further decrease the hardware complexity, we propose to approximate the value of μ with

$$\mu' = 2^{m_1} + 2^{m_2}, \tag{15}$$

where m_1 and m_2 are integers. Using this scheme, Eq. (14) can be transformed into

$$\begin{aligned} f(\alpha) &= \lambda + \mu' \times \phi \\ &= \lambda + \phi >> |m_1| + \phi >> |m_2|. \end{aligned} \tag{16}$$

Thus, the sigmoid function can be approximated with Eq. (16), where only additions and shift operations are required.

D. Scheme II: Further Optimization for Algorithm Precision

In this section, we further propose a scheme denoted by Scheme II for improving approximation precision. The scheme is presented as follows:

$$f(\alpha) = \begin{cases} x >> 2 + 0.5, & n = 0, \\ \lambda + \phi >> |m_1| + \phi >> |m_2|, & n > 0. \end{cases} \tag{17}$$

In Section II-C, the value of μ in range $[0, \ln 2)$ is replaced by $\mu' = 2^{-3} + 2^{-5} = 0.15625$ to reduce computation complexity. However, the approximation in $[0, \ln 2)$ brings relative larger approximation error compared with other segments. When $x = 0$, the value of sigmoid function's derivative is 0.25 and we found $y = 0.25x + 0.5$ can fit the curve in $[0, \ln 2)$ with lower approximation errors. As a result, this trick is adopted in Scheme II for improving the approximation precision.

III. HARDWARE ARCHITECTURE OF THE SIGMOID FUNCTION

A. Hardware Architecture of Scheme I

The overall architecture of the sigmoid calculator is shown in Fig. 2, and mainly consists of complement unit (Com Unit), constant multiplier (CM), special number generated unit (SGU), shift unit and a carry-save adder. The input x is in the range $(-8, 8)$, so 3 bits are enough for the integer part of x, and the decimal part of x is an 8-bit number. In Fig. 2, the signal *Sign* is the sign bit of x. According to the symmetry characteristic of the sigmoid, if x is a negative number, the Com Unit is used to get the sign-and-magnitude of x. Next, according to the sign of x, a multiplexer (MUX) is used to choose between p and x. Then outputs of the MUX denoted by x_i are sent to the CM to compute $n = x_i \times (1/\ln 2)$. Notably, we use the algorithm strength reduction strategy proposed in [14] and carry-save technique to optimize the critical path of the CM.

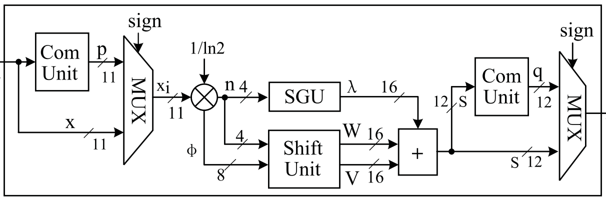

Fig. 2. Top architecture of the sigmoid function calculator.

TABLE II
COMPUTATION EQUATIONS OF EACH BIT OF λ

λ	equation	λ	equation
λ_{15}	$= 1$	λ_7	$= \bar{n}_2(n_1 + n_0) + n_2\bar{n}_1\bar{n}_0$
λ_{14}	$= n_3 + n_2 + n_1$	λ_6	$= n_3 n_1 + \bar{n}_2 n_1 \bar{n}_0 + n_2\bar{n}_1\bar{n}_0$
λ_{13}	$= n_3 + n_2 + n_0$	λ_5	$= n_3 n_1 n_0 + n_2\bar{n}_1 + \bar{n}_3\bar{n}_1 n_0$
λ_{12}	$= n_3 + n_2$	λ_4	$= n_2\bar{n}_1$
λ_{11}	$= n_3 + n_1\bar{n}_0 + \bar{n}_1 n_0 + n_2 n_1 n_0$	λ_3	$= \bar{n}_3\bar{n}_1 n_0 + \bar{n}_3 n_1 \bar{n}_0 + \bar{n}_3\bar{n}_2 n_1$
λ_{10}	$= n_3 + n_2 n_1 + n_1\bar{n}_0$	λ_2	$= n_2\bar{n}_1 n_0 + \bar{n}_3 n_1\bar{n}_0 + \bar{n}_3\bar{n}_2 n_1$
λ_9	$= n_3 + n_1 n_0 + \bar{n}_2 n_0$	λ_1	$= \bar{n}_3 n_0 + \bar{n}_3 n_2 n_1$
λ_8	$= n_3 + \bar{n}_2 n_1 n_0$	λ_0	$= n_2 n_1$

Taking advantage of the regularity of periodic binary numbers, each bit of λ can be computed with a simple logic expression. The expression is based on bits of n, where n_3, n_2, n_1 and n_0 denote the four bits of $n[3{:}0]$ and n_3 is the most significant bit (MSB). We list the computation equations of each bit of λ in Table II. It can be seen that λ can be generated by using several AND gates, OR gates and NOT gates.

