ICIC Express Letters
Part B: Applications
Volume **7**, Number **4**, April **2016**

ICIC International ©2016 ISSN 2185-2766
pp. **813–818**

AREA-EFFICIENT AND ROM-FREE DECIMAL-TO-DECIMAL ANTILOGARITHMIC CONVERTERS

Chao-Tsung Kuo[1] and Tso-Bing Juang[2]

[1]Department of Electronic Engineering
National Quemoy University
No. 1, University Rd., Jinning Township, Kinmen County 892, Taiwan
ctkuo@nqu.edu.tw

[2]Department of Computer Science and Information Engineering
National Pingtung University
No. 51, Min-Sheng E. Rd., Pingtung 900, Taiwan
tsobing@mail.nptu.edu.tw

Received September 2015; accepted November 2015

Abstract. *In this paper, area-efficient and ROM-free decimal-to-decimal antilogarithmic converters are proposed. By employing efficient non-uniform linear approximation with shift-and-add schemes, the total approximation errors of our proposed decimal antilogarithmic converters could achieve 0.049, 0.0194 and 0.0098 for 8-region, 16-region and 28-region approximations respectively, where error analysis is simulated by MATLAB. Meanwhile, our proposed decimal antilogarithmic converters are area-efficient and simple enough so that they are suitable for real VLSI circuit implementations. Our proposed decimal-to-decimal antilogarithmic converters can be applied to commercial applications and signal processing applications that can save computation efforts.*
Keywords: Decimal antilogarithmic converter, Decimal computation arithmetic, VLSI design, Digital circuits

1. **Introduction.** Image segmentation and digital signal processing have been widely used in mobile calculations. These related technologies require many complex arithmetic calculations such as multiplication, division, squaring and square-root. Among many methods for simplifying the computation overhead, Logarithmic Number Systems (LNS)-based operations can be used to transform multiplications/divisions to additions/subtractions, squaring and square root evaluation to left and right shifts, respectively, which can simplify tremendous computation efforts and hardware cost. There are three conversion parts required in LNS-based circuits, which include logarithmic converters that convert inputs into logarithmic numbers, the corresponding simple calculation unit such as additions/subtractions and antilogarithmic converters to convert back to the original representations. Consequently, logarithmic converters and antilogarithmic converters will be the important factors in designing LNS-based conversion system. Several previous LNS-based works have been proposed for the base-2 logarithmic binary conversion system [1-3,5,10-16]. Meanwhile, decimal logarithmic and antilogarithmic technologies [4,6-9,17,18] have become very useful in many applications, such as Internet-based operation, e-commerce, optics, the PH measurement in chemistry, tax calculation and financial analysis. Nowadays, more researchers pay more attention about the inclusion of decimal floating point (DFP) operation in the latest IEEE754-2008. So far, to the best of our knowledge, only a decimal antilogarithmic method [4] based on look-up-table is proposed previously in the literature. However, this method will demand higher area costs as the bit-width of inputs increases. To improve the complex hardware costs of look-up table method, in this paper, we will propose area-efficient and ROM-free decimal antilogarithmic conversion using straight-line approximation with multiple regions, including 8-, 16- and 28-regions, which

can perform area-efficient approximations under tolerant error ranges. The corresponding decimal antilogarithmic conversions are performed using simple shift-and-add operations on a piecewise-linear approximation of the input data.

The remainder of this paper is organized as follows. In Section 2, we will review decimal mathematic and previous decimal-to-decimal antilogarithmic conversion methods. In Section 3, our proposed decimal-to-decimal antilogarithmic converters using shift-and-add schemes are described. Error analysis, comparisons and VLSI hardware implementations are given in Section 4. Finally, we will conclude this work in Section 5.

2. **Decimal Mathematic and Previous Decimal-to-Decimal Antilogarithmic Converters.** In decimal mathematic and decimal antilogarithmic conversion, suppose X is any decimal number which can be expressed in Equation (1). Take $\log_{10}(X)$ as decimal logarithmic value of X, which can be represented by Equation (2), where k and m are the integer part and fraction part, respectively. The decimal antilogarithmic of input Y and the approximated decimal antilogarithmic conversion are shown in Equation (3) and Equation (4), respectively. For example, letting $Y = 1.5783$, here we take $10^{(1.5783)} = 10^1 \times 10^{(0.5783)} \fallingdotseq 10^1 \times (am + b)$, where, $0 =< m < 1$, a is the value of corresponding slope and b is the intercept, respectively. For linear-piecewise approximation, we can take $am + b$ to approximate the actual value of 10^m. Since the value of m is decimal-based, the values of a and b will be used in binary representations for circuit implementations.

