An Optimized Implementation of Logarithm Hardware Generator for Digital Signal Processing

Van-Thuan Sai and Van-Phuc Hoang
Le Quy Don Technical University, 236 Hoang Quoc Viet Str., Hanoi, Vietnam
Email: saivanthuan@gmail.com; phuchv@mta.edu.vn

***Abstract*— This paper presents an optimized hardware approximation of logarithm function for digital signal processing systems. The proposed method combines the multi-segment linear approximation with a small look-up table and an optimization algorithm to trade-off the computation speed with the hardware complexity. As a result, an efficient logarithm generator with simple structure and high computation speed is presented. The implementation results are also discussed.**

Keywords— logarithm generator; ASIC; CMOS; low power

I. INTRODUCTION

In many digital signal processing (DSP) applications, e.g. 3-D graphics [1], a lot of complex operations are required such as multiplication, division, square root, power and so on. By using logarithmic scaled domain, these operations can be simplified significantly. For example, the multiplication can be performed by addition while square root can be replaced by shifting in the logarithmic domain. Also, the logarithmic number system (LNS) which is based on logarithmic scale presentation and arithmetic has been shown to be an alternative for floating-point for precisions up to 32-bit because of its merit in above complex operations [2]. LNS is also promisingly suitable for the low power applications [3]. However, some operations such as addition and subtraction in LNS are more complicated [1], resulting in its low popularity in current systems. Therefore, it is reasonable to find a proper combination of LNS and conventional binary number system called hybrid number system (HNS) to take the advantages of both above number systems [1]. In HNS, the logarithmic converters (LOGC) and anti-logarithmic converters (ALOGC) are essential components. As an example implementation of 3-D graphic processors using HNS [1], the area of LOGC and ALOGC consumes 64% of total chip area. It means that efficient LOGCs and ALOGCs can result in a significant impact on overall performance of HNS-based DSP systems.

Moreover, many real-time DSP applications such as wireless communication systems require the high-performance, low-complexity and low-power consumption logarithm generators. Therefore, the objective of this paper is an efficient architecture for this logarithm generator which can be applied for LNS/HNS systems and general DSP applications. The main contribution of this paper is that a new optimized difference method is proposed to derive an efficient architecture for low complexity and high speed logarithm generator.

The rest of this paper is organized as follows. Section II provides the basics of binary logarithm hardware approximation and section III presents the optimized logarithm approximation method. Sections IV and V show the error analysis and implementation results, respectively. Finally, section VI concludes the paper.

II. BINARY LOGARITHM APPROXIMATION

Without loss of generality, consider the binary (base-2) logarithm of an unsigned number N for LOGC in which N can be expressed as:

$$N = z_k \ldots z_3\, z_2\, z_1\, z_0.\, z_{-1}\, z_{-2}\, z_{-3} \ldots z_j \quad (1)$$

N can be rewritten as:

$$N = \sum_{i=j}^{k} z_i 2^i \quad (2)$$

In which $z_i = 0$ or 1, and z_k is MSB so that $z_k = 1$. As a result, N can be rewritten as:

$$N = 2^k (1 + \sum_{i=j}^{k-1} 2^{i-k} z_i) = 2^k (1 + x); \qquad x = \sum_{i=j}^{k-1} 2^i z_i \quad (3)$$

Since $k \geq j$, the range of fraction part (x) is $0 \leq x < 1$.

According to (3), the binary (base-2) logarithm can be computed as:

$$\log_2 N = k + \log_2 (1 + x) \quad (4)$$

In (4), k, called the characteristics of N, corresponds to the position of the most significant '1' bit in the binary representation of N. Exploiting this mathematic property, $\log_2 N$ can be computed by detecting the most significant '1' bit and approximating $\log_2(1+x)$. Therefore, many researches focused on efficient methods for $\log_2(1+x)$ approximation.

In [4], J. N. Mitchell presented the simple approximation as follows:

$$\log_2 (1 + x) \approx x \quad (5)$$

Figure 1 depicts the error function due to this method as:

$$E_L(x) = \log_2 (1 + x) - x \quad (6)$$

E_L is called Mitchell error for logarithmic approximation. The maximum value of E_L is 0.08639, and the accuracy is only 3.5 bits which is too low for many DSP applications. Therefore, there have been many researches focusing on more efficient methods for this approximation.

In the linear piecewise (or multi-segment) approximation method, the entire range of x in $[0,1)$ is divided into s intervals.

