International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 5, October 2020, pp. 4671~4678
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i5.pp4671-4678

An efficient hardware logarithm generator with modified quasi-symmetrical approach for digital signal processing

Minh Hong Nguyen
Le Quy Don Technical University, Vietnam

Article Info

Article history:

Received Oct 18, 2019
Revised Mar 24, 2020
Accepted Mar 31, 2020

Keywords:

FPGA
Logarithm generator
Quasi-symetrical approach

ABSTRACT

This paper presents a low-error, low-area FPGA-based hardware logarithm generator for digital signal processing systems which require high-speed, real time logarithm operations. The proposed logarithm generator employs the modified quasi-symmetrical approach for an efficient hardware implementation. The error analysis and implementation results are also presented and discussed. The achieved results show that the proposed approach can reduce the approximation error and hardware area compared with traditional methods.

Corresponding Author:

Minh Hong Nguyen,
Le Quy Don Technical University,
236 Hoang Quoc Viet Str., Hanoi, Vietnam.
Email: nguyenhaihong2007@yahoo.com.vn

1. INTRODUCTION

Many real-time digital signal processing (DSP) applications such as digital communication systems, speech recognition, image processing, etc. require logarithm operations with high speed and moderate accuracy. The hardware implementation of the logarithm function has great significance to ensure high-speed, low-power logarithm computation so that it can be used for the applications with real-time requirements. However, the hardware implementation usually leads to a higher hardware complexity compared with the software based implementation. Hence, we have to consider the hardware resource efficiency and complexity to archive a good trade-off between them. Since the hardware implementation of logarithm function is normally very complex and requires much time while real-time DSP applications do not require absolute precision, we often use approximation methods to implement the logarithm generators. Moreover, for the systems imploying the logarithmic number system (LNS) or the hybrid number system (HNS), it is desired to implement the efficient linear binary to logarithm converters (LOGC) as well as the logarithm to linear binary converters (ALOGC: anti-logarithmic converters). For example, as reported in [1-4], in the typical implementation of HNS processors dedicated for the 3-D graphic processing, the total area of LOGC/ALOGC part is 64% of the chip area. Therefore, many researchers are trying to reduce the hardware complexity of these converters. On the other hand, the trend of approximate computing becomes popular recently to meet ther requirement of real-time DSP and artificial intelligence applications [5-13]. The purpose of this research is to find an improved approximation method for the implementation of a low complexity, high speed hardware logarithm approximation method which can be applied for real-time DSP applications with acceptable computation accuracy.

The remainder of the paper is organized as follows. Section 2 introduces briefly about the basics of logarithm hardware approximation methods. Section 3 presents the proposed method and implementation results. Finally, Section 4 concludes of the paper.

2. BINARY LOGARITHM HARDWARE APPROXIMATION

Firstly, without lost of generality, we consider an unsigned number N to compute the binary (base-2) logarithm and it can be decomposed as:

$$N = 2^n(1+x) \tag{1}$$

where n can be determined by detecting the position of the most significant '1' bit of N in the binary representation. Moreover, the range of x, the fraction value, is defined as: $0 \le x < 1$. Then, we can rewrite the binary logarithm expression as:

$$\log_2 N = n + \log_2(1+x) \tag{2}$$

By using (2), to compute the binary logarithm of N, in the first step, we detect the most significant '1' bit in its binary representation. Then, we can approximate the function $\log_2(1+x)$ which is the fractional part of the result. Here, $\log_2(1 + x)$ is considered as the fundamental function. Therefore, many researchers are trying to find the efficient methods for the fundamental function approximation.

