Low-Complexity High-Performance Method for Calculating Arbitrary Logarithm Function

Yongzhen Zhang[1], Yuan Zhang[2], Yonggang Zhang[3*] and Hui Chen[3*]
[1]School of Electrical Engineering, Zhengzhou University, China
[2]School of Urban Railway Transportation, Shanghai University of Engineering Science, China
[3]School of Electronic Science and Engineering, Nanjing University, China

***Abstract*—Logarithms are widely used in signal processing and other fields. In this paper, a hardware-friendly logarithm implementation method based on piecewise linear and the constant multiplier is proposed. Compared with other methods, it has a wider calculation range and stronger scalability. Our method maximizes functionality while minimizing resources. Also, it has the characteristics of low complexity and high precision.**

***Index Terms*—Logarithm, Low Complexity, High Performance**

2022 19th International SoC Design Conference (ISOCC) | 978-1-6654-5971-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISOCC56007.2022.10031359

I. Introduction

The logarithm can not only reduce the computational complexity, but its curve can also describe some natural phenomena. However, due to the particularity of the calculation, it cannot be directly carried out on the hardware. There are some hardware implementation methods, such as Taylor series, CORDIC, improved parabolic synthesis and so on [1]. By using Taylor series approximation, the reference [2] approximates the logarithmic function with base e. The reference [3] presents a CORDIC-based method to compute the logarithm function with base 2. These methods typically require complex hardware or have long time delays, which means they are not suitable for high-speed, low-power applications. Therefore, we propose a low-complexity method for computing arbitrary logarithms based on piecewise linear (PWL) and optimized constant multipliers.

II. Proposed Method

Firstly, we need to transform the formula to simplify the calculation. According to the change of base of logarithms, any logarithmic calculation can be converted to the following form:

$$log_a b = \frac{log_2 b}{log_2 a} = (log_a 2) \times [log_2(b)] \tag{1}$$

To increase the computational support range, b is entered as a single-precision floating-point number. According to the formula rule of the floating-point number, b can be expressed as $1.M \times 2^E$, so we can convert (1) to this function:

$$log_a b = (log_a 2) \times [log_2(1.M) + E] \tag{2}$$

To calculate the value of $log_a b$, the first step is that we use piecewise linear to find the value of $log_2(1.M)$. Then add the value of $log_2(1.M)$ to E. During the piecewise linear for the solution of $log_2(1.M) + E$, the curve is segmented and each segment is approximately simulated as a straight line, $y = kx + m$, and then the integration of multiple straight lines can be regarded as the original curve, that is, the formula:

$$log_2(1.M) + E = y + E = kx + m + E \tag{3}$$

For hardware friendly implementation, we multiply $log_a 2$ and $log_2(1.M) + E$ with constant multipliers. The higher the number of constant multipliers, the higher the accuracy. If a is 2 or 4, the value of $log_a 2$ is simple. If a is some other number, such as 3, 5, 6, 7, 8, 9 and e, the value of $log_a 2$ is as follows:

$$\begin{aligned} log_3 2 &\approx (0.10100001100000101)_2 \\ log_5 2 &\approx (0.011011100100001001)_2 \\ log_6 2 &\approx (0.011000110000101)_2 \\ log_7 2 &\approx (0.0101101100101)_2 \\ log_8 2 &\approx (0.010101010101)_2 \\ log_9 2 &\approx (0.01010000110001)_2 \\ log_e 2 &\approx (0.1011000101101)_2 \end{aligned}$$

Next, based on the above operations, we can complete the calculation of $log_a 2 \times [log_2(1.M) + E]$ through addition, subtraction and shift operations. Take $log_2 b \times log_3 2$ for example, the calculation process is as follows:

$$\begin{aligned} Result = (log_2 b >> 1) + (log_2 b >> 3) + (log_2 b >> \\ 8) + (log_2 b >> 9) + (log_2 b >> 15) + (log_2 b >> 17) \end{aligned} \tag{4}$$

Therefore, the specific calculation steps are as follows:

1) E: Given the form of single-precision floating-point numbers, we can easily calculate E.

2) $log_2(1.M)$: Because the range of $1.M$ is [1, 2), we can use the PWL method to compute $log_2(1.M)$. It divides the corresponding curve of the value range of $log_2 x$ into 32 segments, and then calculate the approximate curve of each segment, $y = kx + m$.

3) $log_a b$: After calculating $log_2(1.M)$, we add $log_2(1.M)$ and E, then perform a constant multiplier operation based on the value of a, and finally get the value of $(log_a 2) \times [log_2(1.M) + E]$.

Based on the proposed method, almost all parts can be reused in hardware implementation, and we can implement the logarithm function with arbitrary base.

III. Software Experiment

First, we use software to simulate our proposed method. To simulate hardware implementation more realistically, fixed-point quantization of data is carried out in software. In software simulation, the piecewise linear is divided into 32 segments, and the fractions of k and m are quantized to 8 decimal places. The number of constant multipliers is 2. In the interval of (0, 32], We take 1000 uniformly distributed values of b to calculate the average error, the maximum error and the mean square error. The results are shown in Table I.

