applied sciences

MDPI

Article

Development of Circuits for Antilogarithmic Converters with Efficient Error–Area–Delay Product Using the Fractional-Bit Compensation Scheme for Digital Signal Processing Applications

Chao-Tsung Kuo

Department of Electrical Engineering, National Quemoy University, Kinmen 89250, Taiwan; ctkuo@nqu.edu.tw; Tel.: +886-82-313562; Fax: +886-82-313569

Abstract: Digital signal processing (DSP) has been widely adopted in sensor systems, communication systems, digital image processing, artificial intelligence, and Internet of Things applications. However, these applications require circuits for complex arithmetic computation. The logarithmic number system is a method to reduce the implementation area and transmission delay for arithmetic computation in DSP. In this study, we propose antilogarithmic converters with efficient error–area–delay products (eADPs) based on the fractional-bit compensation scheme. We propose three mathematical approximations—case 1, case 2, and case 3—to approximate the accurate antilogarithmic curve with different DSP requirements. The maximum percentage errors of conversion for case 1, case 2, and case 3 are 1.9089%, 1.7330%, and 1.2063%, respectively. Case 1, case 2, and case 3 can achieve eADP savings of 15.66%, 80.80%, and 84.61% compared with other methods reported in the literature. The proposed eADP-efficient antilogarithmic converters can achieve lower eADP and digitalized circuit implementation. The hardware implementation utilizes Verilog Hardware Description Language and the digital circuits are created via very-large-scale integration by the Taiwan Semiconductor Manufacturing Company with 0.18 μm CMOS technology. This proposed antilogarithmic converter can be efficiently applied in DSP.

Keywords: antilogarithmic converter; fractional-bit compensation; error–area–delay product; very-large-scale integration; digital circuits; digital signal processing

Citation: Kuo, C.-T. Development of Circuits for Antilogarithmic Converters with Efficient Error–Area–Delay Product Using the Fractional-Bit Compensation Scheme for Digital Signal Processing Applications. *Appl. Sci.* **2024**, *14*, 1487. https://doi.org/10.3390/app14041487

Academic Editors: Sheng-Joue Young, Shoou-Jinn Chang and Liang-Wen Ji

Received: 31 December 2023
Revised: 5 February 2024
Accepted: 9 February 2024
Published: 12 February 2024

1. Introduction

Digital signal processing (DSP) has been widely adopted in the Internet of Things (IoT), sensor systems, communication systems, digital image processing, and artificial intelligence (AI). Mobile handheld electronic devices, such as mobile phones, tablets, and notebooks, involve new technology and applications such as three-dimensional computer graphics and real-time systems in DSP. These require circuits for complex arithmetic computations such as multiplication, division, square root, squaring, and powering, which entail additional hardware costs and longer latency. To reduce the hardware costs and transmission delays, recent studies have developed novel methods to replace the complex arithmetic computations, such as the CORDIC algorithm [1], the table-based algorithm using rectangular multipliers [2], and the logarithmic number system (LNS) [3–22] to handle arithmetic computations. The CORDIC algorithm [1] uses an iterative method, and is not suitable for three-dimensional real-time DSP because of the limitation of operation speed. The table-based algorithm using rectangular multipliers [2] requires quite a large hardware memory for storage. The logarithmic number system (LNS) [3–22] reduces the implementation area and transmission delay for arithmetic computations in DSP. LNS-based arithmetic computation can simplify the complex operations by transforming multiplication to addition, division to subtraction, square root to right shifts, squaring to left shifts, powering

to continuous addition, and reciprocals to complementing. Table 1 shows the LNS-based operations.

Table 1. LNS-based operations for arithmetic computations.

Arithmetic Computation	Arithmetic Operation	LNS-Based Operation
Multiplication	$out = a \cdot b$	$A + B$
Division	$out = a/b$	$A - B$
Square root	$out = \sqrt{a}$	$A >> 1$
Squaring	$out = a^2$	$A << 1$
Powering	$out = a^b$	$\underbrace{A + A + ... + A}_{b}$
Reciprocal	$out = 1/a$	$-X$

The LNS consists of three main units: the logarithmic converter, simple arithmetic unit (SAU), and antilogarithmic converter, as shown in Figure 1. The logarithmic converter is used to convert two binary inputs (e.g., a and b in Table 1) into the logarithmic system format ($\log_2 a$ and $\log_2 b$). The SAU uses the shifter, complement, adder, or subtractor operations to perform a simple mathematical computation in the logarithmic system format. The antilogarithmic function is used to convert the final arithmetic computation result of the logarithmic system and simple arithmetic unit into a fixed-point binary output. The present study focuses on antilogarithmic converters with efficient error–area–delay products (eADPs) using the fractional-bit compensation schemes.

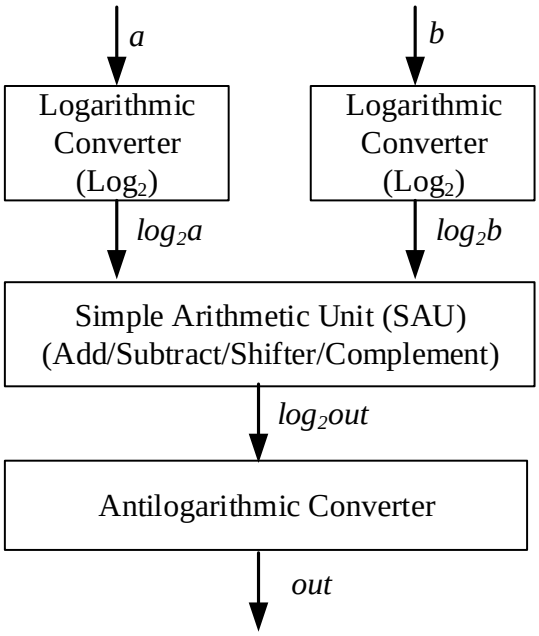

Figure 1. Logarithmic number system.

In recent years, many schemes have been proposed for antilogarithmic conversion systems, such as the straight-line method [3], look-up table method [7,15], shift-and-add method [5,11], bit correction scheme [13], and constant compensation scheme [16]. In 1962, Mitchell [3] first used an approximation method to approximate the antilogarithmic converter, which adopted $out = 1 + m$ to approximate an $out = 2^m$ curve. Although this method produces considerably large antilogarithmic conversion, it is simple and uses little hardware area. Nam et al. [7,15] used the look-up read-only memory (ROM) method to

implement the antilogarithmic converter. This method obtains highly accurate antilogarithmic conversion values from the SAU and logarithmic converter. However, the look-up table scheme demands additional hardware area. Abed and Siferd [5] and Loukrakpam and Choudhury [11] used the shift-and-add scheme to approximate an $out = 2^m$ curve. This method improved the performance in terms of the approximation error, hardware area, and delay, though it had further scope for improvement. Juang et al.'s [13] bit-correction scheme to approximate antilogarithmic conversion showed considerable scope for improvement in approximation error. Kuo and Juang [16] proposed the constant compensation scheme to approximate an $out = 2^m$ curve, which had scope of improvement in terms of hardware cost and latency. The above-mentioned shift-and-add method, bit correction scheme, and constant compensation scheme are ROM-free, and use mathematical function mapping to replace the look-up table method. However, the performance of these methods is not efficient enough for the design of antilogarithmic converters. The proposed eADP-efficient antilogarithmic converters based on the fractional-bit compensation scheme is expected to achieve high performance, considerably low approximation error, small hardware area, and short latency.