The simplest method for the fundamental function approximation was proposed by J. N. Mitchell [14] with very simple linear approximation as follows:

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

This approximation approach is simple and leads to very fast and low complexity hardware implementation, with the tradeoff of the following absolute error function:

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

Whose maximum value is 0.08639 resulting in the accuracy of only 3.53 bits which is too low for most of DSP applications. Therefore, many methods were developed to find error correction techniques for Mitchell's method. There are three commonly used methods to improve the accuracy of this approximation: LUT-based method, piece-wise linear interpolation method and combination method which combines two above methods. In the LUT-based method, a LUT (Look-up Table) that stores an approximation of the residual error is added to Mitchell's approximation to reduce Mitchell error. However, the Mitchell error function maximum value is very high, this method requires very high table size. Another approach is the multipartite method which was presented in [15]. In this method, tables and adders are utilized to reduce the table size significantly compared with the direct LUT based method. A method of using a LUT and a multiplier-less linear interpolation was proposed by S. Paul et al. [16]. It requires less memory than some other LUT-based methods with the same requirement of the accuracy.

In the approximation methods using piecewise linear approach, the range of x is divided into several regions. Then, in each region, E_L is approximated by a linear function called a segment which can be expressed as:

$$y = a_i x + b_i \tag{5}$$

Increasing the number of segments can reduce the approximation error but lead to higher hardware complexity. Some methods for dividing the range of x into different regions were proposed in [1, 17-24]. Papers [17-22] presented the methods with 2, 4 and 6 regions with different values of slopes *ai* and constants *bi*. These values are chosen by "trial and error" method without detail optimization method. Figure 1 represents the error function and the linear approximation method using 4 segments and a small error LUT proposed in [22]. In [23, 24], authors proposed the quasi-symmetrical method to reduce the hardware complexity and approximation error. Moreover, in [1], B.-G. Nam et al. proposed a method with the number of segment of 24 for the logarithmic approximation. However, these methods should be improved for the high accuracy applications. A method which combines the piecewise linear approximation method and LUT-based correction may be the most effective technique for logarithm approximation [21-24]. The basic idea is that after using the linear approximation, a LUT which is utilized to store an approximation of the error between the fundamental function $\log_2(1+x)$ and the approximation function is added to linear approximation as described in Figure 2. Moreover, R. Gutierrez et al. [22] proposed a method using 4-region

linear approximation using a 128×5-bit residual error LUT which outperforms previous methods. However, the coefficients used in this method, which were selected by "trial and error", may be not optimal. Therefore, the objective of this research is to find an improved approach by modifying the quasi-symmetrical method in [23] with an improved method with a modified optimization algorithm to find optimal coefficients of the piece-wise linear approximation.

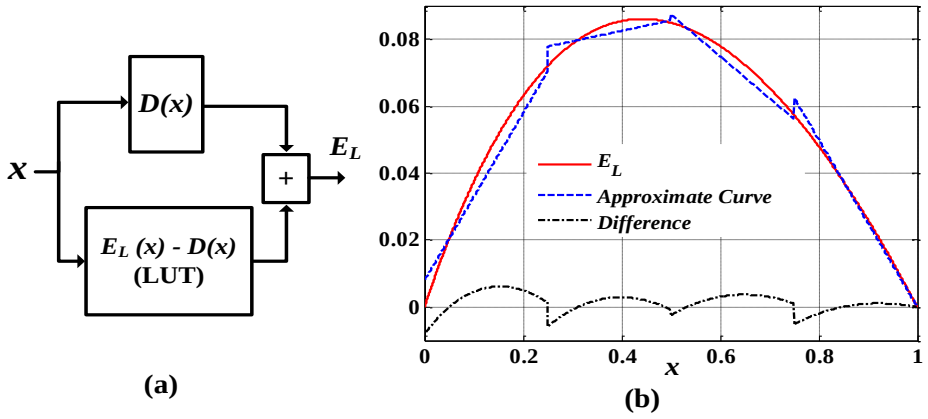


Figure 1. (a) General diagram for the difference method in logarithm approximation and (b) the method using 4-region linear approximation together with a small error LUT in [22]