TABLE I
Error of the Proposed Method

a	Average Error	Maximum Error	Mean Square Error
2	5.81×10^{-5}	1.71×10^{-4}	4.73×10^{-9}
3	1.53×10^{-4}	2.37×10^{-4}	2.54×10^{-8}
4	2.91×10^{-5}	8.53×10^{-5}	1.18×10^{-9}
5	5.76×10^{-5}	1.61×10^{-4}	4.10×10^{-9}
6	4.80×10^{-5}	1.39×10^{-4}	2.96×10^{-9}
7	4.87×10^{-4}	6.55×10^{-4}	2.61×10^{-7}
8	3.15×10^{-4}	4.21×10^{-4}	1.09×10^{-7}
9	7.74×10^{-4}	1.67×10^{-4}	7.01×10^{-9}
e	1.04×10^{-3}	1.40×10^{-3}	1.20×10^{-6}

It can be seen from Table I that the logarithmic mean square error achieved by the proposed method can reach the order of magnitude of 10^{-6} to 10^{-9}. Therefore, the logarithmic function value calculated by the proposed method is close to the real value calculated by the software, which also shows that the proposed method is very effective.

Through software experiments, we found that increasing the accuracy of $log_2 b$, that is, increasing the number of constant multipliers, or increasing the number of segments of the piecewise linear and the accuracy of k and m can reduce the error. Therefore, we can make a trade-off between resource consumption and accuracy as needed [4].

IV. Hardware Implementation

A. Hardware Implementation

Based on the software experiment results, we can design an example of logarithmic function circuit that can support $a = 2, 3, 4...9$ and e by using Verilog HDL. The hardware architecture of the proposed method is shown in Fig. 1.

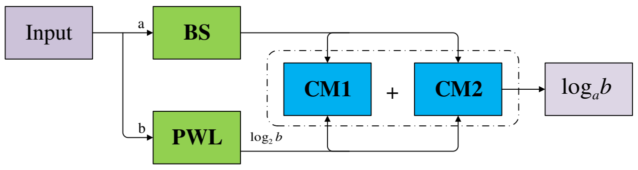

Fig. 1. Hardware architecture of the proposed method

The hardware architecture mainly includes three modules: BS, PWL and CM. BS is a module to select the base of logarithm. PWL is a piecewise linear module. CM1 and CM2 are constant multiplier modules.

According to Fig. 1, first we send b into the PWL module, calculate $log_2 1.M$, and then add E to calculate $log_2 b$. The BS module selects different constant values of $log_2 a$ according to a and sends them to two constant multiplier modules for calculation. Finally, we can obtain the value of $log_a b$.

In the hardware implementation, both the PWL and CM modules can be reused, and the calculation of any logarithmic function can be realized only by adding constant values of $log_2 a$ in the BS module.

B. Result Analysis

We synthesize the example design using the Synopsys Design Compiler (DC) under CMOS 28nm and 65nm technology, and the comparison results with other works are shown in Table II.

TABLE II
The Performance Comparision of Different Methods

	[1]Ref. [1]	Ref. [3]	Proposed
Base of logarithm	e	2	2–9,e
Technology	65nm	65nm	65nm
Frequency (MHz)	20	1024	512
Area(um^2)	3100	24100	6040

[1] Employing optimized GS proposal.

It can be seen from Table II that although the area of our proposed method is larger than that of [1], the frequency we can achieve is higher. Compared with [3], the area of our proposed method has obvious advantages. In addition, both reference [1] and reference [3] can only realize one base of the logarithmic function, but our method can support the base of $2, 3, 4...9$ and e. More importantly, our method can support arbitrary logarithmic function calculation.

V. Conclusion

We propose a hardware implementation method of arbitrary logarithmic function. It only needs to add an optional constant value of $log_2 a$ to the BS module to realize the logarithm of any base. At the same time, we use two constant multipliers to replace the traditional multiplication operation, which can save hardware resources. Compared with the existing methods, it has the characteristics of low complexity, high performance and high flexibility. Therefore, the proposed method has great application prospects in the field of the logarithmic hardware implementation.

References

[1] P. cker *et al.*, "Optimizing iterative-based dividers for an efficient natural logarithm operator design," in *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, 2020, pp. 1–4.

[2] M. R. Weirich *et al.*, "A fixed-point natural logarithm approximation hardware design using taylor series," in *2018 New Generation of CAS (NGCAS)*, 2018, pp. 53–56.

[3] H. Chen *et al.*, "Hyperbolic CORDIC-based architecture for computing logarithm and its implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2652–2656, 2020.

[4] M. S. Ansari, B. F. Cockburn, and J. Han, "A hardware-efficient logarithmic multiplier with improved accuracy," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 928–931.