This paper is organized as follows. The methods reported in the literature are described in Section 2. The algorithm for the converter design incorporating the proposed eADP-efficient antilogarithmic converter is described in Section 3. Section 4 presents the results and comparisons of the various methods. Finally, the conclusions are presented in Section 5.

2. Antilogarithmic Conversion Methods

For antilogarithmic conversion, $out = 2^m$, m can be written as $m = i + f$, where i denotes the integer part and f is the fraction part. Thus, $2^m = 2^{(i+f)} = 2^i 2^f$. For simplified computation, consider the integer part i to be zero and the fraction part f to be between 0 and 1 ($0 \leq f < 1$). Taking $2^{0.4}$ as an example, let the input value be 0.4 = 0 + 0.4, where $i = 0$ and $f = 0.4$. The output will be $2^{0.4} = 2^0 \times 2^{0.4} = 2^{0.4} \approx 1.319$. In 1962, Mitchell [3] proposed out = $1 + f$ to approximate 2^f. This method is quite simple and has quite a low hardware cost. However, it produces a large approximation error. The maximum approximation error occurs at $f = 0.52877$, as shown by:

$$\text{difference}(f) = [(1 + f) - 2^f] \tag{1}$$

$$\frac{d}{df}\text{difference}(f) = \frac{d}{df}[(1 + f) - 2^f] = 1 - \ln 2 \times 2^f \tag{2}$$

$$Let\ 1 - \ln 2 \times 2^f = 0,\ 2^f = \frac{1}{\ln 2}, f = \frac{1}{\ln 2} \times \ln(\frac{1}{\ln 2}) \approx 0.52877 \tag{3}$$

The conversion error of Mitchell's [3] straight-line $out = 1 + f$ and the accurate antilogarithmic curve $out = 2^f$ are shown in Figure 2.

Over the past sixty years, many antilogarithmic conversion methods have been developed to improve the performance of approximation, latency, and hardware area for antilogarithmic converters. Among them, Abed and Siferd's shift-and-add scheme [5], Nam et al.'s ROM-based look-up table [7,15], Juang et al.'s bit correction scheme [13], Kuo and Juang's constant compensation scheme [16], and Loukrakpam and Choudhury's shift-and-add method [11] are some of the most efficient methods. In 2003, Abed and Siferd [5] proposed antilogarithmic converters with two-region, six-region, and seven-region shift-and-add linear approximation methods to reduce antilogarithmic conversion errors. Their two-region equation is:

$$Y' = 2^{m'} \approx \begin{cases} [m + \frac{3}{16} \times \overline{m_{7MSBits}} + \frac{13}{16} + \frac{1}{1024} + \frac{1}{2048}],\ 0 \leq m < 0.5; \\ [m + (\frac{3}{16}) \times m_{7MSBits} + \frac{13}{16}],\ 0.5 \leq m \leq 1; \end{cases} \tag{4}$$

where $\overline{m_{qMSBits}}$ is represented as $1 - m_{qMSBits} - (1/2^q)$ and $m_{qMSBits}$ is the first q most significant bits after the point of binary input.

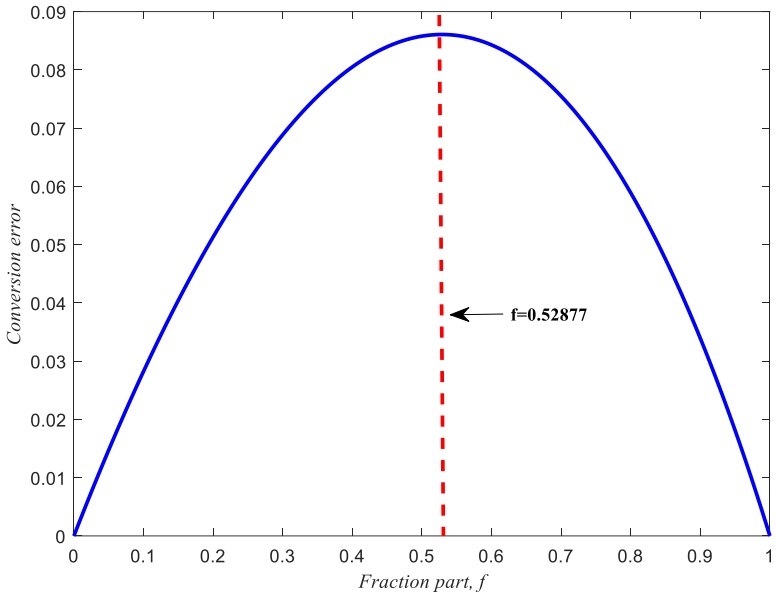

Figure 2. The conversion error of Mitchell's [3] straight-line $out = 1 + f$ and the accurate antilogarithmic curve $out = 2^f$.

The mathematical model of Loukrakpam and Choudhury's [11] two-region shift-and-add scheme is:

$$Y' = 2^f \approx \begin{cases} [f-(\frac{1}{8}+\frac{1}{32}+\frac{1}{64})\times f_{7MSBits}+\frac{2047}{2048}],\ 0 \le f < 0.5; \\ [f+(\frac{1}{8}+\frac{1}{32}+\frac{1}{64})\times f_{7MSBits}+\frac{1696}{2048}],\ 0.5 \le f \le 1; \end{cases} \tag{5}$$

The mathematical model of Juang et al.'s [13] bit correction scheme is:

$$Y' = 2^m \approx \begin{cases} [1+m-(\frac{m_{-2}}{16}+\frac{\overline{\overline{m_{-3}}\vee m_{-2}}}{32}+\frac{m_{-4}}{64}+\frac{m_{-3}}{128})],\ 0 \le m < 0.5; \\ [1+m-(\frac{\overline{m_{-2}}}{16}+\frac{\overline{m_{-3}\vee m_{-2}}}{32}+\frac{\overline{m_{-4}}}{64}+\frac{\overline{m_{-3}}}{128})],\ 0.5 \le m \le 1; \end{cases} \tag{6}$$

where $\vee$ is the logic OR gate and $\wedge$ is the logic AND gate. For the antilogarithmic converter's constant compensation scheme, Kuo and Juang's [16] 14-region constant compensation scheme is given by:

$$Y' = 2^m \approx \begin{cases} [1+m],\ 0 \le m < \frac{1}{32}; \\ [1+m-(\frac{1}{128})],\ \frac{1}{32} \le m < \frac{2}{32}; \\ [1+m-(\frac{2}{128})],\ \frac{2}{32} \le m < \frac{3}{32}; \\ [1+m-(\frac{3}{128})],\ \frac{3}{32} \le m < \frac{4}{32}; \\ [1+m-(\frac{4}{128})],\ \frac{4}{32} \le m < \frac{5}{32}; \\ [1+m-(\frac{5}{128})],\ \frac{5}{32} \le m < \frac{6}{32}; \\ [1+m-(\frac{6}{128})],\ \frac{6}{32} \le m < \frac{8}{32}; \\ [1+m-(\frac{8}{128})],\ \frac{8}{32} \le m < \frac{12}{32}; \\ [1+m-(\frac{10}{128})],\ \frac{12}{32} \le m < \frac{22}{32}; \\ [1+m-(\frac{8}{128})],\ \frac{22}{32} \le m < \frac{24}{32}; \\ [1+m-(\frac{7}{128})],\ \frac{24}{32} \le m < \frac{26}{32}; \\ [1+m-(\frac{5}{128})],\ \frac{26}{32} \le m < \frac{28}{32}; \\ [1+m-(\frac{3}{128})],\ \frac{28}{32} \le m < \frac{30}{32}; \\ [1+m],\ \frac{30}{32} \le m < 1; \end{cases} \tag{7}$$

The antilogarithmic conversion methods in the literature still have scope for improvement in terms of the approximation error, area cost, and delay time of antilogarithmic conversion. The eADP-efficient antilogarithmic converter using the fractional-bit compensation scheme

proposed in this study is expected to minimize the approximation error, area cost, and delay time further. Section 3 describes the proposed algorithm of the fractional-bit compensation.