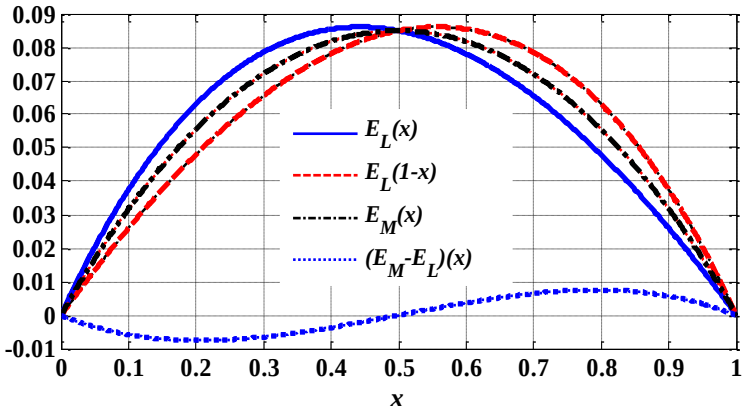


Figure 2. The idea for proposed quasi-symmetrical approach [23]

3. PROPOSED APPROXIMATION METHOD AND IMPLMENTATION RESULTS

Firstly, consider the fundamental function $F(x) = \log_2(1+x)$ which is represented in Figure 3. The graph line, which is slightly curved, is nearly a straight line. Therefore, it would be promisingly efficient if we use the piecewise linear approximation method for this function instead of E_L. The full range is divided into 4 segments to so that a simple selection circuit can be used with an acceptable accuracy. The approximation can be expressed as:

$$F(x) = \log_2(1+x) \approx D(x) = a_i x + b_i \tag{6}$$

In which $i \in \{1, 2, 3, 4\}$

Moreover, the slopes a_i are chosen to be sum of power-of-two values (2^k) so that we can implement the multiplications by simple shifting operations. Then, the error function causing by this approximation method can be expressed as:

$$E(x) = F(x) - D(x) = \log_2(1+x) - (a_i x + b_i) \tag{7}$$

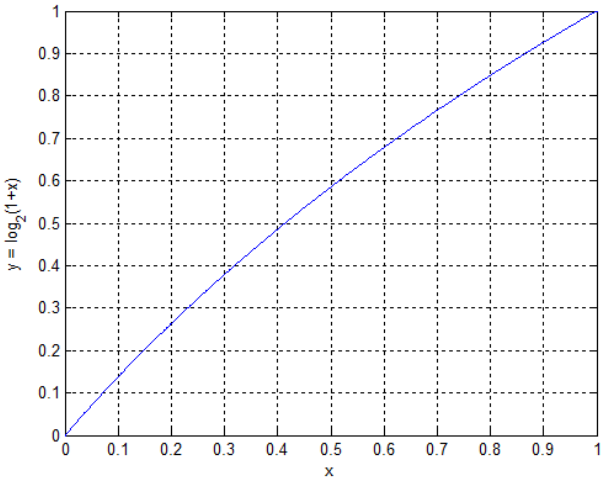


Figure 3. The fundamental function $F(x) = \log_2(1 + x)$

An LUT is used to store the optimized values of the error function $E(x)$. Then, the LUT output is added to the 4-segment linear approximation function to further reduce the residual error. The higher LUT size, the higher accuracy level of the approximation can be achieved. However, it also leads to the higher hardware complexity of the approximation circuit.

In this paper, in order to reduce the final approximation error with the small enough LUT size, we use an algorithm to find optimal values of a_i and b_i. We have to consider the approximation function complexity as well as the size of the correction LUT. Therefore, we proposed an improved 2-step optimization algorithm based on the one in [23] to achieve a better trade-off of the approximation circuit complexity to the correction LUT size. The proposed optimization algorithm aims to find the optimal values a_i, b_i for 4 linear segments and the LUT size can be reduced as much as possible by minimizing the maximum value of the absolute error function $|E(x)|$ (*MaxError*). The optimization algorithm is performed by Matlab software.

