An Efficient ASIC Implementation of Logarithm Approximation for HDR Image Processing

Van-Phuc Hoang and Xuan-Tien Do
Le Quy Don Technical University
236 Hoang Quoc Viet Str., Hanoi, Vietnam

Cong-Kha Pham
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan

***Abstract*— This paper presents an efficient ASIC implementation for the hardware approximation of the logarithm function which can be used for emerging high dynamic range image processing applications. By employing a new logarithm approximation method, the modified barrel shifter circuit and optimized leading one detector and encoder, the proposed approach can reduce the hardware area and improve the logarithm computation speed significantly while achieve the similar accuracy compared with other methods. The implementation results in 0.18-µm CMOS technology are also presented and discussed.**

I. INTRODUCTION

Currently, with the increasing demand for high dynamic range (HDR) image rendering, processing and display devices and applications, the efficient HDR image processing hardware implementation is highly required for such real-time applications. In this kind of applications, the efficient logarithm hardware approximation method in the logarithm transformation is one of the key issues [1]-[3]. Fig. 1 presents the block diagram of a typical HDR image encoder using logarithm transformation [2] where L denotes the luminance component. Using logarithm for HDR video compression was presented in [3] as well. Also, efficient logarithm function generators are highly required in digital communication systems and other real time digital signal processing (DSP) applications [4].

Moreover, the linear binary to logarithm converter and logarithm to linear binary converter are essential components in logarithmic number system (LNS) and hybrid number system (HNS) processors [5]. Hence, reducing the hardware complexity of these components can lead to a great impact on the overall HNS-DSP systems.

Therefore, many researchers have been focused on efficient methods to implement the logarithm and anti-logarithm approximations [4]-[16]. However, a more efficient method is desired, especially for emerging HDR image/video processing applications and LNS/HNS processors. In this paper, we present a novel and efficient architecture for the binary logarithm approximation used in HDR imaging and its optimized ASIC (Application Specific Integrated Circuit) implementation.

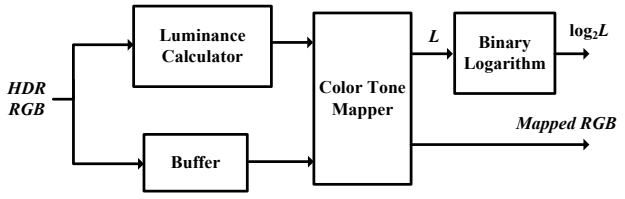


Fig. 1. An typical HDR image encoder [2].

The rest of this paper is organized as follows. Section II provides the brief introduction and hardware architecture for logarithm computation. Section III and section IV describe the proposed method for the key components in a logarithm generator. Then, section V presents the implementation results in 0.18-µm CMOS technology. Finally, the conclusion is included in section VI.

II. LOGARITHM HARDWARE COMPUTATION

Without lost of generality, consider the binary (base-2) logarithm of an unsigned number N which can always be decomposed as [4]:

$$N = 2^n (1+x) \quad (1)$$

where n, called the characteristics of N, corresponds to the position of the most significant "one" bit of N in its binary representation and x is the fraction value with: $0 \le x < 1$. As a result, the binary logarithm of N can be expressed as:

$$\log_2 N = n + \log_2 (1+x) \quad (2)$$

Therefore, to compute $\log_2 N$, firstly, we detect the most significant "one" bit of N and then, approximate $\log_2(1+x)$ called the fundamental function. Many researchers have been focused on finding the efficient methods to approximate the fundamental function $\log_2(1+x)$ since it is the essential step in the logarithm computation.

The proposed hardware architecture for a logarithm generator is depicted in Fig. 2. The leading one detector and encoder (LODE) block computes the characteristic n and encodes it into the binary form. An inverter block (INV) and the modified barrel shifter are used to generate the fraction part x in (1). The fundamental function approximation block provides the fraction part (F) of logarithm result by approximating $\log_2(1+x)$. Moreover, a flag (z) is used to indicate the case of zero input.