3. Proposed Algorithm for Fractional-Bit Compensation

This section discusses the proposed eADP-efficient antilogarithmic converters using the fractional-bit compensation schemes. To design an efficient antilogarithmic converter, we first compare Mitchell's [3] straight-line $out = 1 + f$ with the accurate antilogarithmic curve $out = 2^f$ and analyze the percentage conversion error, as shown in Figures 3 and 4, respectively. The percentage conversion error is defined as 100% × (conversion error divided by 2^f):

$$\begin{aligned} &\text{Percentage Conversion Error } = \frac{\text{Conversion Error}}{2^f} \\ &= \frac{((1+f)-2^f)}{2^f} \times 100\%\,,\ 0 \le f < 1 \end{aligned} \tag{8}$$

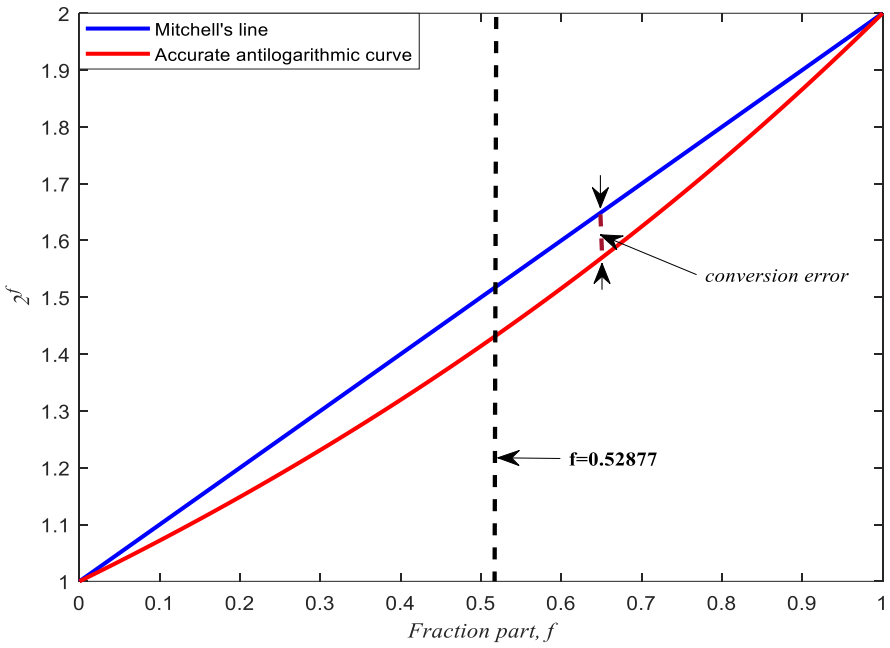

Figure 3. Comparison of Mitchell's [3] straight-line $out = 1 + f$ and the accurate antilogarithmic curve $out = 2^f$.

The maximum percentage error is defined as 100% × (sum of absolute value of maximum positive percentage error and minimum negative percentage error):

$$\begin{aligned} &\text{Maximum Percentage Error} \\ &= \left(\frac{|\text{Maximum positive percentage error}|+|\text{Minimum negative percentage error}|}{2^f}\right) \times 100\%\,,\ 0 \le f < 1 \end{aligned} \tag{9}$$

For the fractional-bit compensation method, we found that the first region and the last region could not be compensated by Figure 3. The compensated region is divided by the 2^n region, where n denotes the first most significant bits (MSB). For example, $n = 4$ is divided by 16 compensated regions. That is, f_{-1}, f_{-2}, f_{-3}, and f_{-4} bits in the fraction part are used to partition 16 regions.

Table 2 shows the maximum conversion error and percentage error for the uncompensated first region and last region, where n is set to 3 to 8. The partition region is considered to be a uniform partition. Table 2 presents the local errors and local percentage errors for the first and last regions. The larger values of the first region and the last region are considered to be the absolute maximum errors and absolute maximum percentage errors. Table 2 presents the important indexes for deciding the partition and compensation values. Taking $n = 4$ for example, the partition number is 16 regions. The upper bound of the maximum percentage conversion error is 1.7327%. That is, a designed maximum percent-

age conversion error below 1.7327% cannot be achieved for 16 uniform regions, even if a full-precision fractional-bit compensation value is used.

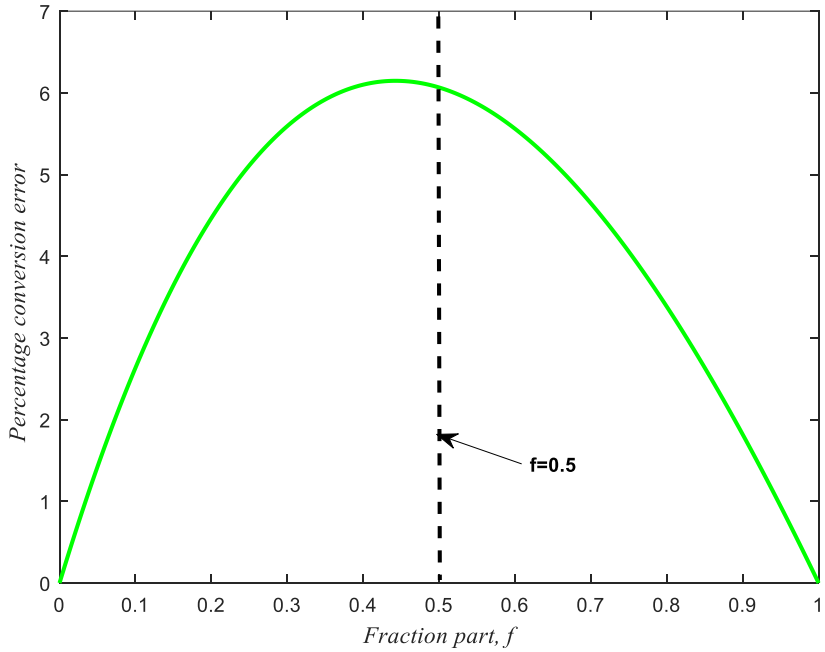

Figure 4. Percentage conversion error of Mitchell's [3] straight-line $out = 1 + f$ and the accurate antilogarithmic curve $out = 2^f$.