978-1-5090-1801-7/16/$31.00 ©2016 IEEE

In each interval, the logarithm function is approximated by a polynomial. For high-speed applications, linear (first order) polynomial is often used since it requires simple multiplier, adder and small look-up table (LUT) storing linear segment coefficients. Moreover, increasing the number of segments (s) can lead to the reduction of the approximation error. However, it will also result in higher hardware complexity. The approximation method with 2 regions (or two-segment implementation) has been proposed in [5] and [6]. On the other hand, authors in [7]-[10] presented the 4-segment implementation scheme. Besides, authors in [11] investigated the approximation methods using 2, 3 or 6-segments. However, in the above papers, the authors have not presented the method to choose suitable coefficients of the linear segments. Moreover, the optimization method has not been mentioned as well. In references [12]-[13], some piecewise shift-add linear approximation methods with different numbers of segments and different forms of linear difference functions were presented.

M.B. Sullivan *et al.* [14] proposed the truncated logarithmic approximation method which can improve the resource usage efficiency. However, its accuracy is very low due to the simplicity of Mitchell method which is the base of this paper.

Therefore, in this paper, we will present an efficient logarithm design based on 4-segment linear approximation (s=4). Also, it is combined with a small LUT for high accuracy logarithm approximation. Optimized hardware architecture for 16-bit logarithm generator is also proposed.

III. OPTIMIZED LOGARITHM APPROXIMATION METHOD

With the purpose of designing a high-speed, low complexity logarithm generator, we propose an approximation method based on Mitchell equation in which $\log_2(1+x)$ is approximated by multiple linear segments. The coefficient values are chosen with power-of-two form (2^i) so that the multiplications can be simplified into simple shifting operations. Moreover, the full range of x is divided into 4 intervals (s = 4) to make the selection circuit simpler. The approximation can be written as (7).

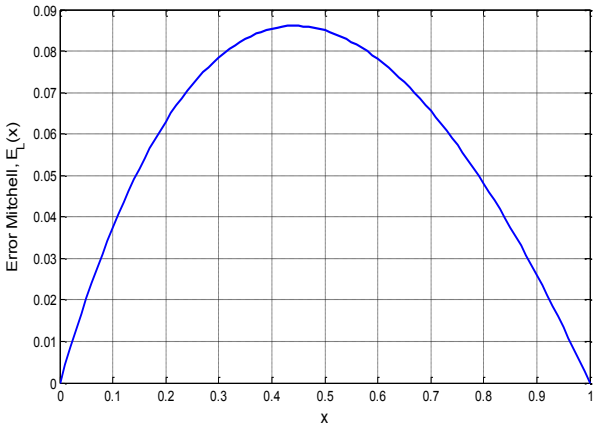


Fig. 1. Approximation of error function E_L with Mitchell method.

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

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

As mentioned above, if a_i value is in the form of 2^i, the multiplication $a_i x$ is simplified to a shift. However, to achieve a good tradeoff between the linear approximation part and LUT, we propose an algorithm to find optimal a_i and b_i values in the following expressions:

$$\begin{cases} a_i = \sum_{k=1}^{M} p_k 2^{-n}, & p_k \in \{-1,0,1\} \\ b_i = B_i 2^{-N} & , \quad B_i, N \in Z \end{cases} \tag{8}$$

Then, the approximation is expressed as:

$$E(x) = \log_2(1+x) - (a_i x + b_i) \tag{9}$$

An LUT is also used to further reduce the approximation error. Figure 2 presents the proposed approximation method basing on the difference method [10]. In this method, x is the input, $D(x)$ is a simple approximation function (often a linear function) and an LUT is used to store the error between the function to be computed ($\log_2(1+x)$ in this case) and $D(x)$. An adder is required to provide the final result. In theory, increasing the LUT size can improve the final approximation accuracy, but it also leads to the high circuit complexity.

Therefore, to reduce the final approximation error and keep the LUT size small enough, the proposed optimization algorithm is to find a_i and b_i values to minimize the approximation error. The algorithm including two steps: the first one is to find the optimal values of a_i and b_i by MMSE (*minimum mean square error*) method and the second step is to re-assign a_i and b_i according to (8).