$$X = \sum_{i=j}^{k} 10^i z_i = 10^k \sum_{i=j}^{k} 10^{i-k} z_i, \text{ where } z_i = 0 \text{ to } 9, \ z_i \text{ is integer} \tag{1}$$

$$Y = \log_{10}(X) = \log_{10}\left(10^k \sum_{i=j}^{k} 10^{i-k} z_i\right) = k + \log_{10}\left(\sum_{i=j}^{k} 10^{i-k} z_i\right) = k + m, \ 0 \le m < 1 \tag{2}$$

$$\text{Anti}\log_{10}(Y) = 10^Y = 10^{k+m} = 10^k 10^m \tag{3}$$

Taking the linear approximations of $10^{(Y)}$,

$$\begin{gathered}\text{Anti}\log_{10}(Y) = 10^Y = 10^{(k+m)} = 10^k \cdot 10^m \approx 10^k \cdot (am + b), \\ 0 \le am + b < 1, \ 0 \le m < 1\end{gathered} \tag{4}$$

As for decimal-to-decimal antilogarithmic conversions, only one study can be found in the literature. Chen's [4] method uses look-up table proposed in 2008, using 45 segmentations of linear approximation to approximate the 10 power of Y. The coefficients of Chen's linear approximation are all stored in the look-up-tables (i.e., ROM). Chen's method can achieve smaller errors but it will suffer from complex area consumptions due to ROM-based methods. To achieve area-efficient and ROM-free under tolerable error range of decimal antilogarithmic conversions, we will propose an area-efficient and ROM-free decimal antilogarithmic converter using 8-, 16- and 28-region shift-and-add linear approximations to approximate the actual values of 10^m.

3. **Proposed Area-Efficient and ROM-Free Decimal-to-Decimal Antilogarithmic Converters.** Similar to Mitchell's method proposed in [1], m is the input value of antilogarithmic values. The curves of 10^m and $y = m$ are shown in Figure 1, where the differences between actual curve 10^m and $y = m$ are the compensation values that are needed for obtaining the approximations with low errors. The mathematical derivation of proposed non-uniform piecewise-linear approximation is shown in the following Equation (5) to Equation (8). In Equation (6), $m_{dMSBits}$ is denoted as the first d most significant bits after the point of m.

$$\text{Anti}\log(Y) = 10^Y = 10^k \cdot 10^m \approx 10^k \cdot (m + \textit{compensation value}) \tag{5}$$

$$\begin{gathered}\text{Anti}\log_{10}(Y) = 10^{Y} \approx 10^{k} \cdot (a'm + b') = 10^{k} \cdot (m + am_{dMSBits} + b), \\ 0 \le a'm + b' < 1,\ 0 \le m < 1\end{gathered} \tag{6}$$

$$compensation\ value = (am_{dMSBits} + b),\ 0 \le m < 1 \tag{7}$$

$$Approximation\ Error = 10^{k} \cdot [10^{m} - (m + am_{dMSBits} + b)] \tag{8}$$