The Shift Unit is used to generate $W = \phi >> m_1$ and $V = \phi >> m_2$. This unit is specially designed and two shift operations are combined in one unit to save resources.

Finally, the additions of λ, W and V are simplified with carry-save technique to reduce operation time and hardware complexity. The results of the carry-save adder refer to the function value of a positive input. If x is a negative number, $f(x) = 1 - S$ will be further computed by the Com Unit as the final outputs.

With above optimization techniques, the sigmoid function can be implemented with purely combinational logic.

B. Hardware Architecture of Scheme II

The corresponding hardware architecture for Scheme II is shown in Fig. 3. A multiplexer is used after the constant multiplier to choose between x_i and ϕ. If $n = 0$, x_i is sent to the Modified Shift Unit (M-SU) to perform the shift operations.

TABLE III
COMPARISONS OF HARDWARE IMPLEMENTATION RESULTS WITH OTHER WORKS (90 nm)

Work	**Clock Frequency**	**Area** (μm^2)	**Delay** (ns)	**Area × Delay** (μm^2 × ns)	**Power** (μW)	**Power × Delay** (μW × ns)
LUT [1]	-	4466.06^b	1.23^b	5470.92	-	-
RALUT [1]	-	2967.88^b	1.06^b	3145.96	-	-
Typical PWL method [3]	1 GHz	4634.54^a	1.38^a	6417.06	2104.62^a	2904.37
PLAN [7]	-	336.81^b	1.86^b	624.78	134.10^b	249.43
Proposed Scheme I	1 GHz	1684.27	0.98	1650.58	519.52	509.13
Proposed Scheme II	1 GHz	2024.37	0.98	1983.88	667.00	653.66

[a] This is a scaled result of from 65 nm technology.
[b] This is a scaled result of from 0.18 μm technology.

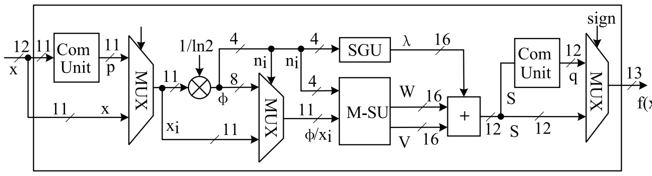

Fig. 3. Architecture of the scheme II.

TABLE IV
COMPARISONS OF APPROXIMATION PRECISION WITH OTHER WORKS

Work	Input range	E_{max}	E_{ave}
LUT [1]	$(-8, 8)$	0.0180	-
RALUT [1]	$(-8, 8)$	0.0178	-
Typical PWL method [3]	$(-8, 8)$	0.0010	0.00055
PLAN [7]	$(-8, 8)$	0.0189	0.0059
Proposed Scheme I	$(-8, 8)$	0.0114	0.0018
Proposed Scheme II	$(-8, 8)$	0.0076	0.0016

The shift operations differ from the shift unit of Scheme I. When $\lambda = 0$, the corresponding shift bits of m_1 and m_2 are 2 and 0, respectively. The other modules in Fig. 3 are the same as the original design and the details have been introduced above.

IV. EXPERIMENT RESULTS AND COMPARISONS

A. Approximation Performance Analysis and Comparisons

1) Metrics of Approximation Precision: As floating point arithmetic is not efficient for hardware implementation, fixed-point notation is commonly used. Assume that the input x is represented by a fixed-point number, whose integer part has P bits and decimal part has Q bits. Assume that $f(x)$ is approximated by a function $g(x)$ in the interval $x \in (a, b)$. Then there are $N = 2^{(P+Q)}$ sampled points in the interval. To evaluate the precision of an approximation method, the average errors (E_{ave}) and maximum absolute errors (E_{max}) are usually used. E_{ave} and E_{max} are defined as follows:

$$\begin{cases} E_{ave} = \frac{\sum_{i=0}^{N-1} |g(x_i) - f(x_i)|}{N}, \\ E_{max} = max(|g(x_i) - f(x_i)|). \end{cases} \tag{18}$$

2) Approximation Precision Analysis: In hardware implementation, the parameters are all fixed-point numbers, so the hardware truncation errors should be taken into consideration. In our simulations, input value x is a 12-bit number, where 1 bit is sign bit, 3 bits are integer bits and 8 bits are decimal bits. The bit-widths of λ and ϕ are both 16 bits. The output value is a 12-bit number. Plus, ln2 is approximated with 1.4375 and represented by a 4-bit number.