In the proposed algorithm, firstly, the range of x is divided into 2 halves and the algorithm for each half is proceeded independently. The left half ($0 \le x \le 0.5$) is divided into two equal regions ($0 \le x \le 0.25$) and ($0.25 \le x \le 0.5$). Figure 4 describes the optimization algorithm for the left half in which 2 linear segments are chosen independently. In step 1, we choose the ranges of *offset1* and *offset2* in which *offset1* and *offset2* represent the values of approximation function when $x = 0$ and $x = 0.5$, respectively. The ranges of *offset1* and *offset2* are chosen to ensure the acceptable accuracy of approximation results. Then, a comprehensive search in the ranges of *offset1* and *offset2* is performed to find the optimal values of a_1 and a_2 that minimize the *MaxError*. After that, in step 2, a_1 and a_2 are re-assigned to the adjacent values which are the sum of power-of-two values to simplify the multiplications and one more search is performed to find the optimal values of b_1 and b_2 which minimize *MaxError*. For the right half ($0.5 \le x < 1$), the optimization algorithm is implemented similarly. Figure 5 depicts the 2-step algorithm for the right half range of x.

Table 1 summarizes the results of optimization achieved by the proposed algorithm in each approximation step for $\log_2(1+x)$. After step 2, *MaxError* increases a little but the LUT size is not changed compared with the results in step 1. Hence, the approximation function can be expressed as (8).

Algorithm 1. The improved 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.

$$\log_2(1+x) = \begin{cases} (2^0 + 2^{-2} + 2^{-5})x + 0.005, & 0 \le x < 0.25 \\ (2^0 + 2^{-5})x + 0.0684, & 0.25 \le x < 0.5 \\ (2^{-1} + 2^{-2} + 2^{-3})x + 0.1505, & 0.5 \le x < 0.75 \\ (2^{-1} + 2^{-2})x + 0.248, & 0.75 \le x < 1 \end{cases} \quad (8)$$