III. DECOMPOSED LEADING ONE DETECTOR-ENCODER AND MODIFIED BARREL SHIFTER

As shown in Fig. 2, the leading one detector and encoder (LODE) is used to generate the integer part of the result and provide the control signals for barrel shifter, together with the inverter block. In this section, we present an efficient LODE circuit based on the merged LODE cell which can reduce both hardware area and computation delay. Merged LODE means that the leading one detector and one-hot encoder are merged

978-1-4799-1089-2/13/$31.00 ©2013 IEEE

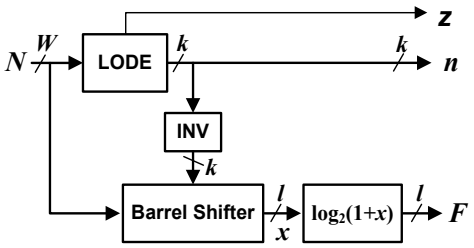
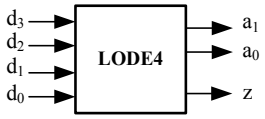


Fig. 2. Proposed logarithm function generator.

into a module in which the binary coded output (for the position of the leading one bit) is computed directly from the input word. Fig. 3 presents the block symbol and its truth table for the 4-bit merged LODE in which a_1a_0 denotes the 2-bit result from a 4-bit input ($d_3d_2d_1d_0$). A zero flag (z) is also used to indicate the case of zero input. This method can be applied for 8-bit and higher bit-width LODE circuits. However, applying this direct merged LODE method for designs with the very high bit-width may lead to high hardware complexity and power consumption.

Therefore, to get the best trade-off between speed and hardware complexity for high bit-width designs, we employ the method of decomposing the input binary word into smaller parts which are processed in parallel. Some additional combinational circuits are also needed to generate final results as shown in Fig. 4 which is the architecture of the proposed 16-bit LODE. The input word is decomposed into four parts which are fed to four 4-bit LODEs. Then, two 4-input multiplexers (MUX4) are used to select the proper 4-bit input part and constant value which are then fed to the 4-bit adder to provide the final result. The hardware architectures for 32-bit and 64-bit LODE circuits are similar to 16-bit LODE with the primitive components of 8-bit and 16-bit LODEs, respectively, instead of 4-bit LODE in 16-bit LODE design.

Moreover, the modified barrel shifter (BS) architecture is used to reduce the hardware complexity by minimizing the number of its control bits. In a conventional BS, the number of shifts is (W+1-m) where m is the output of LODE. In our modified BS, the INV (inverter) block is used to generate the value (W-m) to control the conventional BS and then, an additional shift is performed to generate the fraction part (x).



d_3	d_2	d_1	d_0	a_1a_0	z
1	x	x	x	11	1
0	1	x	x	10	1
0	0	1	x	01	1
0	0	0	1	00	1
0	0	0	0	00	0

Fig. 3. The 4-bit LODE and its truth table.

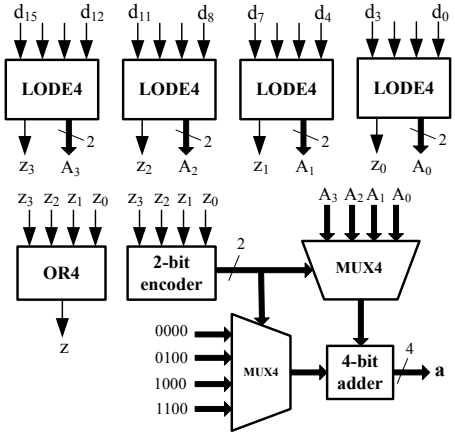


Fig. 4. Proposed merged 16-bit LODE with input decomposition.

IV. EXISTING AND PROPOSED METHODS FOR THE FUNDAMENTAL FUNCTION APPROXIMATION

A. Brief Review of Existing Approximation Methods

J. N. Mitchell [6] proposed a method of using the simple linear approximation as follows:

$$\log_2(1+x) \approx x \tag{3}$$

The error function due to this approximation method is:

$$E_L(x) = \log_2(1+x) - x \tag{4}$$

where E_L, called Mitchell error function, denotes the error function for the approximation in (3). The maximum value of this error function is 0.08639, and the accuracy is only 3.5 bits. Therefore, there have been many researches focusing on more accurate methods.

In the piecewise linear approximation method, the range of x of [0, 1) is divided into number of regions. For each region, E_L is approximated by a linear function called a segment which can be expressed as:

$$y = slope * x + offset \tag{5}$$

However, a linear segment can also be defined by two of following parameters: slope, offset and one point (with x and y coordinates). It also can be defined by two points. The multiplication with *slope* in (5) can be simplified by using the shift-add logic.