Step 1: Find a_i and b_i values by MMSE method for the approximation in the range of $[x_1, x_2]$:

$$\overline{E^2} = \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} \{\log_2(1+x) - (ax+b)\}^2 dx; \; 0 \le x_1 < x_2 < 1 \tag{10}$$

a_i and b_i values are derived using equation (10).

$$\begin{cases} \dfrac{\partial \overline{E^2}}{\partial a} = 0 \\ \dfrac{\partial \overline{E^2}}{\partial b} = 0 \end{cases} \Leftrightarrow \begin{cases} \int_{x_1}^{x_2} -2x\{\log_2(1+x) - (ax+b)\} dx = 0 \\ \int_{x_1}^{x_2} -2\{\log_2(1+x) - (ax+b)\} dx = 0 \end{cases} \tag{11}$$

Table I presents the results of step 1 optimization for (a_{step1}, b_{step1}) with 4 even intervals of x in $[0,1)$.

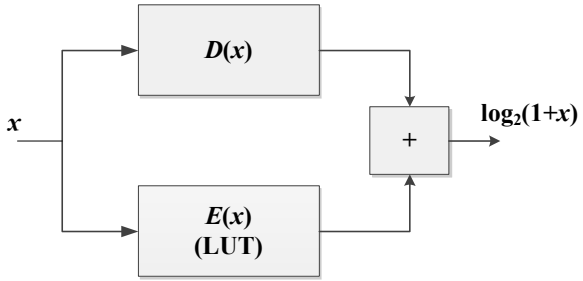


Fig. 2. The linear difference method with LUT-based correction.

Step 2: Find the values of a_i and b_i which satisfy (8) as follows. Firstly, a_i is assigned to the adjacent value of a_{step1} and satisfies (8). To achieve a tradeoff between hardware complexity and approximation accuracy, we choose a_i which is sum of two power-by-two values (M = 2). For example, in the first segment, $a_1 = 2^0 + 2^{-2} \approx 1.2856$. Then, the algorithm finds the optimal b_i value satisfying (8) as shown in Fig. 3. The optimal values of N and b_i in (8) are selected to minimize the approximation error.

After the re-assigning a_i values and finding the coefficients b_i according above algorithm flowchart, the values of a_i, b_i after the second step (a_{step2}, b_{step2}) is shown in Table II. The optimization algorithm is performed by Matlab software. As a result, the approximation function is presented in (12).

$$\log_2(1+x) \approx \begin{cases} (2^0+2^{-2})x + 2^{-7}, & 0 \le x < 0.25 \\ (2^0+2^{-4})x + 2^{-4}, & 0.25 \le x < 0.5 \\ (2^0-2^{-3})x + 77\times 2^{-9}, & 0.5 \le x < 0.75 \\ (2^{-1}+2^{-2})x + 2^{-2}, & 0.75 \le x < 1 \end{cases} \quad (12)$$

TABLE I. OPTIMAL VALUES OF a_i AND b_i AFTER STEP 1.

x	a_{step1}	b_{step1}
$0 \le x < 0.25$	1.2856	0.0062
$0.25 \le x < 0.5$	1.0510	0.0633
$0.5 \le x < 0.75$	0.8890	0.1435
$0.75 \le x < 1$	0.7701	0.2320

TABLE II. OPTIMAL VALUES OF a_i AND b_i AFTER STEP 2.

x	a_{step2}	b_{step2}
$0 \le x < 0.25$	2^0+2^{-2}	2^{-7}
$0.25 \le x < 0.5$	2^0+2^{-4}	2^{-4}
$0.5 \le x < 0.75$	2^0-2^{-3}	7.7×2^{-8}
$0.75 \le x < 1$	$2^{-1}+2^{-2}$	2^{-2}

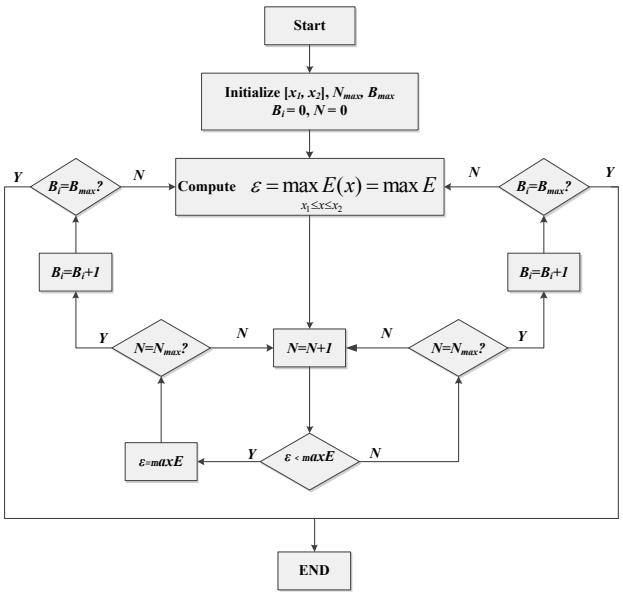


Fig. 3. Parameter optimization algorithm.