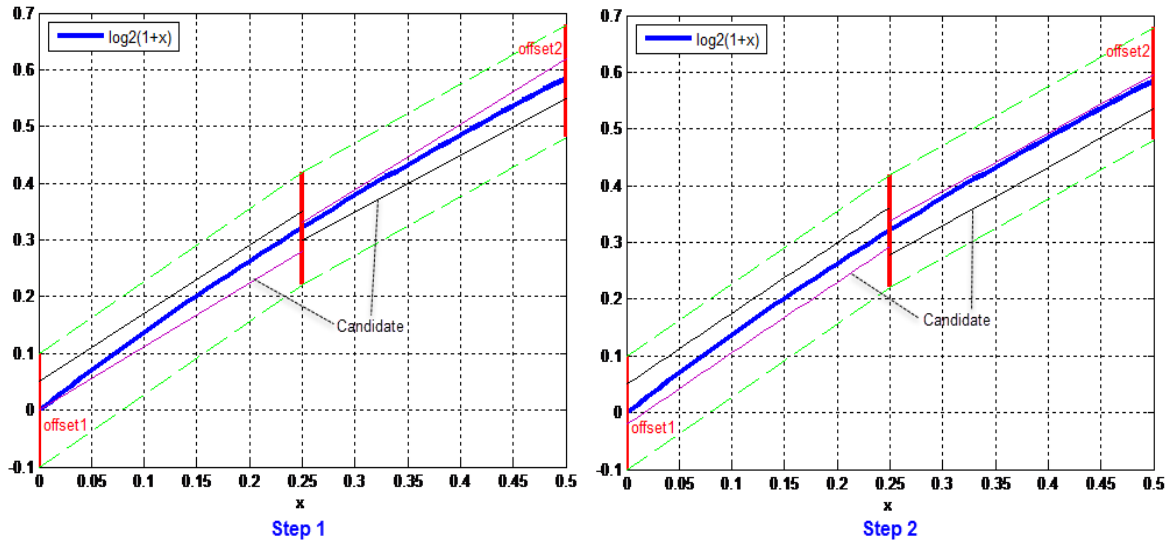


Figure 4. The improved 2-step algorithm for the left half range

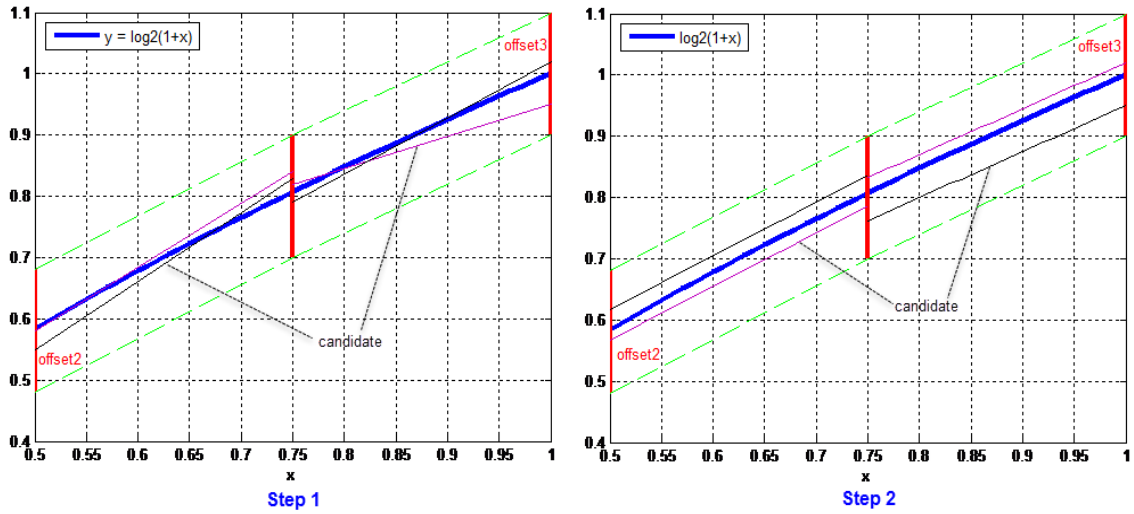


Figure 5. The improved 2-step algorithm for the right half range

Table 1. Optimization results of the improved 2-step optimization algorithm

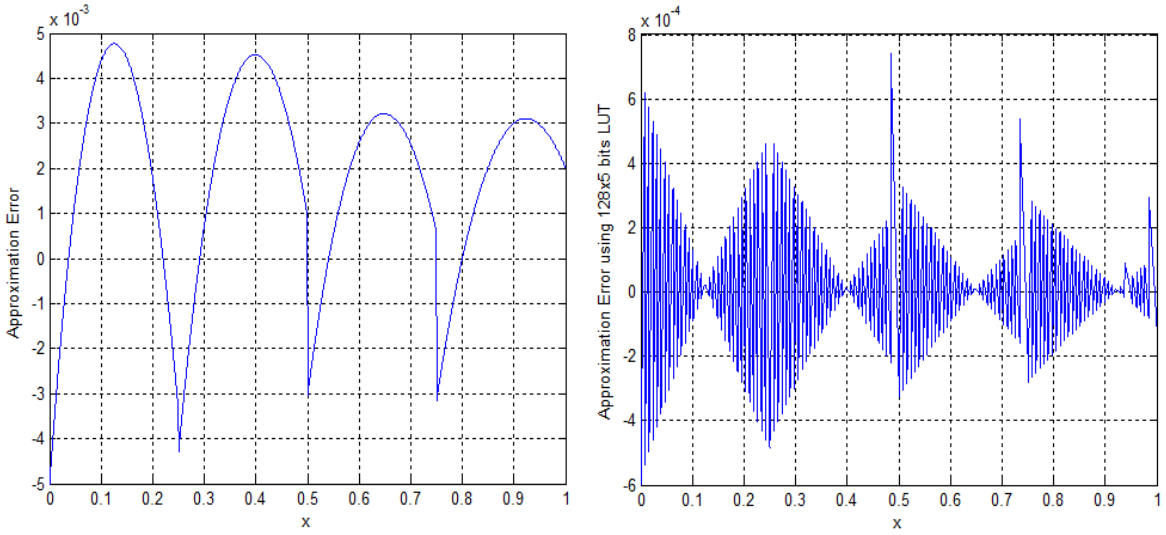
Step		Step 1	Step 2
$(0 \le x \le 0.25)$	a_1	1.2905	$2^0 + 2^{-2} + 2^{-5}$
	b_1	0.004	0.005
$(0.25 \le x \le 0.5)$	a_2	1.0316	$2^0 + 2^{-5}$
	b_2	0.0682	0.0684
$(0.5 \le x \le 0.75)$	a_3	0.8905	$2^{-1} + 2^{-2} + 2^{-3}$
	b_3	0.1418	0.1505
$(0.75 \le x < 1)$	a_4	0.7621	$2^{-1} + 2^{-2}$
	b_4	0.2379	0.248
MaxError		0.0047	0.005