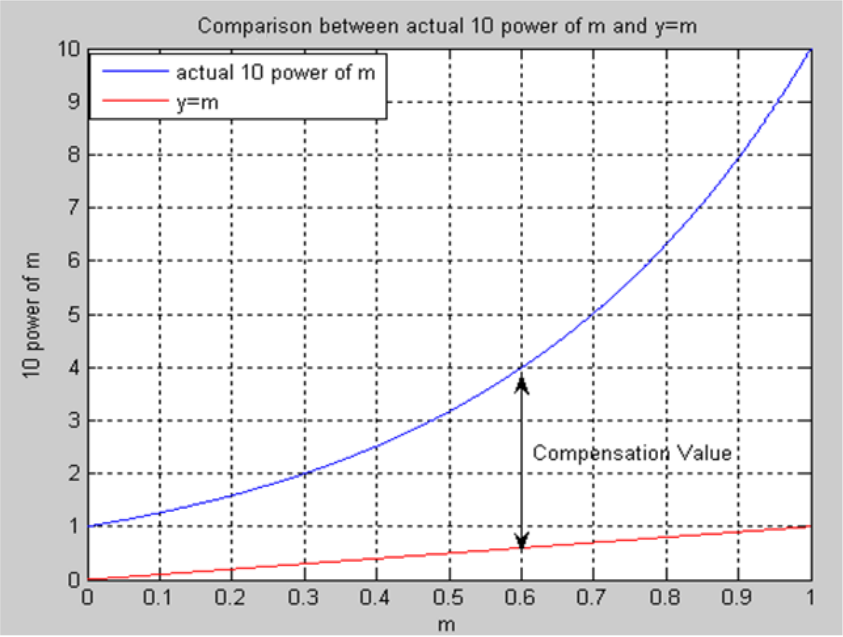

FIGURE 1. The comparison between actual curve 10 power of m and $y = m$

As for Equation (8), the key for obtaining the antilogarithmic conversion with lower approximation errors is to use the values of a and b that can lead to smaller errors as possible. Therefore, the concept of our method is to divide the entire range of input into N regions, and to find the values of a_i and b_i to make the approximation error for region i, where $1 \le i \le N$. For each region, we can design the corresponding compensation circuit which is to perform the corresponding shift-and-add operations for the corresponding conversions. In our method, the value of m is between 0 and 1, which can be also divided into N ranges and each of them denotes the range for decimal antilogarithmic approximation in each region. Taking $N = 8$ for example, m is divided into $[0, 0.203125)$, $[0.203125, 0.375)$, $[0.375, 0.515625)$, $[0.515625, 0.625)$, $[0.625, 0.734375)$, $[0.734375, 0.828125)$, $[0.828125, 0.921875)$ and $[0.921875, 1.00)$, respectively.

The total error is defined as the sum of the maximum positive errors and the maximum absolute negative errors. The algorithm of obtaining optimal coefficients a and b in the proposed non-uniform 8-, 16- and 28-region piecewise-linear decimal antilogarithmic converter is shown in Figure 2. After obtaining the optimal values of a_i and b_i for each region, the fine tune process will be manually adjusted to get the minimum total errors to the entire range. All values of a_i and b_i in the proposed algorithm are obtained by Matlab software. Considering the feasible implementation on digital circuits, the values of a and b are set to be the combination of power of 2.

Using our proposed algorithm shown in Figure 2, the optimal values of a_i and b_i for different regions of decimal antilogarithmic conversion can be easily obtained. Therefore, 8-, 16- and 28-region decimal antilogarithmic conversion can be expressed in Equation (9) and Tables 1, 2 and 3, respectively, where $m_{11MSBits}$ is denoted as the first 11 most significant bits after the point of m.

$$10^{m} \approx Y'_{proposed} = m + am_{11MSBbits} + b \tag{9}$$

[Input] Given N sub-regions,
For each sub-region i do
For $a_i = 0$ to 20 step 1/128 do
For $b_i = -40$ to 40 step 1/2048 do
Find the values of a_i and b_i which can produce the minimum approximation errors
End for
End for
End for
Set the values of a_i and b_i into the combination of power of 2.
[Output] N pairs for a_i and b_i.

FIGURE 2. The proposed algorithm for obtaining the corrected values for each sub-region

TABLE 1. Parameters of proposed 8-region linear decimal antilogarithmic converter

Items	Partition Region	a	b	Items	Partition Region	a	b
1	$[0, 0.203125)$	31/16	127/128	**5**	$[0.625, 0.734375)$	161/16	$-347/128$
2	$[0.203125, 0.375)$	15/4	9/16	**6**	$[0.734375, 0.828125)$	413/32	$-307/64$
3	$[0.375, 0.515625)$	175/32	$-17/256$	**7**	$[0.828125, 0.921875)$	521/32	$-243/32$
4	$[0.515625, 0.625)$	123/16	$-123/128$	**8**	$[0.921875, 1)$	1293/64	$-1435/128$

TABLE 2. Parameters of proposed 16-region linear decimal antilogarithmic converter