MATLAB is used to simulate the approximation Precision. We first simulate the basic approximation method as a baseline. As shown in Fig. 4(a), the approximated curve of Eq. (14) and the approximation errors magnified by 50× are presented. The values of E_{max} and E_{ave} are 0.0066 and 0.0017, respectively. Fig. 4(b) shows the approximated curve of Scheme I. The E_{max} of scheme I is 0.0114 and E_{ave} is 0.0018. From the curve of approximation error in Fig. 4(b), it can be found that the E_{max} locates in the interval of $x \in [0, \ln 2)$. As shown in Fig. 4(c), using scheme II, the approximation precision in $x \in [0, \ln 2)$ is obviously improved, where the maximum approximation error can be reduced from 0.0114 to 0.0078.

B. Hardware Implementation Results and Comparisons

The hardware architectures have been implemented in Verilog HDL and synthesized using the Synopsys Design Compiler (DC) under the SMIC 90 nm technology. A frequency of 1 GHz has been achieved. Synthesis results of the circuits and comparisons with previous works are presented in Table III. The synthesis results of the LUT method, the RALUT method and the PLAN method are obtained from [15]. In Table IV, we compare approximation precision with other works.

Compared with the LUT and RALUT methods, the E_{max} of our method can be improved by 2.37× and 2.34×, respectively. Achieving higher approximation precision, the proposed method employs special properties of the sigmoid function so that no sampled values are stored in LUTs. However, the LUT and RALUT methods require to store many sampled values, leading to significantly more area and up to 64% lower area efficiency than the proposed method.

The typical PWL-based method [3] requires multipliers, LUTs, and complex designs to choose the range of the input, which leads to larger hardware complexity than the proposed method. Thus, the proposed circuits occupied 56% less area than the method in [3]. For the circuits in [3], the critical path contains a subtractor, a multiplexer, an LUT and a multiplier. In the proposed method, however, several optimized schemes are adopted to remove the multipliers and LUTs. With the optimizations, the critical path is reduced by 23% and mainly contains a constant multiplier, a carry-save adder, a shift unit and three multiplexers. More importantly, the proposed circuits

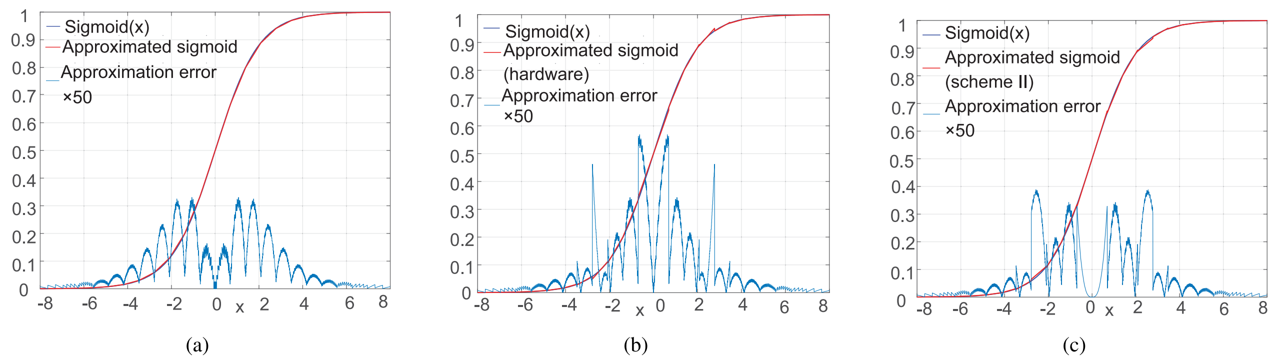

Fig. 4. Approximation errors of different schemes. (a) Original approximation function curve (b) Scheme I: approximation function with hardware optimization (c) Scheme II: improved scheme for precision.

are purely combinational logic, but the PWL-based method in [3] contains one delay element, requiring longer latency than our method. In addition, the power of the proposed circuits can be reduced by 68%. In terms of precision, the method in [3] achieves higher precision by using a non-uniform piecewise scheme. In contrast, this brief adopts a uniform piecewise scheme to employ the special properties of the sigmoid function. Although this piecewise scheme brings some precision loss, the hardware complexity and critical path are significantly reduced.