Table 2. Maximum conversion errors and percentage errors for the uncompensated first region and last region.

f_{-1}–f_{-n}	Uniform Partition Numbers	Key Uncompensated Region	Local Error	Local Percentage Error	Absolute Maximum Error	Absolute Maximum Percentage Error
$n = 3$	8	First region	0.0342	3.1427%	0.0410	3.1427%
		Last region	0.0410	2.2351%		
$n = 4$	16	First region	0.0181	1.7327%	0.0221	1.7327%
		Last region	0.0221	1.1551%		
$n = 5$	32	First region	0.0093	0.9083%	0.0115	0.9083%
		Last region	0.0115	0.5883%		
$n = 6$	64	First region	0.0045	0.4502%	0.0057	0.4502%
		Last region	0.0057	0.2873%		
$n = 7$	128	First region	0.0021	0.2126%	0.0027	0.2126%
		Last region	0.0027	0.1347%		
$n = 8$	256	First region	0.0009	0.0916%	0.0012	0.0578%
		Last region	0.0012	0.0578%		

The process of optimizing eADP is shown in the flowchart in Figure 5. In the algorithm shown in Figure 3, we first consider the upper bound of each partition region for different fractional bits n. After determining the desired maximum percentage error, the compensation precision for 2^{-k} (k = 3, 4, 5, . . ., 26) is selected and adjusted. In this work, we adopt the Q6.26 format, which contains 6 bits of the integer part and 26 fractional bits. It should be noted that the smaller the partition number and the larger the compensation precision, the smaller the hardware area and the shorter the delay time, respectively. To obtain the truth table for the compensation bits, MATLAB software R2017b is used to simulate the percentage conversion error. Finally, the Karnaugh map is used to simplify the circuit of the truth table. Subsequently, we obtain two regions of the coarse equation for eADP-efficient antilogarithmic converters using fractional-bit compensation schemes. The hardware area and delay time for circuit implementation via very-large-scale integration (VLSI) are obtained using the coarse equation. If the product of error, hardware area, and delay time is larger than the desired target, then the process is repeated until the desired value is achieved. After the fine-tuning process, the eADP-efficient antilogarithmic converter using fractional-bit compensation schemes is designed.

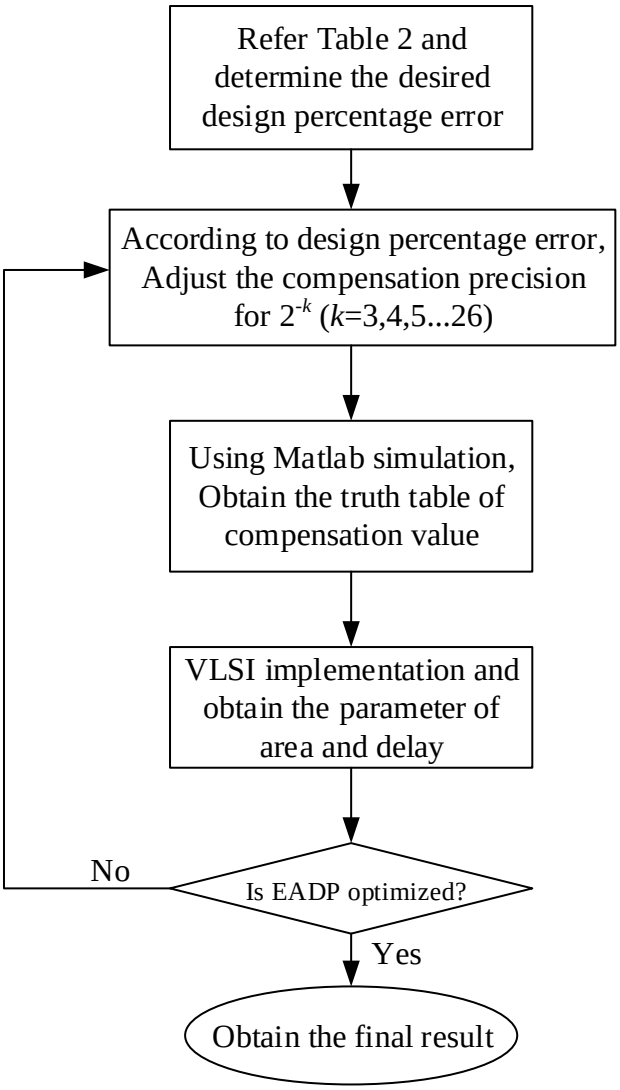

Figure 5. Flowchart for optimization of eADP in the range $0 \leq f < 1.0$.

According to the above algorithm, we propose three different specifications for eADP-efficient antilogarithmic converters using fractional-bit compensation schemes. We denote the three equations as case 1, case 2, and case 3. For n = 3 (m_{-1}–m_{-3}), the maximum

percentage error is larger (3.1427%), so it is not used in the proposed algorithms. First, we use $n = 4$ (m_{-1}–m_{-4}) to give sixteen regions and the compensation values of 2^{-4}, 2^{-5}, 2^{-6}, and 2^{-7}, considering the hardware cost and latency. Note that 2^{-1}, 2^{-2}, and 2^{-3} are not used, owing to their larger step compensation values. Table 3 shows the truth table and compensation bits for case 1. The corresponding partition regions and compensation values are shown in Table 4. After simplification of the Karnaugh map from Table 3, the proposed equation of the eADP-efficient antilogarithmic converter is given by Equation (10). The maximum percentage error of case 1 is 1.9089%.

$$out = 2^f \approx \begin{cases} [1+f-(\frac{(f_{-2}\wedge f_{-3})\vee(f_{-2}\wedge(f_{-3}\oplus f_{-4}))}{16} \\ +\frac{(\overline{f_{-2}}\wedge f_{-3})\vee(f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}})}{32}+\frac{(\overline{f_{-2}}\wedge f_{-4})\vee(f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}})}{64})], \ 0 \le f < 0.5; \\ [1+f-(\frac{\overline{f_{-2}}}{16}+\frac{f_{-2}\wedge\overline{f_{-3}}}{32}+\frac{f_{-2}\wedge f_{-3}\wedge\overline{f_{-4}}}{64}+\frac{f_{-2}\wedge\overline{f_{-3}}}{128})], \ 0.5 \le f < 1; \end{cases} \quad (10)$$

Table 3. Truth table and compensation bits of case 1.

f_{-1}–f_{-4}	Compensation Bit				f_{-1}–f_{-4}	Compensation Bit			
	2^{-4}	2^{-5}	2^{-6}	2^{-7}		2^{-4}	2^{-5}	2^{-6}	2^{-7}
0000	0	0	0	0	1000	1	0	0	0
0001	0	0	1	0	1001	1	0	0	0
0010	0	1	0	0	1010	1	0	0	0
0011	0	1	1	0	1011	1	0	0	0
0100	0	1	1	0	1100	0	1	0	1
0101	1	0	0	0	1101	0	1	0	1
0110	1	0	0	0	1110	0	0	1	0
0111	1	0	0	0	1111	0	0	0	0

Table 4. Partition regions and compensation values of case 1.

Partition Items	Partition Region	Compensation Value	Partition Items	Partition Region	Compensation Value
1	[0, 1/16)	0	**5**	[5/16, 12/16)	−8/128
2	[1/16, 2/16)	−2/128	**6**	[12/16, 14/16)	−5/128
3	[2/16, 3/16)	−4/128	**7**	[14/16, 15/16)	−2/128
4	[3/16, 5/16)	−6/128	**8**	[15/16, 1)	0

Table 5 shows the truth table and compensation bits for case 2. The corresponding partition regions and compensation values are shown in Table 6. After simplification of the Karnaugh map from Table 5, the proposed equation of the eADP-efficient antilogarithmic converter is given by Equation (11). The maximum percentage error of case 2 is 1.7330%.

$$out = 2^f \approx \begin{cases} [1+f-(\frac{(f_{-2}\wedge f_{-3})\vee(f_{-2}\wedge(f_{-3}\oplus f_{-4}))}{16} \\ +\frac{(\overline{f_{-2}}\wedge f_{-3})\vee(f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}})}{32}+\frac{(\overline{f_{-2}}\wedge f_{-4})\vee(f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}})}{64} \\ +\frac{f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}}}{128})], \ 0 \le f < 0.5; \\ [1+f-(\frac{\overline{f_{-2}}}{16}+\frac{f_{-2}\wedge\overline{f_{-3}}}{32}+\frac{f_{-2}\wedge f_{-3}\wedge\overline{f_{-4}}}{64}+\frac{f_{-2}\wedge\overline{f_{-3}}}{128})], \ 0.5 \le f < 1; \end{cases} \quad (11)$$

Table 5. Truth table and compensation bits of case 2.

f_{-1}–f_{-4}	Compensation Bit				f_{-1}–f_{-4}	Compensation Bit			
	2^{-4}	2^{-5}	2^{-6}	2^{-7}		2^{-4}	2^{-5}	2^{-6}	2^{-7}
0000	0	0	0	0	1000	1	0	0	0
0001	0	0	1	0	1001	1	0	0	0
0010	0	1	0	0	1010	1	0	0	0
0011	0	1	1	0	1011	1	0	0	0
0100	0	1	1	1	1100	0	1	0	1
0101	1	0	0	0	1101	0	1	0	1
0110	1	0	0	0	1110	0	0	1	0
0111	1	0	0	0	1111	0	0	0	0

Table 6. Partition regions and compensation values of case 2.