Items	Partition Region	a	b	Items	Partition Region	a	b
1	$[0, 0.125)$	105/64	1	**9**	$[0.6875, 0.75)$	357/32	$-447/128$
2	$[0.125, 0.25)$	41/16	227/256	**10**	$[0.75, 0.8125)$	13	$-625/128$
3	$[0.25, 0.34375)$	115/32	5/8	**11**	$[0.8125, 0.84375)$	235/16	$-25/4$
4	$[0.34375, 0.4375)$	149/32	33/128	**12**	$[0.84375, 0.875)$	125/8	$-451/64$
5	$[0.4375, 0.5)$	185/32	$-29/128$	**13**	$[0.875, 0.90625)$	265/16	$-513/64$
6	$[0.5, 0.5625)$	219/32	$-97/128$	**14**	$[0.90625, 0.9375)$	295/16	$-1223/128$
7	$[0.5625, 0.625)$	129/16	$-185/128$	**15**	$[0.9375, 0.96875)$	20	$-353/32$
8	$[0.625, 0.6875)$	19/2	$-601/256$	**16**	$[0.96875, 1)$	683/32	$-1579/128$

TABLE 3. Parameter of proposed 28-region linear decimal antilogarithmic converter

Items	Partition Region	a	b	Items	Partition Region	a	b
1	$[0, 0.09375)$	101/64	255/256	**15**	$[0.75, 0.78125)$	801/64	$-2311/512$
2	$[0.09375, 0.1875)$	35/16	965/1024	**16**	$[0.78125, 0.8125)$	435/32	$-343/64$
3	$[0.1875, 0.25)$	45/16	211/256	**17**	$[0.8125, 0.828125)$	231/16	$-387/64$
4	$[0.25, 0.3125)$	55/16	171/256	**18**	$[0.828125, 0.84375)$	471/32	$-21/8$
5	$[0.3125, 0.375)$	263/64	29/64	**19**	$[0.84375, 0.859375)$	485/32	$-851/128$
6	$[0.375, 0.4375)$	155/32	23/128	**20**	$[0.859375, 0.875)$	65/4	$-971/128$
7	$[0.4375, 0.5)$	185/32	$-59/256$	**21**	$[0.875, 0.890625)$	1055/64	$-499/64$
8	$[0.5, 0.53125)$	207/32	$-73/128$	**22**	$[0.890625, 0.90625)$	545/32	$-265/32$
9	$[0.53125, 0.578125)$	235/32	$-531/512$	**23**	$[0.90625, 0.921875)$	565/32	$-283/32$
10	$[0.578125, 0.625)$	523/64	$-3107/2048$	**24**	$[0.921875, 0.9375)$	605/32	$-1279/128$
11	$[0.625, 0.65625)$	289/32	$-525/256$	**25**	$[0.9375, 0.953125)$	315/16	$-687/64$
12	$[0.65625, 0.6875)$	317/32	$-21/8$	**26**	$[0.953125, 0.96875)$	321/16	$-11361/1024$
13	$[0.6875, 0.71875)$	171/16	$-811/256$	**27**	$[0.96875, 0.984375)$	325/16	$-2901/256$
14	$[0.71875, 0.75)$	1463/128	$-3787/1024$	**28**	$[0.984375, 1)$	355/16	$-3373/256$

The values of a and b of each sub-region is given in Tables 1 to 3.

According to Tables 1 to 3, we can observe that all conversions are based on shift-and-add operations, which are helpful for circuit implementations and can achieve more area efficiency than Chen's [4] look-up-table based methods.

4. **Simulation Results and VLSI Implementation.** The total errors of our proposed decimal antilogarithmic converters to actual decimal antilogarithmic curve could achieve 0.049, 0.0194 and 0.0098 for 8-region, 16-region and 28-region antilogarithmic conversions, respectively, where error analysis is simulated by MATLAB. VLSI implementations using TSMC 0.18 μm CMOS technology and error analysis of our proposed 8-, 16- and 28-region work are shown in Table 4. In Table 4, it can be shown that our proposed 8-, 16-, and 28-region decimal antilogarithmic conversion methods could attain the area-efficient and ROM-free under tolerant errors compared with Chen's [4]. It should be noted that in [4], the authors did not provide any information of VLSI design implementations. The delays of our proposed converters are short enough so that we can achieve high-speed conversions with area-efficient implementations.

TABLE 4. VLSI implementations and error analysis of our proposed 8-, 16- and 28-region method

Items	**Proposed**		
# of Regions	**8**	**16**	**28**
Area (μm^2)	55288	81970.80	160608.58
Minimum Delay (ns)	4.8	4.9	5.4
Maximum Positive Error	0.029	0.0114	0.005
Maximum Negative Error	-0.02	-0.008	-0.0048
Total Errors	0.049	0.0194	0.0098

5. **Conclusions.** In this paper, area-efficient and ROM-free decimal antilogarithmic converters are proposed. The proposed converters can be applied in many real practical applications to ease tremendous computation overhead. Future direction is to observe the entire performance incorporating our proposed antilogarithmic converters to the overall LNS-based systems.