In references [5], [7]-[12], some piecewise shift-add linear approximation methods with different numbers of segments and different forms of linear difference functions were presented. In [7]-[11], some methods were proposed with the number of segments of 2, 4 and 6 in which the parameters and coefficients of each linear segment are selected by the "trial and error" method. B.-G. Nam *et al.* [5] presented a method of dividing the input range into 24 regions for the logarithmic and 16 regions for the anti-logarithmic approximation. In [12], the authors also presented an optimizing method for the integer binary number to logarithmic conversion. However, these methods should be improved when they are employed for the high accuracy applications. Increasing the number of segments [5], [12] or using higher order approximation [13]

can improve the accuracy but also result in higher hardware complexity.

Moreover, the multipartite table method (MTM) for approximating the elementary functions including logarithm and anti-logarithm was presented in [14] in which only tables and adders are used. This approach can reduce the table size considerably compared with the direct look-up table (LUT) based method in which only an LUT is used to provide the approximation result. S. Paul *et al.* [15] presented a table-based method for LOG which combines an LUT and a multiplier-less linear interpolation so that the table size can be reduced compared with some other table-based methods for the same accuracy.

A combination of the linear difference method and the LUT-based correction is also used for logarithm approximation. The basic idea is that an LUT is used to store the difference function which is defined by the difference between the original function and the approximated one $D(x)$ as depicted in Fig. 5. In a simple method in [16], E_L is stored in an LUT. The LUT output is added with Mitchell approximation function to get the final result. R. Gutierrez *et al.* [4] proposed an improved method by using 4-segment linear approximation together with a small error LUT. It is also reported in [4] that this method outperforms previous methods. However, more improvements are desired and the process of selecting the linear function parameters and coefficients which is performed by the "trial and error" method may lead to a non-optimal architecture. Therefore, in the proposed method, an optimization algorithm is employed to find optimal parameters.

B. Proposed Fundamental Function Approximation Method

Firstly, consider the fundamental function $E_L(x)$, its mirror function $E_L(1\text{-}x)$ and the mean function $E_M(x)$ defined as:

$$E_M(x) = \frac{E_L(x) + E_L(1-x)}{2}$$

It can be seen that:

$$E_M(x) = E_M(1-x)$$

Therefore, $E_M(x)$ is symmetrical via the vertical line $x = 0.5$. As a result, it will be promisingly efficient if we can approximate this mean function because it requires only the approximation for a half range of x and the other half can be interpolated easily by a simple complement module in which the maximum significant bit is used as the control bit.

The optimization algorithm has to take both the complexity of approximated function and the LUT size into account. Therefore, we propose the 2-step optimization algorithm to achieve a good trade-off of the hardware complexity and computation speed. The objective of our proposed optimization algorithm is to find the optimal parameters for two segments that can reduce the LUT size as much as possible by minimizing the maximum value of the difference function, while achieve the low complexity of the approximation circuit. The quasi-symmetrical approach means that E_L is actually just nearly symmetrical, but it can be treated like a symmetrical function with some modifications.

Firstly, the entire range of x ($0 \le x < 1$) is divided into two halves to employ the quasi-symmetrical method. Then, we further divide the left half range of x ($0 \le x \le 0.5$) into 2 equal intervals [0; 0.25) and [0.25; 0.5] to make the selecting circuit simple. The algorithm 1 summaries the optimization process to derive the optimal parameters of the linear segments in which *peak_point* denotes the value of approximate function when $x = 0.5$ and the difference function is defined by the difference value between E_L and the linear function.

Fig. 6 depicts this optimization algorithm in which two linear segments are chosen independently. In step 1, a full search in the restricted ranges of *peak_point* and *offset1* is performed to find the optimal values of *slope1* and *slope2* that minimize the maximum value (*MaxDiff*) of the difference function which is stored in the LUT. Then, in step 2, *slope1* and *slope2* are re-assigned to the adjacent power-of-2 values to avoid the multiplications and another search is performed to find the optimal offset values which minimize *MaxDiff*. The ranges of *peak_point* and *offset1* are chosen to guarantee the acceptable accuracy of approximated results.