Partition Items	Partition Region	Compensation Value	Partition Items	Partition Region	Compensation Value
1	[0, 1/16)	0	**6**	[5/16, 12/16)	−8/128
2	[1/16, 2/16)	−2/128	**7**	[12/16, 14/16)	−5/128
3	[2/16, 3/16)	−4/128	**8**	[14/16, 15/16)	−2/128
4	[3/16, 4/16)	−6/128	**9**	[15/16, 1)	0
5	[4/16, 5/16)	−7/128			

Table 7 shows the truth table and compensation bits of case 3. The corresponding partition regions and compensation values are shown in Table 8. After simplification of the Karnaugh map from Table 7, the proposed equation of the eADP-efficient antilogarithmic converter is given by Equation (12). The maximum percentage error of case 3 is 1.2063%. In Equations (10)–(12), $\vee$ represents the logic OR gate, $\wedge$ represents the logic AND gate, $\overline{\ }$ represents the logic NOT gate, and $\oplus$ represents the logic Exclusive OR gate.

$$out = 2^{f} \approx \begin{cases} [1+f-(\frac{(f_{-2}\wedge f_{-3})\vee(f_{-2}\wedge f_{-4})\vee(f_{-2}\wedge f_{-5})}{16} \\ +\frac{(\overline{f_{-2}}\wedge f_{-3})\vee(f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}}\wedge\overline{f_{-5}})}{32}+\frac{(\overline{f_{-2}}\wedge f_{-4})\vee(f_{-2}\wedge\overline{f_{-3}}\wedge\overline{f_{-4}}\wedge\overline{f_{-5}})}{64} \\ +\frac{(\overline{f_{-2}}\wedge f_{-5})\vee(f_{-2}\wedge f_{-3})\vee(f_{-2}\wedge f_{-4})\vee(f_{-2}\wedge\overline{f_{-5}})}{128})], \ 0 \le f < 0.5; \\ [1+f-(\frac{(\overline{f_{-2}}\wedge\overline{f_{-3}})\vee(\overline{f_{-2}}\wedge\overline{f_{-4}})\vee(\overline{f_{-2}}\wedge\overline{f_{-5}})}{16}+\frac{(f_{-2}\wedge\overline{f_{-3}})\vee(\overline{f_{-2}}\wedge f_{-3}\wedge f_{-4}\wedge f_{-5})}{32} \\ +\frac{(f_{-2}\wedge\overline{f_{-4}})\vee(\overline{f_{-2}}\wedge f_{-3}\wedge f_{-4}\wedge f_{-5})}{64}+\frac{\overline{f_{-2}}\vee\overline{f_{-3}}\vee\overline{f_{-4}}}{128})], \ 0.5 \le f < 1; \end{cases} \quad (12)$$

Under the different error tolerances of specific applications of DSP, digital image processing, or AI, the three cases present three different approaches to reducing circuit complexity. However, the more fractional bits there are, the less the maximum percentage error and approximation error will be. The circuit complexity of case 1 (Equation (10)) is simpler than that of case 2 (Equation (11)) and case 3 (Equation (12)). Therefore, case 1 will have larger approximation error. Case 3 uses more fractional bits to compensate for the approximate error, so it has larger hardware cost and longer delay time. The three proposed equations that can be employed under the different error tolerances of eADP-efficient antilogarithmic converters are easy to implement in a digital VLSI circuit with ROM-free requirements. In the next section, we discuss the simulation results using MATLAB software, the hardware implementation, and VLSI synthesis using Verilog Hardware Description Language (HDL) and we compare the system performance with previous schemes.

Table 7. Truth table and compensation bits of case 3.

f_{-1}–f_{-5}	Compensation Bit				f_{-1}–f_{-5}	Compensation Bit			
	2^{-4}	2^{-5}	2^{-6}	2^{-7}		2^{-4}	2^{-5}	2^{-6}	2^{-7}
00000	0	0	0	0	10000	1	0	0	1
00001	0	0	0	1	10001	1	0	0	1
00010	0	0	1	0	10010	1	0	0	1
00011	0	0	1	1	10011	1	0	0	1
00100	0	1	0	0	10100	1	0	0	1
00101	0	1	0	1	10101	1	0	0	1
00110	0	1	1	0	10110	1	0	0	1
00111	0	1	1	1	10111	0	1	1	1
01000	0	1	1	1	11000	0	1	1	1
01001	1	0	0	0	11001	0	1	1	1
01010	1	0	0	1	11010	0	1	0	1
01011	1	0	0	1	11011	0	1	0	1
01100	1	0	0	1	11100	0	0	1	1
01101	1	0	0	1	11101	0	0	1	1
01110	1	0	0	1	11110	0	0	0	0
01111	1	0	0	1	11111	0	0	0	0

Table 8. Partition regions and compensation values of case 3.

Partition Items	Partition Region	Compensation Value	Partition Items	Partition Region	Compensation Value
1	[0, 1/32)	0	**8**	[7/32, 9/32)	−7/128
2	[1/32, 2/32)	−1/128	**9**	[9/32, 10/32)	−8/128
3	[2/32, 3/32)	−2/128	**10**	[10/32, 23/32)	−9/128
4	[3/32, 4/32)	−3/128	**11**	[23/32, 26/32)	−7/128
5	[4/32, 5/32)	−4/128	**12**	[26/32, 28/32)	−5/128
6	[5/32, 6/32)	−5/128	**13**	[28/32, 30/32)	−3/128
7	[6/32, 7/32)	−6/128	**14**	[30/32, 1)	0

4. Experimental Results and Hardware Implementation

We proposed three equations for eADP-efficient antilogarithmic converters based on fractional-bit compensation schemes under different specifications and requirements for DSP applications. In general, the larger the approximation and maximum percentage error, the less the hardware area cost and delay time. Herein, we first sort the similar maximum percentage errors of antilogarithmic converters reported in the literature as one group and then compare their eADP efficiencies. Next, we compare the performances of the three equations as three groups with those of the methods in the literature. The percentage conversion errors of case 1, case 2, and case 3 are obtained using MATLAB and compared with those of the previous schemes. The maximum percentage conversion errors of case 1, case 2, and case 3 are 1.9089%, 1.7330%, and 1.2063%, respectively. Figure 6 compares case 1 with Juang et al.'s two-region bit correction [13]. Figure 7 compares case 2 with Abed and Siferd's two-region shift-and-add method [5] and Kuo and Juang's 11-region constant compensation scheme [16]. Figure 8 compares case 3 with Abed and Siferd's

six-region shift-and-add method [5], Loukrakpam and Choudhury's two-region shift-and-add method [11], and Kuo's 14-region constant compensation scheme [16]. Figures 6–8 clearly show that the three cases yield considerably lower percentage conversion errors than the previous methods. Figure 9 shows the approximated curve of case 3 compared to the accurate antilogarithmic curve; it is noted that the two-region equation of case 3 is extremely close to the accurate antilogarithmic curve.