IV. APPROXIMATION ERROR ANALYSIS

We have analyzed the approximation error of proposed method and that reported in [10], [12] and [13]. Author in [13] also used the 4-segment linear approximation. To evaluate the error of the approximation method, we calculate the difference between the exact value of the function and the result of the function approximation (called ε) as shown in (13).

$$\varepsilon = \log_2 N - \log_2 N^* = \log_2(1+x) - \log_2(1+x)^* \quad (13)$$

In (12), $\log_2 N^*$ is the approximated value of $\log_2 N$. The difference ε may be a positive or a negative value. The maximum absolute value of the positive and negative deviations respectively is denoted as $\varepsilon^+_{\max}$ and $\varepsilon^-_{\max}$, these parameters are used to evaluate the approximation error of the proposed method. The maximum difference $\varepsilon_{\max} = \max\{\varepsilon^+_{\max}, \varepsilon^-_{\max}\}$ is also considered because it determines the LUT size in the proposed generator. The errors of the proposed method are shown in Fig. 4a and Fig. 4b for the cases of without LUT and with a 5×128-bit LUT, respectively.

Another approximation error is the percentage error is defined by the ratio of the error over the exact value of the $\log_2 N$ as the formula (14).

$$e(N) = \frac{\log_2 N - \log_2 N^*}{\log_2 N} = \frac{\log_2(1+x) - \log_2(1+x)^*}{k + \log_2(1+x)} \quad (14)$$

Figures 5a and 5b show the percentage error results of the proposed method for the cases of approximation without LUT and with the LUT, respectively.

Table III depicts the evaluation results for the approximation error of the proposed method. The notations max $e(N)^+$ and max $e(N)^-$ correspond the maximum absolute value of $e(N)$ in the cases of positive and negative errors. The values of mean error and average percent error are also calculated and compared. The values of the maximum difference, maximum percentage errors are similar to the results in [10], [12], [13]. Especially, the values of mean error and average percentage error are reduced significantly compared with the results in [10], [12] and [13].

TABLE III. APPROXIMATION ERROR COMPARISON OF THE PROPOSED METHOD WITH OTHER METHODS.

Method	**E. L. Hall [12]**	**D. De Caro [13]**	**H. Phuc [10]**	**Proposed**
$\varepsilon_{\max}$	8.1×10^{-3}	9.8×10^{-3}	10.1×10^{-3}	$\mathbf{8.8\times10^{-3}}$
$\varepsilon^+_{\max}$	8.1×10^{-3}	9.8×10^{-3}	10.1×10^{-3}	$\mathbf{6.3\times10^{-3}}$
$\varepsilon^-_{\max}$	-2.3×10^{-3}	-6.8×10^{-3}	-10.1×10^{-3}	$\mathbf{-8.8\times10^{-3}}$
Mean Error	1.2×10^{-3}	2.1×10^{-3}	2.1×10^{-3}	$\mathbf{1.76\times10^{-4}}$
max $e(N)^+$	0.13%	0.18%	0.31%	**0.18%**
max $e(N)^-$	-0.78%	-0.43%	-0.4%	**-0.78%**
Average Percentage Error	5.81×10^{-5}	2.03×10^{-4}	2.48×10^{-4}	$\mathbf{4.98\times10^{-5}}$

V. IMPLEMENTATION RESULTS

Figure 6 presents the architecture for the proposed logarithm generator with 16-bit integer input N and the result includes 4-bit integer part and 13-bit fraction part of the logarithmic converter output. The leading-one-detector-and-encoder (LODE) is to compute the characteristic k and encodes it [10]. The fraction part x is generated by the inverter block (INV) and a modified barrel shifter as shown in (3). The two significant bits are used to select one of the four regions of the linear piecewise approximation. The multiplexer is used to select the right shift operation of the input bits sequence to create the slope (a_i), a small LUT contains coefficients of the four segments (Cof. LUT). One LUT contains the error compensation values (Error LUT). The higher the LUT size, the better approximation accuracy can be achieved. However, to tradeoff the hardware complexity and accuracy, an LUT with the size of 5×128 bits is used. The approximation block provides the output as the fraction part (F) of logarithm computation result, presented as $\log_2(1+x)$.