Table I summaries the optimization results in each step of the proposed algorithm for $\log_2(1+x)$ function. After step 2, *MaxDiff* increases slightly but the LUT size remains the same as the result of step 1. The error analysis results are presented in Table II for the case of $l = 13$ in which l denotes the fraction bit-widths. Fig. 7 depicts the proposed 2-segment quasi-symmetrical method for the fundamental function in logarithm approximation. It can be seen that the proposed method can achieve the similar accuracy and LUT size compared with the method in [4]. However, the proposed method can reduce the hardware complexity significantly because E_L is approximated by 4 segments, but only 2 segments are required to be computed actually. Fig. 8 shows the proposed hardware architecture to compute $\log_2(1+x)$.

Moreover, since the fundamental functions for logarithm and anti-logarithm approximations are similar [15], the proposed method can also be applied for the anti-logarithm function.

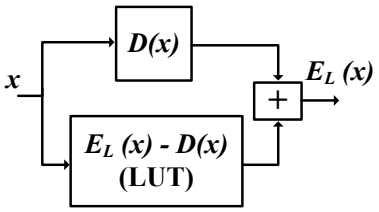


Fig. 5. Combined linear difference/LUT-based correction method.

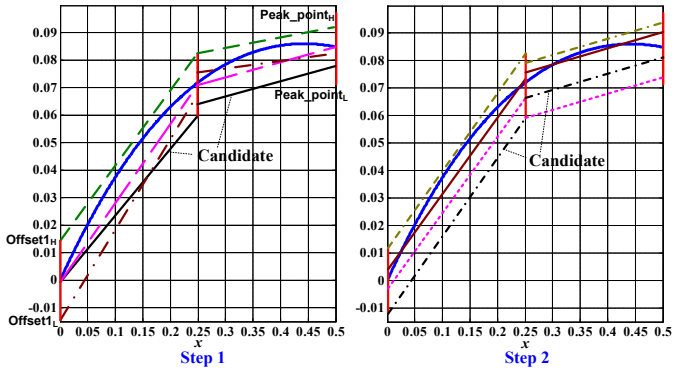


Fig. 6. The 2-step optimization for logarithmic computation.

Algorithm 1. Proposed 2-step optimization algorithm.

Step 1: For $\{offset1_L \le offset1 \le offset1_H$ and $peak\_point_L \le peak\_point \le peak\_point_H\}$:
Find the optimal values of *slope*1 and *slope*2.

Step 2: Re-assign the optimal *slope*1 and *slope*2 values in step 1 to the adjacent power-of-2 values and find the optimal offset values.

TABLE I. OPTIMIZATION RESULTS USING THE PARAMETER OPTIMIZATION ALGORITHM.

Step	Slope1	Offset1	Slope2	Offset2	MaxDiff
Step 1	0.2332	0.008	0.728	0.0341	0.0089 (1/112)
Step 2	0.25	0.004	0.0625	0.0518	0.0101 (1/99)

TABLE II. SUMMARY OF PARAMETERS AND ERROR ANALYSIS RESULTS FOR THE PROPOSED METHOD AND METHOD IN [4] FOR 13-BIT FRACTION.

Method	In [4]	Proposed
No. of segments	4	2 (4)
MaxDiff	0.0080 (1/125)	0.0101 (1/99)
LUT size (bit)	640	640
Mean Error	2.3×10^{-4}	2.3×10^{-4}
Maximum Error	8.0×10^{-4}	8.0×10^{-4}

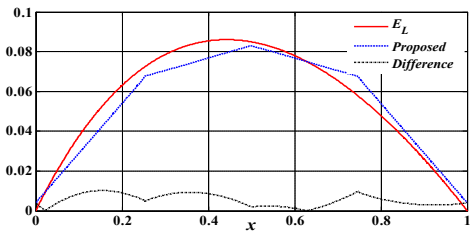


Fig. 7. Proposed quasi-symmetrical linear method.

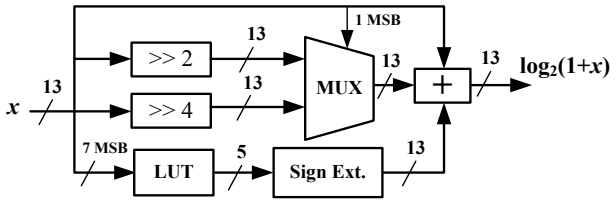


Fig.8. Proposed architecture to approximate $\log_2(1+x)$ with $l = 13$.