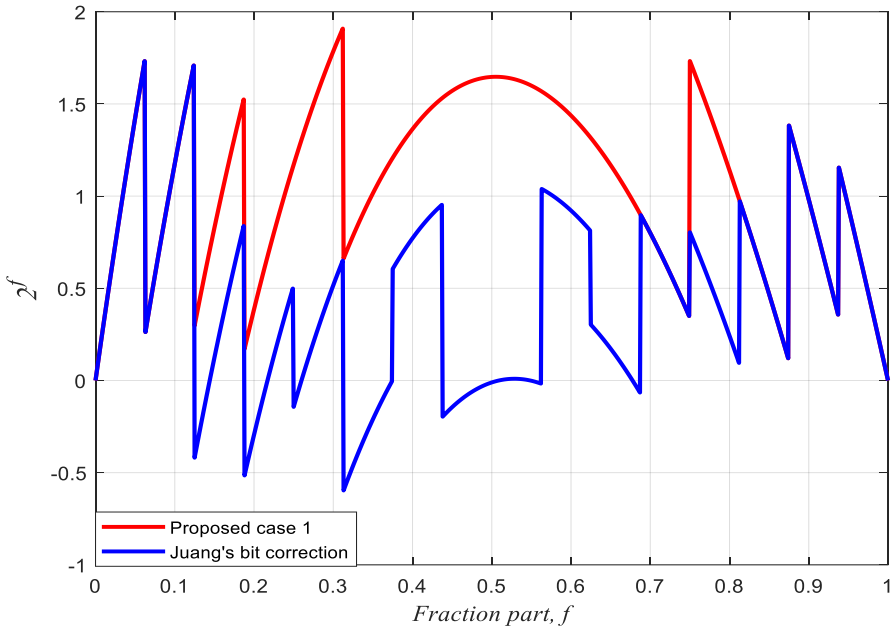

Figure 6. Comparison of the antilogarithmic percentage conversion errors of case 1 and Juang et al.'s [13] 2-region bit correction scheme.

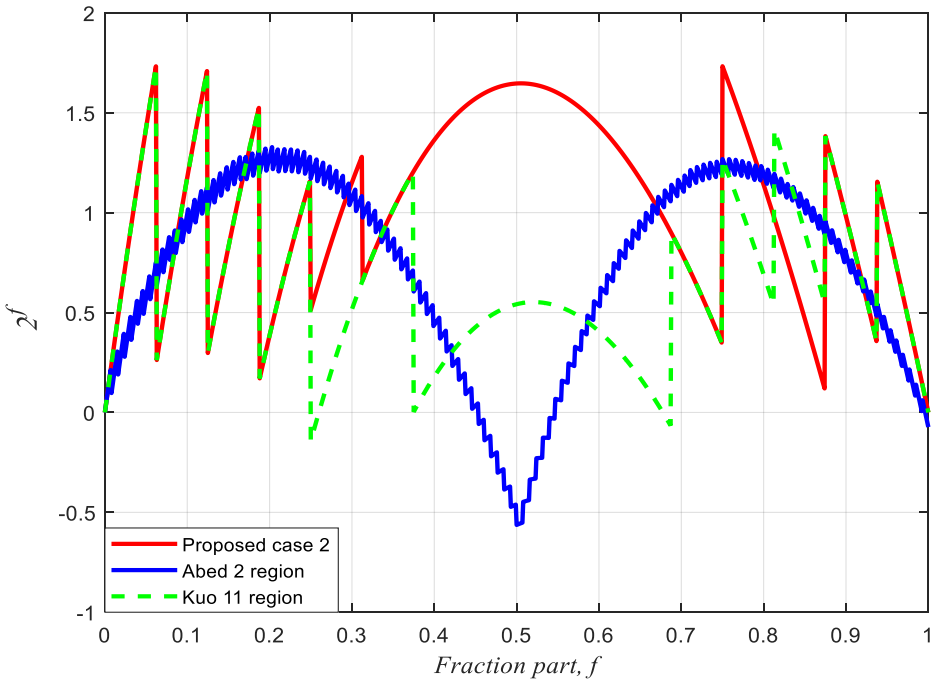

Figure 7. Comparison of the antilogarithmic percentage conversion errors of case 2 and Abed and Siferd's [5] 2-region shift-and-add and Kuo and Juang's [16] 11-region bit constant compensation schemes.

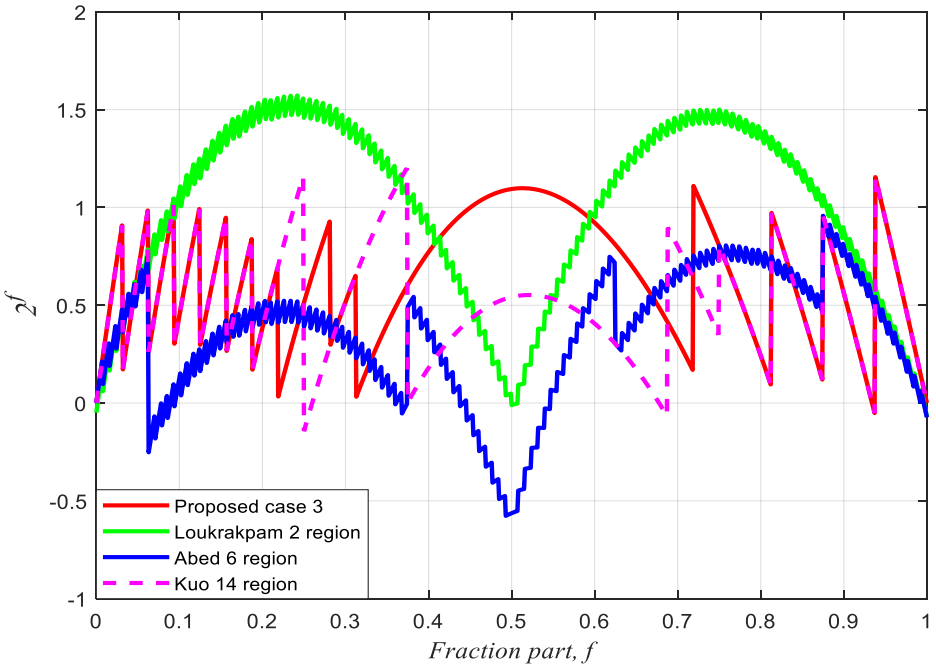

Figure 8. Comparison of the antilogarithmic percentage conversion errors of case 3 and Abed and Siferd's [5] 6-region shift-and-add, Loukrakpam and Choudhury's [11] 2-region shift-and-add, and Kuo and Juang's [16] 14-region bit constant compensation schemes.