The proposed architecture was designed with VHDL and implemented on Spartan-3E Xilinx FPGA device. The area in the FPGA implementation is calculated by the number of FPGA LUTs used. The area-delay product (ADP) results are also used to compare the overall performance of the converter. Table IV shows the implementation results on FPGA of the proposed architecture compared with the results in [9], [12], [13]. It is shown that the proposed method can lead to the ADP reduction of 36% compared with [12], by 14% and 13%, compared with [9] and [13], respectively.

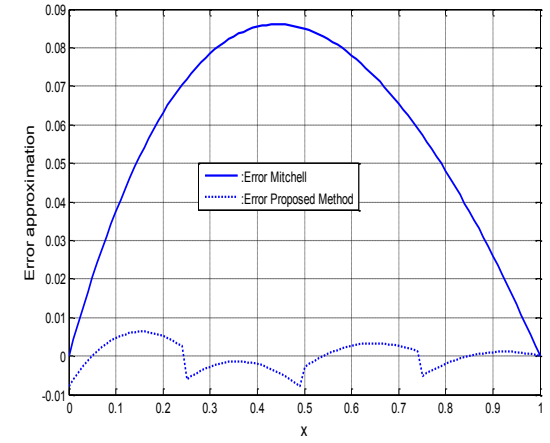
Moreover, Table V presents the implementation results on an ultra-low power SOTB CMOS 65nm ASIC technology using Synopsys Design Compiler tool. It can be seen that the proposed logarithm generator achieves significant improvements in area, computation speed and power consumption. The estimated power consumption is based on the synthesized netlists. Figure 7 is the netlist of proposed logarithm generator and Fig. 8 depicts the flow for applying the proposed logarithm generator IP core for DSP systems in both FPGA and ASIC approaches.

TABLE IV. FPGA IMPLEMENTATION RESULTS OF 16-BIT LOGARITHM GENERATOR USING DIFFERENT METHODS.

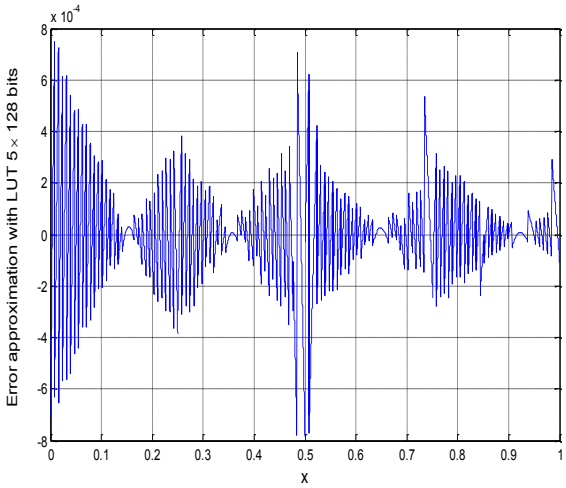
Method	FPGA LUTs	Delay (ns)	ADP (×10^3)
Guitierz [9]	163	24.479	3.989
Hall [12]	188	28.574	5.371
De Caro [13]	162	24.479	3.965
Proposed	**149**	**23.066**	**3.436**

TABLE V. IMPLEMENTATION RESULTS IN A 65NM SOTB CMOS ASIC LIBRARY OF 16-BIT LOGARITHM GENERATOR USING DIFFERENT METHODS.

Method	Area (×10^{3}µm^2)	Delay (ns)	ADP (×10^3)	Dynamic power (µW)	Leakage power (µW)
Guitierz [9]	42.9	12.2	523.4	17.4	0.30
Hall [12]	54.1	13.2	714.1	22.7	0.39
De Caro [13]	40.3	12.0	483.6	14.7	0.28
Proposed	**27.2**	**11.5**	**312.8**	**9.22**	**0.18**



(a)



(b)

Fig. 4. Error of approximately: (a) without LUT, (b) using the LUT.