V. IMPLEMENTATION RESULTS

The proposed and reference architectures for the fundamental function, LODE circuit and the whole logarithm generation are implemented in 0.18-µm CMOS technology using a standard cell library with the same design parameters and constraints. To compare the overall converter performance, the results of area-delay product (ADP) are also presented. For the first prototype of our proposed method, we assume the implementation with 16-bit integer input (W = 16) and 13-bit fraction in the output (l = 13).

A. LODE

Table III presents the implementation results of different LODE architectures with 0.18-µm CMOS technology in which the conventional LODE indicates the architecture with separate leading one detector and one-hot encoder circuits and direct merged LODE indicates the merged LODE without input decomposition. It is shown that the proposed merged LODE architecture outperforms other methods in both hardware area and delay. ADP is also used to compare overall implementation results of different methods. Especially, for 64-bit LODE, the proposed architecture leads to significant reductions of ADP of 11x compared with the conventional LODE.

It is obvious that the higher values of bit-width, the higher ADP ratios between direct merged LODE and split LODE are achieved. Therefore, for the high bit-width LODE designs, the input decomposing technique should be employed to further improve LODE performance and area efficiency.

B. Fundamental Function Approximation

Table IV shows the implementation results of $\log_2(1+x)$ approximation with the fraction length of l = 13. In [4], a comparison for $\log_2(1+x)$ computation shows that the method proposed in [4] can reduce the hardware complexity by about 43% compared with the method presented in [15] while achieve the similar conversion speed and accuracy. Therefore, we compare the implementation results of our method with the method in [4] with the same hardware platform to clarify the improvements of the proposed approach. It is shown in this table that the proposed method can reduce the ADP value of 44% compared with the method in [4] for $\log_2(1+x)$.

C. Logarithm Generator

Table V presents the implementation of 16-bit logarithm generator (W=16). It can be seen that the proposed method can lead to the significant reduction of ADP value compared with other methods. Especially, compared with the best previous method for logarithm hardware computation in [4], the proposed method can reduce the ADP of 37%. Fig. 9 shows the chip microphotograph of the proposed 16-bit logarithm generator fabricated with 0.18-µm CMOS technology using a standard cell library.

TABLE III. IMPLEMENTATION OF DIFFERENT LODE ARCHITECTURES USING 0.18-µM CMOS TECHNOLOGY.

W	Architecture	Area (μm^2)	Delay (ns)	ADP ($\times10^3$)
16	Conventional LODE	900	5.8	5.22
	Proposed direct merged LODE	506	1.9	0.96
	Proposed split LODE	**490**	**1.7**	**0.83**
32	Conventional LODE32	3741	6.0	22.45
	Proposed direct merged LODE	1148	3.2	3.67
	Proposed split LODE	**1045**	**2.8**	**2.93**
64	Conventional LODE	12160	7.0	85.12
	Proposed direct merged LODE	3196	6.0	19.18
	Proposed split LODE	**2284**	**3.4**	**7.77**

TABLE IV. FUNDAMENTAL FUNCTION IMPLEMENTATION RESULTS.

Design	Area ($\times10^3\ \mu m^2$)	Delay (ns)	ADP ($\times10^3$)
Direct LUT	37.2	6.3	234.4
MTM-based	7.5	9.6	72.0
Method in [4]	5.2	8.7	45.2
Proposed	4.2	6.0	25.2

TABLE V. IMPLEMENTATION RESULTS FOR 16-BIT LOGARITHM GENERATOR.

Method	Area ($\times 10^3 \mu m^2$)	Delay (ns)	ADP ($\times 10^3$)
Direct LUT	32.6	12.2	397.7
MTM-based	23.4	13.0	304.2
Method in [4]	9.4	10.3	96.8
Proposed	7.6	8.0	60.8

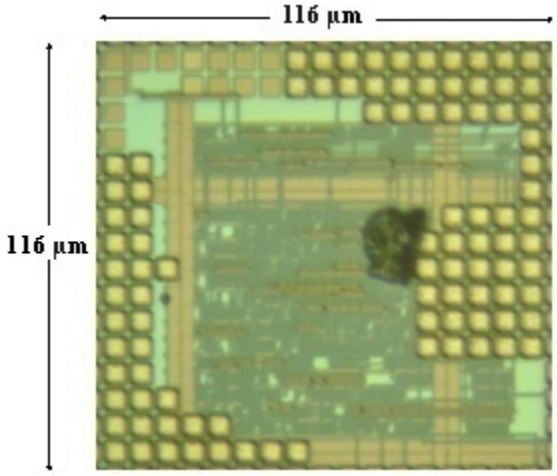


Fig. 9. Chip microphotograph of the proposed 16-bit logarithm generator.