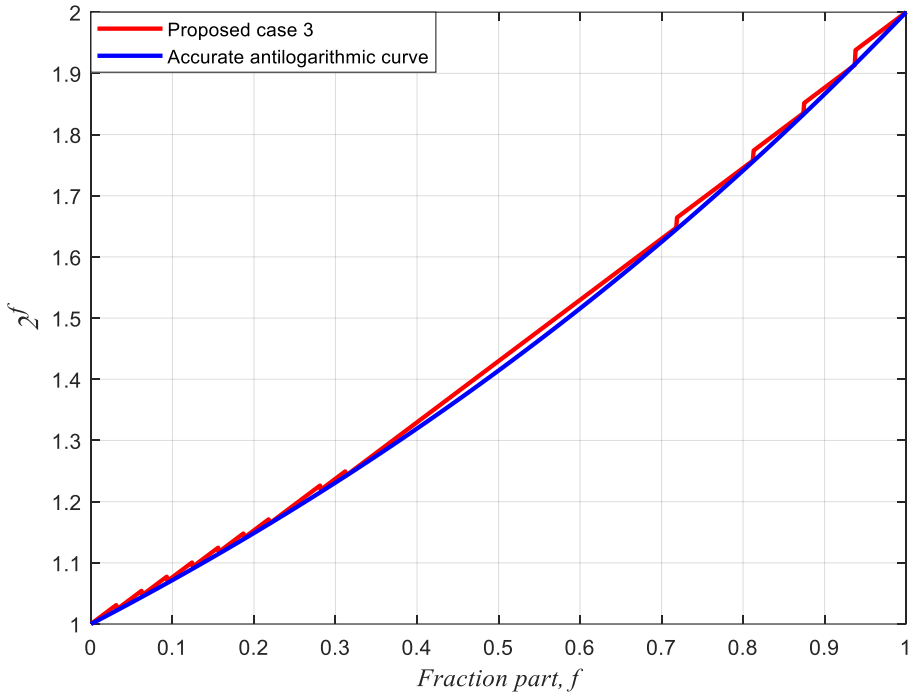

Figure 9. Approximate curve of proposed case 3 compared to the accurate antilogarithmic curve.

Comparisons of the results for VLSI hardware realization and maximum percentage conversion errors for cases 1, 2, and 3 are presented in Tables 9–11, respectively. The same hardware and software operation environments are used for all cases and reported methods. For hardware implementation, we used Verilog HDL and had the digital circuits created via VLSI by the Taiwan Semiconductor Manufacturing Company with 0.18 µm CMOS technology. The percentage conversion error is simulated using MATLAB software. In Tables 9–11, ADP is defined as the product of hardware area and delay time, while eADP

is defined as the product of the approximation's maximum percentage conversion error, hardware area, and delay time. The approximation's maximum percentage conversion error is defined as the sum of the absolute value of the positive maximum percentage error and the absolute value of the negative minimum percentage error. Tables 9–11 indicate that case 1, case 2, and case 3 can achieve eADP savings of 15.66%, 80.80%, and 84.61% compared with the other methods. The antilogarithmic percentage conversion error, delay time, hardware area, and eADP savings of the proposed cases are superior to those of the other reported methods.

Table 9. Comparison of percentage conversion errors and results for proposed case 1 and Juang et al.'s [13] two-region bit correction schemes.

Compared Items	Juang et al. [13]	Proposed Case 1
Compensation scheme	Bit	**Fractional bits**
Segment number	2	**2**
Significant fractional bit	none	**none**
Max. positive percentage error	1.72%	**1.9089%**
Min. negative percentage error	−0.6%	**0**
Total percentage conversion error	2.32%	**1.9089%**
Area (μm^2)	1729.62	**1772.97**
Delay (ns)	0.7	**0.7**
ADP	1210.734	**1241.079**
ADP saving	0	**−2.51%**
eADP	28.0890	**23.6910**
eADP saving	0	**15.66%**

Table 10. Comparison of percentage conversion errors and results for proposed case 2 and Abed and Siferd's [5] two-region and Kuo and Juang's [16] 11-region schemes.

Compared Items	Abed & Siferd [5,16]	Kuo & Juang [16]	Proposed Case 2
Compensation scheme	Shift-and-Add	Constant	**Fractional bits**
Segment number	2	11	**2**
Significant fractional bit	7	none	**none**
Max. positive percentage error	1.3310%	1.7327%	**1.7330%**
Min. negative percentage error	−0.5631%	−0.0992%	**0**
Total percentage conversion error	1.8941%	1.8319%	**1.7330%**
Area (μm^2)	3562.57	2807.48	**1869.44**
Delay (ns)	2	1.4	**0.8**
ADP	7125.14	3930.47	**1495.55**
ADP saving	0	44.84%	**79.01%**
eADP	134.9573	71.2594	**25.9179**
eADP saving	0	47.20%	**80.80%**

It should be noted that De Morgan's law of logic circuits can be used to reduce the gate-count number for hardware realization in Equations (10)–(12). Taking $f_{-2} \wedge \overline{f_{-3}} \wedge \overline{f_{-4}} \wedge \overline{f_{-5}}$ in Equation (12) as an example, $f_{-2} \wedge (\overline{f_{-3} \vee f_{-4} \vee f_{-5}})$ can be used to save two gate-counts. Hence, the hardware cost of the proposed eADP-efficient antilogarithmic converters can be significantly reduced by the proposed algorithm of the fractional-bit compensation, Karnaugh map and De Morgan's law of logic circuits.

Table 11. Comparison of percentage conversion errors and results for proposed case 3 and Abed and Siferd's [5] six-region, Kuo and Juang's [16] 14-region, and Loukrakpam and Choudhury's [11] two-region schemes.

Compared Items	Abed & Siferd [5,16]	Kuo & Juang [16]	Loukrakpam [11]	Proposed Case 3
Compensation scheme	Shift-and-Add	Constant	Shift-and-Add	**Fractional bits**
Segment number	6	14	2	**2**
Significant fractional bit	7	none	7	**none**
Max. positive percentage error	0.9572%	1.2%	1.5054%	**1.1551%**
Min. negative percentage error	−0.5786%	−0.1436%	−0.0488%	**−0.0512%**
Total percentage conversion error	1.5358%	−0.1436%	1.5542%	**1.2063%**
Area (μm^2)	6439.91	3319.75	5425.36	**2122.24**
Delay (ns)	6439.91	1.5	1.8	**1.1**
ADP	11,913.83	4979.63	9765.65	**2334.46**
ADP saving	0	58.20%	18.31%	**80.41%**
eADP	182.9726	66.9063	151.78	**28.1606**
eADP saving	0	63.43%	17.05%	**84.61%**

The circuit block diagram for hardware implementation for the three cases is shown in Figure 10, where HS is a half subtractor, FS is a full subtractor, f_{-n} (where n = 1 to 26) is the uncompensated fractional bit, and $f_{-n}{}^{*}$ (where n = 1 to 26) is the compensated bit. Two regions ($0 \leq f < 0.5$ and $0.5 \leq f < 1$) of the equations are selected by f_{-1}. The block of logic gates for the compensation circuit (e.g., OR, AND, XOR, and NOT gates) incorporates the combinational logic in the numerators of the equations. In Figure 10, $f_{1/16}$, $f_{1/32}$, $f_{1/64}$, and $f_{1/128}$ are the compensated output values of 1/16, 1/32, 1/64, and 1/128 in Equations (10)–(12), respectively. For the DSP applications of cases 1, 2, and 3, the more fractional bits there are, the lower the maximum percentage error and approximation error will be. However, more fractional bits will lead to larger hardware area and longer delay time. Therefore, the maximum percentage conversion errors (1.9089%, 1.7330%, and 1.2063%) of cases 1, 2, and 3 can be individually adopted for the error tolerances of specific DSP, digital image processing, or AI applications. We should note that, in this study, the circuit block of Figure 10 is implemented using Verilog HDL code and the digital circuits are integrated by the Taiwan Semiconductor Manufacturing Company with 0.18 μm CMOS technology. MATLAB software is used to display simulation results from Equations (10)–(12) and the other reported methods.