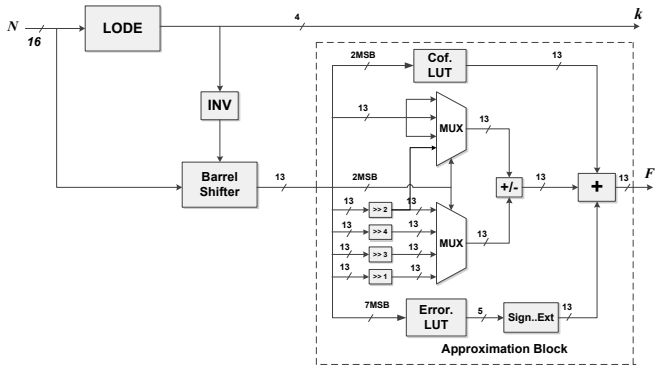


Fig. 6. Proposed architecture for 16-bit logarithm generator.

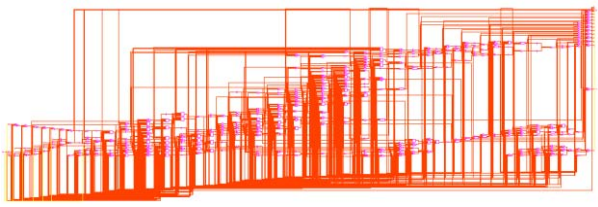


Fig. 7. Synthesized netlist of proposed 16-bit logarithm generator using a 65nm SOTB CMOS ASIC library.

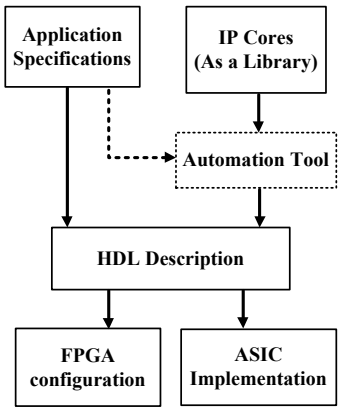


Fig. 8. System design flow to ultilize the proposed logarithm generator IP core.

VI. CONCLUSIONS

The proposed logarithm approximation method is based on Mitchell's method, and uses an optimized multi-segment linear approximation in which the slope values of the segments of the form are in the form of sum of powers of two. To increase the approximation accuracy, a 5×128 bit LUT is used. Based on that, the architecture of a logarithmic converter for the 16-bit unsigned integer was also proposed and implemented on both FPGA and ASIC hardware platforms. In the future work, we will implement a logarithm generator with the input data having different formats, apply the proposed method of combining LUT and piecewise linear approximations for the implementation of other functions. Moreover, we will apply the proposed logarithm generator for a speech recognition chip in ASIC technology.

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] J. Detrey and F. D. Dinechin, "A VHDL library of LNS operators," in Proc. 37th Asilomar Conference on Signals, Systems & Computers, vol. 2, pp. 2227-2231, Nov. 2003.

[3] V. Paliouras and T. Stouraitis, "Low-power properties of the logarithmic number system," in Proc. 15th IEEE Symposium on Computer Arithmetic, pp. 229-236, Jun. 2001.

[4] J. N. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electron. Comput., vol. 11, no. 11, pp. 512–517, Aug. 1962.

[5] S. L. SanGregory, R. E. Siferd, C. Brother, and D. Gallagher, "A fast, low-power logarithm approximation with CMOS VLSI implementation," in Proc. IEEE MWSCAS, pp. 388-391, Aug. 1999.

[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 Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 12, pp. 931-935, Dec. 2009.

[7] M. Combet, H. V. 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] S. L. SanGregory, R. E. Siferd, C. Brother, and D. Gallagher, "A fast, low-power logarithm approximation with CMOS VLSI implementation," in Proc. IEEE MWSCAS, vol. 1, pp. 388-391, Aug. 1999.

[9] 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.

[10] Van-Phuc Hoang and Cong-Kha Pham, "Novel Quasi-Symmetrical Approach for Efficient Logarithmic and Anti-logarithmic Converters," in Proc. VDE-IEEE 8th Conference on Ph.D. Research in Microelectronics & Electronics (PRIME2012), pp. 111-114, Jun. 2012.

[11] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," IEEE Trans. Comput., vol. 52, no. 11, pp. 1421-1433, Nov. 2003.

[12] E. L. Hall, D. D. Lynch, and S. J. Dwyer, III, "Generation of products and quotients using approximate binary logarithms for digital filtering applications," IEEE Trans. Electron. Comput., vol. C-19, no. 2, pp. 97-105, Feb. 1970.

[13] 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.

[14] M.B. Sullivan and E.E. Swartzlander, "Truncated Logarithmic Approximation," *in Proc. 2013 21st IEEE Symposium on Computer Arithmetic (ARITH)*, pp. 191-198, Apr. 2013.