The PLAN method uses six special segments to approximate the sigmoid function in range $[-5, 5]$ so that only shift operations and adders are required in the implementation of the sigmoid function. However, the rough approximation scheme brings too much precision loss. In contrast, the proposed method uses 24 segments to achieve higher precision in range $[-8, 8]$, while remaining very high area efficiency. As a result, E_{ave} and E_{max} of the PLAN method are 2.49× and 3.69× lower than the proposed scheme II, respectively. In terms of area efficiency, the PLAN method is higher than the proposed Scheme II due to the trade-off of precision for smaller area. As PLAN method has lower work frequency, it is unfair to compare the dynamic power according to the power numbers. If under the same work frequency, the proposed method can possibly achieve the same order of power consumption as the PLAN method.

V. Conclusion

In this brief, we present a novel approximation method and efficient hardware architectures for the sigmoid function. The proposed method exploit a special piecewise scheme and Taylor series expansion-based equations to approximate the function with high precision and low-complexity. Taking advantage of special characteristics of the sigmoid function, we further propose optimized implementation schemes, where no multipliers and LUTs are required. As the proposed circuits contain purely combinational logic, the sigmoid function can be computed in one clock cycle. According to the experimental results, the proposed circuits can achieve a good trade-off among speed, area and precision.

References

[1] K. Leboeuf, A. H. Namin, R. Muscedere, H. Wu, and M. Ahmadi, “High speed VLSI implementation of the hyperbolic tangent sigmoid function,” in *Proc. Int. Conf. Convergence Hybrid Inf. Technol. (ICCIT)*, vol. 1, Nov. 2008, pp. 1070–1073.

[2] F. Piazza, A. Uncini, and M. Zenobi, “Neural networks with digital LUT activation functions,” in *Proc. Int. Conf. Neural Netw. (IJCNN)*, vol. 2, Oct. 1993, pp. 1401–1404.

[3] H. Sun *et al.*, “A universal method of linear approximation with controllable error for the efficient implementation of transcendental functions,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 1, pp. 177–188, Jan. 2020.

[4] M. Bajger and A. Omondi, “Low-error, high-speed approximation of the sigmoid function for large FPGA implementations,” *J. Signal Process. Syst.*, vol. 52, no. 2, pp. 137–151, Aug. 2008.

[5] A. Armato, L. Fanucci, G. Pioggia, and D. D. Rossi, “Low-error approximation of artificial neuron sigmoid function and its derivative,” *Electron. Lett.*, vol. 45, no. 21, pp. 1082–1084, Oct. 2009.

[6] A. Armato, L. Fanucci, E. P. Scilingo, and D. De Rossi, “Low-error digital hardware implementation of artificial neuron activation functions and their derivative,” *Microprocess. Microsyst.*, vol. 35, no. 6, pp. 557–567, 2011.

[7] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, “Piecewise linear approximation applied to nonlinear function of a neural network,” *IEE Proc. Circuits Devices Syst.*, vol. 144, no. 6, pp. 313–317, Dec. 1997.

[8] N. B. Gaikwad, V. Tiwari, A. Keskar, and N. C. Shivaprakash, “Efficient FPGA implementation of multilayer perceptron for real-time human activity classification,” *IEEE Access*, vol. 7, pp. 26696–26706, 2019.

[9] M. Zhang, S. Vassiliadis, and J. G. Delgado-Frias, “Sigmoid generators for neural computing using piecewise approximations,” *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 1045–1049, Sep. 1996.

[10] I. D. Campo, R. Finker, J. Echanobe, and K. Basterretxea, “Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons,” *Electron. Lett.*, vol. 49, no. 25, pp. 1598–1600, Dec. 2013.

[11] S. Gomar, M. Mirhassani, and M. Ahmadi, “Precise digital implementations of hyperbolic tanh and sigmoid function,” in *Proc. Conf. Rec. Asilomar Conf. Signals Syst. Comput. (ACSSC)*, Nov. 2016, pp. 1586–1589.

[12] M. T. Tommiska, “Efficient digital implementation of the sigmoid function for reprogrammable logic,” *IEE Proc. Comput. Digit. Tech.*, vol. 150, no. 6, pp. 403–411, Nov. 2003.

[13] V. Beiu, J. Peperstraete, J. Vandewalle, and R. Lauwereins, “Closse approximations of sigmoid functions by sum of step for VLSI implementation of neural networks,” *Sci. Ann. Cuza Univ.*, vol. 3, pp. 5–34, Jan. 1994.

[14] M. Wang, S. Lu, D. Zhu, J. Lin, and Z. Wang, “A high-speed and low-complexity architecture for softmax function in deep learning,” in *Proc. IEEE Asia-Pac. Conf. Circuits Syst. (APCCAS)*, Oct. 2018, pp. 223–226.

[15] B. Zamanlooy and M. Mirhassani, “An analog CVNS-based sigmoid neuron for precise neurochips,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 3, pp. 894–906, Mar. 2017.