VI. CONCLUSIONS

In this paper, we presented an efficient approach for logarithm hardware computation which is highly required for HDR image/video processing applications and HNS DSP processors. The presented ASIC implementation combined the efficient quasi-symmetrical logarithm approximation method, the modified barrel shifter circuit and the decomposed leading one detector and encoder techniques. The implementation results in 0.18-µm CMOS technology have clarified the efficiency of the proposed method. In our future work, we will implement a full HDR image processing system using the proposed logarithm computation module.

ACKNOWLEDGMENT

This chip presented in this paper was fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), The University of Tokyo in collaboration with ROHM CO. LTD.

REFERENCES

[1] S. P. R. Xu and C. E. Hughes, "High-dynamic-range still image encoding in JPEG 2000," *IEEE Computer Graphics and Applications*, vol. 25, issue 6, pp. 57–64, Nov. 2005.

[2] F. Hassan and Joan Carletta, "A high throughput encoder for high dynamic range images," in *Proc. IEEE International Conference on Image Processing* (ICIP 2007) , vol.6, pp.VI-213 - VI-216, Sep. 2007.

[3] A. Motra and H. Thoma, "An adaptive Logluv transform for high dynamic range video compression," in *Proc. 17th IEEE International Conference on Image Processing (ICIP)*, pp.2061-2064, Sep. 2010.

[4] R. Gutierrez and J. Valls, "Low cost hardware implementation of logarithm approximation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 12, pp. 2326-2330, Dec. 2011.

[5] B.-G. Nam, H. Kim, and H.-J. Yoo, "A low-power unified arithmetic unit for programmable handheld 3-D graphics systems," *IEEE J. Solid-State Circuits*, vol. 42, no. 8, pp.1767-1778, Aug. 2007.

[6] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IEEE Trans. Electron. Comput.*, vol. 11, no.11, pp. 512-517, Aug. 1962.

[7] M. Combet, H. Van Zonneveld, and L. Verbeek, "Computation of the base two logarithm of binary numbers," *IEEE Trans. Electron Comput.*, vol. EC-14, no. 6, pp. 863-867, Dec. 1965.

[8] E. L. Hall, D. D. Lynch, and S. J. Dwyer, "Generation of products and quotients using approximate binary logarithms for digital filtering applications," *IRE Trans. Comput.*, vol. 19, pp. 97-105, Feb. 1970.

[9] S. Sangregory, C. Brothers, D. Gallagher, and R. Siferd, "A fast, low-power logarithm approximation with CMOS VLSI implementation," in *Proc. 42nd Midwest Symp. Circuits Syst.*, pp. 388-391, Aug. 1999.

[10] K. H. Aded and R. E. Siferd, "CMOS VLSI implementation of low power logarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1221-1228, Nov. 2003.

[11] T. B. Juang *et al*., "A lower error and ROM-free logarithmic converter for digital signal processing applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 12, pp. 931-935, Dec. 2009.

[12] D. De Caro, N. Petra and A. G. M. Strollo, "Efficient logarithmic converters for digital signal processing applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 10, pp. 667-671, Oct. 2011.

[13] S. Nasayama, T. Sasao, and J. T. Butler, "Programmable numerical function generators based on quadratic approximation: architecture and synthesis method," in *Proc. Asia and South Pacific Conference on Design Automation*, pp. 378-383, Jan. 2006.

[14] Florent de Dinechin, and Arnaud Tisserand, "Multipartite table methods," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 319-330, Mar. 2005.

[15] S. Paul *et al.*, "A fast hardware approach for approximate, efficient logarithm and antilogarithm computations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 269-277, Feb. 2009.

[16] G. L. Kmetz, "Floating point/logarithmic conversion systems," U.S. patent 4583180, Apr. 1986.