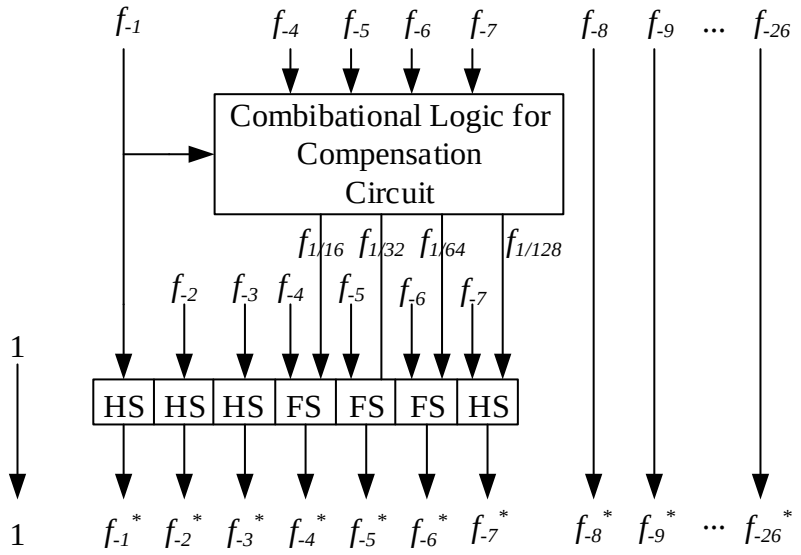

Figure 10. Circuit block diagram for hardware implementation.

5. Conclusions

We proposed three mathematical equations for antilogarithmic converters with efficient error–area–delay product (eADP) using the fractional-bit compensation scheme. The proposed converters achieved high performance in terms of lower approximation errors, smaller hardware implementation areas, and shorter latency. We compared the proposed algorithms with the previously reported shift-and-add, bit correction, and constant compensation schemes, and found the proposed converters achieved faster result, lower hardware implementation area, and efficient ADP compared to existing methods. The eADP savings of case 1, case 2, and case 3 were 15.66%, 80.80%, and 84.61%, respectively. The digital circuit for the proposed antilogarithmic converters is simple and easy to implement with very-large-scale integration. The proposed eADP-efficient antilogarithmic converters using the fractional-bit compensation scheme are superior to other methods and can be effectively applied to digital signal processing with different specifications and requirements.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: This paper was partly supported by the Ministry of Science and Technology in Taiwan, under grant number MOST 108-2221-E-507-010.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Walther, J.S. A Unified Algorithm for Elementary Functions. In Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, USA, 18–20 May 1971; pp. 379–385.
2. Wong, W.F.; Goto, E. Fast Hardware-Based Algorithms for Elementary Function Computations Using Rectangular Multipliers. *IEEE Trans. Comput.* **1994**, *43*, 278–294. [CrossRef]
3. Mitchell, J.N. Computer multiplication and division using binary logarithms. *IRE Trans. Electron. Comput.* **1962**, *EC-11*, 512–517. [CrossRef]
4. Stine, J.E.; Schulte, M.J. The symmetric table addition method for accurate function approximation. *J. VLSI Sig. Proc.* **1999**, *21*, 167–177. [CrossRef]
5. Abed, K.H.; Siferd, R.E. VLSI implementation of a low-power antilogarithmic converter. *IEEE Trans. Comput.* **2003**, *52*, 1221–1228. [CrossRef]
6. Juang, T.B.; Chen, S.H.; Cheng, H.J. A lower-error and ROM-free logarithmic converter for digital signal processing applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2009**, *56*, 931–935.
7. Nam, B.G.; Kim, H.J.; Yoo, H.J. Power and area-efficient unified computation of vector and elementary functions for handheld 3D graphics system. *IEEE Trans. Comput.* **2008**, *57*, 490–504. [CrossRef]
8. Paul, S.; Jayakumar, N.; Khatri, S. A fast hardware approach for approximate, efficient logarithm and antilogarithm computations. *IEEE Trans. VLSI Syst.* **2009**, *17*, 269–277. [CrossRef]
9. Liu, C.W.; Ou, S.H.; Chang, K.C.; Lin, T.C.; Chen, S.K. A low-error, cost-efficient design procedure for evaluating logarithms to be used in a logarithmic arithmetic processor. *IEEE Trans. Comput.* **2016**, *65*, 1158–1164. [CrossRef]
10. Kuo, C.T. Design and realization of high performance logarithmic converters using non-uniform multi-regions constant adder correction schemes. *Microsyst. Technol.* **2018**, *24*, 4237–4245. [CrossRef]
11. Loukrakpam, M.; Choudhury, M. Error-Aware Design Procedure to Implement Hardware-Efficient Antilogarithmic Converters. *Circuit Syst. Signal Process.* **2019**, *38*, 4266–4279. [CrossRef]
12. Kuo, C.T. Design and Circuit Implementation of Area-Delay-Product-Efficient Logarithmic Converters Using Mantissa-Bit Compensation Scheme. *Circuit Syst. Signal Process.* **2022**, *41*, 4266–4279. [CrossRef]
13. Juang, T.B.; Kuo, H.L.; Jan, K.S. Lower-error and area-efficient antilogarithmic converters with bit correction schemes. *J. Chin. Inst. Eng.* **2016**, *39*, 57–63. [CrossRef]
14. Ha, H.; Lee, S. Accurate hardware-efficient logarithm circuit. *IEEE Trans. Circuits Syst.—II Express Briefs* **2017**, *64*, 967–971. [CrossRef]
15. Kim, H.; Nam, B.G.; Sohn, J.H.; Woo, J.H.; Yoo, H.J. A 231-MHz, 2.18Mw 32-bit logarithmic arithmetic unit for fixed-point 3-D graphics system. *IEEE J Solid State Circuits.* **2006**, *41*, 2373–2381. [CrossRef]
16. Kuo, C.T.; Juang, T.B. Area-efficient and highly accurate antilogarithmic converters with multiple regions of constant compensation schemes. *Microsyst. Technol.* **2018**, *24*, 219–225. [CrossRef]
17. Chaudhary, M.; Lee, P. Two-stage logarithmic converter with reduced memory requirements. *IET Comput. Digit. Tech.* **2014**, *8*, 23–29. [CrossRef]

18. Chaudhary, M.; Lee, P. An improved two-step binary logarithmic converter for FPGAs. *IEEE Trans. Circuits Syst. II Express Briefs* **2015**, *62*, 476–480. [CrossRef]
19. Pineiro, J.A.; Ercegovac, M.D.; Bruguera, J.D. Algorithm and architecture for logarithm, exponential, and powering computation. *IEEE Trans. Comput.* **2004**, *53*, 1085–1096. [CrossRef]
20. Gutierrez, R.; Valls, J. Low cost hardware implementation of logarithm approximation. *IEEE Trans. Very Large Scale Integr. Syst.* **2011**, *19*, 2326–2330. [CrossRef]
21. Juang, T.B.; Meher, P.K.; Jan, K.S. High-performance logarithmic converters using novel two-region bit-level manipulation schemes. In Proceedings of the 2011 International Symposium on VLSI Design, Automation and Test, Hsinchu, Taiwan, 25–28 April 2011.
22. Caro, D.D.; Petra, N.; Strollo, A.G.M. Efficient logarithmic converters for digital signal processing applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2011**, *58*, 667–671.