Table 2 shows the results of the error analysis with the proposed method compared with other 4-segment linear approximation methods. As mention previously, *MaxError* is the maximum value of the absolute error function $|E(x)|$. *MaxError*(+) and *MaxError*(-) represent the maximum positive value and the minimum negative values of the error function $E(x)$, respectively. The mean error denotes the mean of the absolute error function $|E(x)|$. It can be seen that the proposed method achieves comparative results over other ones. Moreover, Figure 6 shows the approximation error results of the proposed method for two

cases: without LUT and with 128×5 bits LUT. Error analysis results of these two cases are presented in Table 3. It can be observed from Table 3 that the errors of the case of using 128×5 bits LUT reduce significantly compared with the case without LUT.

Table 2. Error comparison results

Method	In [23]	In [21]	Proposed
MaxError	10.1×10^{-3}	8×10^{-3}	5×10^{-3}
MaxError(+)	10.1×10^{-3}	6.1×10^{-3}	4.8×10^{-3}
MaxError(-)	-10.1×10^{-3}	-8×10^{-3}	-5×10^{-3}
Mean error	5.4×10^{-3}	2.5×10^{-3}	2.5×10^{-3}



(a) (b)

Figure 6. The error of the proposed approximation method in two cases:
(a) without LUT and (b) with 128×5 bits LUT

Table 3. Error analysis results of the proposed method for the cases without LUT and with 128×5 bit LUT

	MaxError	*MaxError*(+)	*MaxError*(-)	Mean Error
Without LUT	5×10^{-3}	4.8×10^{-3}	-5×10^{-3}	2.5×10^{-3}
With LUT	7.4×10^{-4}	7.4×10^{-4}	-6×10^{-4}	9.8×10^{-5}

The proposed hardware architecture of a logarithm generator for the 16-bit integer input N with the 4-bit integer part and 13-bit fraction part output is shown in Figure 7. The LODE (leading one detector and encoder) block generates n from the input N and n is encoded into the binary form. We use the INV (inverter) block and a modified barrel shifter to generate the fraction part x as shown in (1). Meanwhile, $\log_2(1+x)$ is approximated by the 4-segment linear approach as described above. The two most significant bits of x are used as the selection bits to choose one of the four regions in the linear piecewise approximation. The shifters operate the right shift operations of x and 3 multiplexers are used to select the terms of slope a_i. Coefficients b_i is stored in the Coef. LUT. The Error LUT stores the residual error. We can increase the LUT size to achieve the better accuracy of the approximation. However, to archive a good tradeoff of the hardware complexity with the accuracy, a 128×5-bit LUT is used. Finally, an adder is used to add these 5 components to provide the fraction value (F) of the binary logarithm result. For the control purpose, a flag (z) is used to indicate the special case of zero input.

The proposed 16-bit logarithm generator was modeled in VHDL and implemented with Xilinx FPGA device (Spartan-3E). The area results of the FPGA implementation in the number of FPGA LUTs used is shown in Table 4. It can be seen that the proposed method results in the significant improvements both in area and computation delay.