Acknowledgment. This work is partially supported by National Quemoy University and National Pingtung University in Taiwan.

REFERENCES

[1] J. N. Mitchell, Computer multiplication and division using binary logarithms, *IRE Trans. Electronics Computers*, vol.11, no.11, pp.512-517, 1962.

[2] K. H. Abed and R. E. Siferd, VLSI implementation of a low-power antilogarithmic converter, *IEEE Trans. Computers*, vol.52, no.9, pp.1221-1228, 2003.

[3] H. Kim, B.-G. Nam, J.-H. Sohn, J.-H. Woo and H.-J. Yoo, A 231-MHz, 2.18mw 32-bit logarithmic arithmetic unit for fixed-point 3D graphics system, *IEEE Journal of Solid State Circuits*, vol.41, no.11, pp.2373-2381, 2006.

[4] D. Chen, Y. Zhang, L. Chen, D. Teng, K. Wahid and S.-B. Ko, A decimal-to-decimal antilogarithmic converter, *Proc. of the Canadian Conference on Electrical and Computer Engineering*, pp.1223-1226, 2008.

[5] T.-B. Juang, P. K. Meher and K.-S. Jan, High-performance logarithmic converters using novel two-region bit-level manipulation schemes, *Proc. of 2011 VLSI Symposium on Design, Automation, and Testing*, pp.390-393, 2011.

[6] D. Chen, Y. Zhang, L. Chen, D. Teng, K. Wahid and S.-B. Ko, A novel decimal-to-decimal logarithmic converter, *Proc. of the IEEE International Symposium on Circuits and Systems*, pp.688-691, 2008.

[7] C. Vudadha, S. Veeramachaneni and M. B. Srinivas, Non-linear partitioning for decimal logarithm approximation, *Asia Pacific Conference on Postgraduate Research*, pp.102-105, 2011.
[8] R. Tajallipour, Md. A. Islam and K. A. Wahid, Fast algorithm of a 64-bit decimal logarithmic converter, *Journal of Computers*, vol.5, no.12, pp.1847-1855, 2010.
[9] D. Chen, L. Han, Y. Choi and S.-B. Ko, Improved decimal floating-point logarithmic converter based on selection by rounding, *IEEE Trans. Computers*, vol.61, no.5, pp.607-621, 2012.
[10] B.-G. Nam, H.-J. Kim and H.-J. Yoo, Power and area-efficient unified computation of vector and elementary functions for handheld 3D graphics system, *IEEE Trans. Computers*, vol.57, no.4, pp.490-504, 2008.
[11] 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 and Systems II*, vol.56, no.12, pp.931-935, 2009.
[12] J.-A. Pineiro, Algorithm and architecture for logarithm, exponential, and powering computation, *IEEE Trans. Computers*, vol.53, no.9, pp.1085-1096, 2004.
[13] R. Gutierrez and J. Valls, Low cost hardware implementation of logarithm approximation, *IEEE Trans. VLSI Systems*, vol.19, no.12, pp.2326-2330, 2011.
[14] D. D. Caro, N. Petra and A. G. M. Strollo, Efficient logarithmic converters for digital signal processing applications, *IEEE Trans. Circuits and Systems II*, vol.58, no.10, pp.667-671, 2011.
[15] M. Chaudhary and P. Lee, Two-stage logarithmic converter with reduced memory requirements, *IET Computer and Digital Techniques*, vol.8, no.1, pp.23-29, 2014.
[16] M. Chaudhary and P. Lee, An improved two-step binary logarithmic converter for FPGAs, *IEEE Trans. Circuits and Systems II*, vol.62, no.5, pp.476-480, 2015.
[17] B. Liu, L. He and X. Yang, Base-N logarithm implementation on FPGA for the data with random decimal point position, *IEEE the 9th International Colloquium on Signal Processing and Its Applications*, pp.17-20, 2013.
[18] K. M. N. H. Khan, M. L. Ali and S. Islam, A new technique for high speed decimal logarithm computation of decimal floating-point number, *Proc. of the 14th International Conference on Computer and Information Technology*, pp.208-212, 2011.