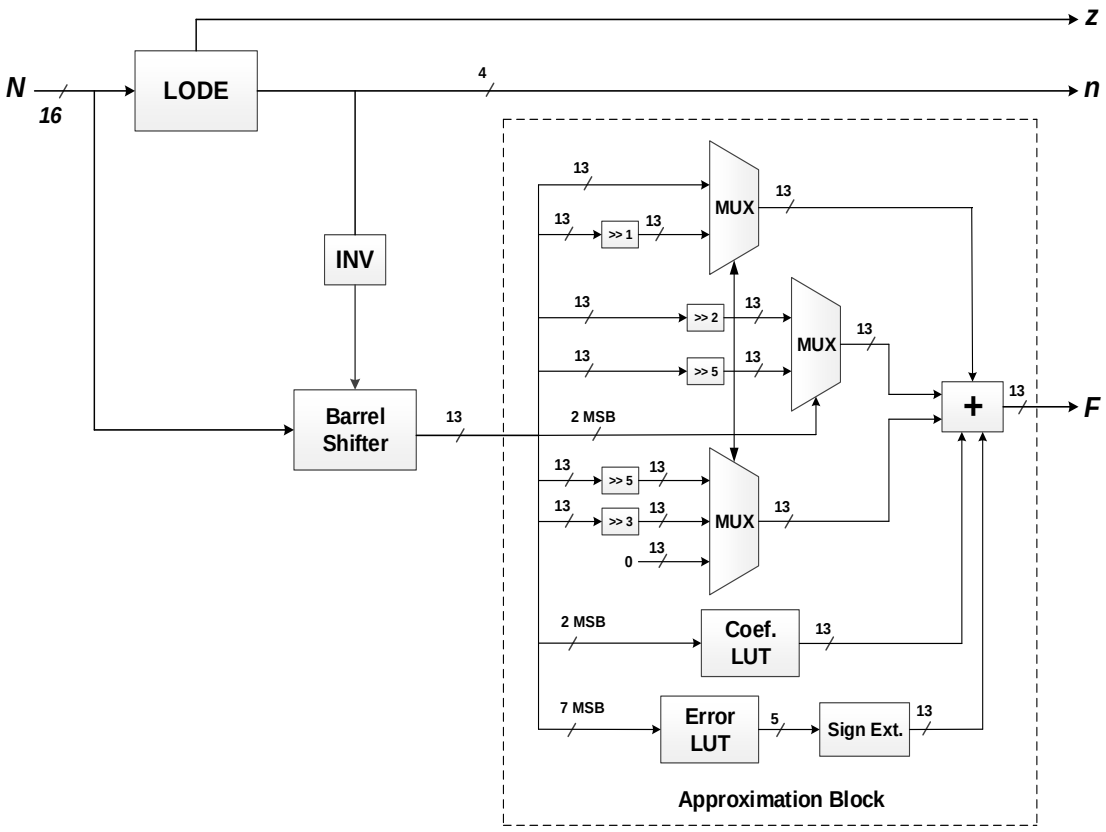


Figure 7. Hardware architecture of the 16-bit logarithm generator employing the proposed method

Table 4. FPGA hardware implementation results of the 16-bit logarithm generator with different methods

Method	FPGA LUT Count	Delay (ns)
In [22]	163	24.479
In [25]	162	24.479
Proposed	147	24.066

4. CONCLUSION

This paper presented an improved approach of modified quasi-symmetrical method to implement the low-error, low-area hardware logarithm generator for digital signal processing systems which require high-speed, real time logarithm operations. The error analysis and FPGA hardware implementation results have clarified that the proposed logarithm generator can be applied for emerging DSP systems. Especially, the proposed approximation method can reduce the approximation error and hardware complexity compared with other methods. In the future work, we will apply the proposed method for the implementation of completed speech processing system for real time applications.

REFERENCES

[1] 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.
[2] H. Kim, B.-G. Nam, J.-H. Sohn, J. H. Woo, and H.-J. Yoo, "A 231- MHz, 2.18-mW 32-bit logarithmic arithmetic unit for fixed-point 3-D graphicssystem," *IEEE J. Solid-State Circuits*, vol. 41, no. 11, pp. 2373-2381, Nov. 2006.
[3] B.-G. Nam, H. Kim, and H.-J. Yoo, "Power and area-efficient unified com-putation of vector and elementary functions for handheld 3-D graphicssystems," *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 490-504, Apr. 2008.
[4] B.-G. Nam and H.-J. Yoo, "An embedded stream processor core based on logarithmic arithmetic for a low-power 3-D graphics SoC," *IEEE J. Solid-State Circuits*, vol. 44, no. 5, pp. 1554-1570, May 2009.
[5] Van-Phuc Hoang and Cong-Kha Pham, "Efficient LUT-Based Truncated Multiplier and Its Application in RGB to YCbCr Color Space Conversion," *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, vol. E95.A, no. 6, pp. 999-1006, Jun. 2012.

[6] T. B. Juang, S. H. Chen, and H. J. Cheng, "A Lower Error and ROM-Free Logarithmic Converter for Digital Signal Processing Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, pp. 931-935, 2009.
[7] K. Parhi and Y. Liu, "Computing Arithmetic Functions Using Stochastic Logic by Series Expansion," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, pp. 1-1, 2016.
[8] K. Meher, J. Valls, T. B. Juang, K. Sridharan, and K. Maharatna, "50 Years of CORDIC: Algorithms, Architectures, and Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, pp. 1893-1907, 2009.
[9] C. Wilcox, M. M. Strout, and J. M. Bieman, "Optimizing Expression Selection for Lookup Table Program Transformation," in *2012 IEEE 12th International Working Conference on Source Code Analysis and Manipulation*, pp. 84-93, 2012.
[10] Y. L. Low and C. C. Jong, "A Memory-Efficient Tables-and-Additions Method for Accurate Computation of Elementary Functions," *IEEE Transactions on Computers*, vol. 62, pp. 858-872, 2013.
[11] Van-Lan Dao, Van-Danh Nguyen, Hai-Duong Nguyen, and Van-Phuc Hoang, "Hardware Implementation of MFCC Feature Extraction for Speech Recognition on FPGA," *Advances in Intelligent Systems and Computing, Springer*, vol. 538, pp. 248-254, Dec. 2016.
[12] Van-Tinh Nguyen, Van-Phuc Hoang, Van-Thuan Sai, Tieu-Khanh Luong, Minh-Tu Nguyen, and Han Le Duc, "A New Approach of Stochastic Computing for Arithmetic Functions in Wideband RF Transceivers," *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS2017)*, pp. 1525-1528, Aug. 2017.
[13] Van-Tinh Nguyen, Tieu-Khanh Luong, Han Le Duc, and Van-Phuc Hoang, "An Efficient Hardware Implementation of Activation Functions Using Stochastic Computing for Deep Neural Networks," *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 233-236, Sep. 2018.
[14] J. N. Mitchell, "Computer multiplication and division using a binary logarithms," *IEEE Trans. Electron. Comput*., vol. 11, pp. 512-517, 1962.
[15] Florent de Dinechin and Arnaud Tisserand, "Multipartite table methods," *IEEE Trans. Comput*., vol. 54, no. 3, pp. 319-330, Mar. 2005.
[16] S. Paul, N. Jayakumar, and S. P. Khatri, "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.
[17] 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, 1965.
[18] 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, 1970.
[19] S. Sangregory, C. Brothers, D. Gallagher, and R. Siferd, "A fast low power logarithm approximation with CMOS VLSI implementation," *Proc. 42nd Midwest Symp. Circuits Syst*., vol. 1, pp. 388-391, 1999.
[20] 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, 2003.
[21] T. B. Juang, S. H. Chen, and H. J. Cheng, "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.
[22] 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.
[23] Van-Phuc Hoang and Cong-Kha Pham, "Novel Quasi-Symmetrical Approach for Efficient Logarithmic and Anti-logarithmic Converters," *VDE-IEEE 8th Conference on Ph.D. Research in Microelectronics & Electronics (PRIME2012)*, pp.111-114, Jun. 2012.
[24] Van-Phuc Hoang, Xuan-Tien Do, and Cong-Kha Pham, "An Efficient ASIC Implementation of Logarithm Approximation for HDR Image Processing," *Proc. 2013 International Conference on Advanced Technologies for Communications (IEEE ATC)*, pp. 535-539, Oct. 2013.
[25] 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.

BIOGRAPHY OF AUTHOR

Minh-Hong Nguyen was born in 1960. He received PhD degree in Automation Engineering from Le Quy Don Technical University, Hanoi, Vietnam. He is working as the Deputy Director of Center for Automation Engineering, Le Quy Don Technical University. His research interests include automation engineering and control theory, embedded systems for Internet of Things, VLSI architecture for digital signal processing, digital circuits and systems.