A New Floating Point Implementation of Base 2 Logarithm

Ahmed M. Mansour,   Ali M. El-Sawy,   Ahmed T Sayed

Abstract — Logarithms reduce products to sums and powers to products; they play an important role in signal processing, communication and information theory. They are primarily used for hardware calculations, handling multiplications, divisions, powers, and roots effectively. There are three commonly used bases for logarithms; the logarithm with base-10 is called the common logarithm, the natural logarithm with base-e and the binary logarithm with base-2. This paper demonstrates different methods of calculation for log_2 showing the complexity of each and finds out the most accurate and efficient besides giving insights to their hardware design. We present a new method called Floor Shift for fast calculation of log_2, and then we combine this algorithm with Taylor series to improve the accuracy of the output, we illustrate that by using two examples. We finally compare the algorithms and conclude with our remarks.

Keywords — Logarithms, log_2, floor, iterative, CORDIC, Taylor series.

I. Introduction

LOGARITHM tables have been used extensively in the early 17th century to perform calculations used for many applications in mathematics and science, until replaced in the latter half of the 20th century by electronic calculators and computers. Logarithmic scales reduce wide-ranging quantities to smaller scopes; the binary logarithm is often used in digital communications and information theory because it is closely connected to the binary numeral system, information entropy involves the binary logarithm, this is needed to compare the efficiency of different probable implementation alternatives. If a number n, greater than 1, is divided by 2 repeatedly, the number of iterations needed to get a value at most 1 is the integral part of $log_2(n)$. This idea is used in the analysis of several algorithms that we present.

The effective methods to compute the logarithmic values of data are divided into two main types; one is the look-up table based algorithms and the other is Iterative methods. This paper presents different algorithms that are used to calculate log_2 showing the accuracy and complexity in hardware design. The remaining of this paper is organized as follows; Section II covers calculating log_2 using a look-up table, Section III shows an Iterative method for calculating log_2 where accuracy depends on the number of iterations, Section IV refers to the

Ahmed M. Mansour is with the Department of Electrical and Communication Engineering, Alexandria University, Egypt (e-mail: ahmed.mansour@wasiela.com).

Ali M. El-Sawy and Ahmed T. Sayed are with wasiela , Cairo ,Egypt (e-mail: ali.mohamed@wasiela.com , ahmed.sayed@wasiela.com).

CORDIC method and how it is used for calculating log_2, Section V presents our proposed method for calculating log_2, Section VI presents our proposed method in calculating log_2 using Taylor series. We compare the algorithms in Section VII, and finally Section VIII presents our conclusions and future work.

II. Look-Up Table Method

This algorithm depends on selecting a certain range of interest and storing a LUT in ROM. It is the traditional method for calculating the logarithm for any base but we must consider memory resources needed for look-up tables and available space. Implementation is direct forward since there are not any decisions taken [1][2].

III. Iterative method

The input value X is first divided into mantissa m and exponent e representation as:

$$X = m2^e \qquad (1)$$

The logarithmic value of X can be expressed in terms of m and e as:

$$log_2(X) = e + log_2(m) = \overbrace{\underbrace{e}_{\text{Interger Part}} + \underbrace{log_2(m)}_{\text{Fraction Part}}}^{log_2(\text{x})} \qquad (2)$$

The logarithmic output is simply the sum of the integer part and the fractional part as shown in Fig. 1. The exponent e is a signed integer and it is exactly the integer part of y. Since the exponent is of base-2 and the mantissa m takes values in the range of [1, 2[, then its logarithmic value is in the range of [0, 1[. Fig. 2, shows the block diagram for calculating $log_2(x)$.
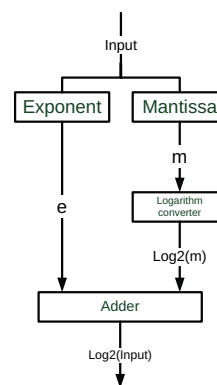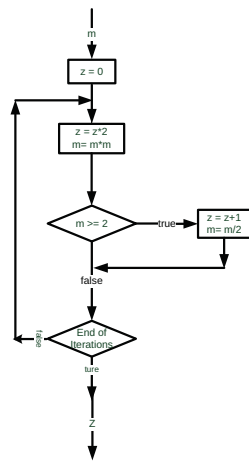

Fig. 1. Main block diagram for Iterative algorithm

World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:8, No:10, 2014

Fig. 2. $Log_2(m)$ calculation

Fig. 3, presents the exact value and the calculated value of $Log_2(m)$ with different number of iterations, showing that as number of iterations increases accuracy increases and vice versa.
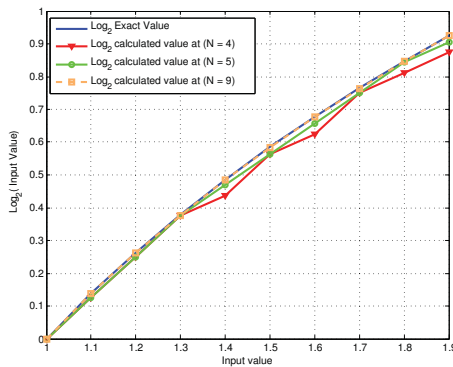

Fig. 3. Exact value and calculated value of $Log_2(m)$

IV. CORDIC METHOD

The CORDIC algorithm uses only adders and shifters. It provides a relatively high precision output, which is suitable for hardware implementation. Calculating log_2 using the CORDIC algorithm depends on linear and inverse hyperbolic tangent modes of CORDIC, then using (4) for conversion between logarithm base-2 and natural logarithm [4][3].

$$ln(m) = 2 * tanh^{-1}(\frac{m-1}{m+1}) \qquad (3)$$

$$log_2(m) = log_e(m) * log_2(e) \qquad (4)$$

The design of CORDIC algorithm for calculating $log_e(X)$ using CORDIC method is presented in Fig. 4.
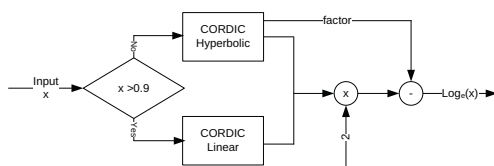

Fig. 4. Main block diagram for CORDIC method

Error analysis shows that our CORDIC design is more accurate than the design presented in [4] for the same example points that we mentioned in Table I showing our minimized error.

TABLE I
ERROR ANALYSIS

Input data	0.9375	0.4375	1.3125	3.375
Error	3.67E-005	1.40E-004	8.31E-005	3.53E-005

V. NEW FLOOR-SHIFT METHOD

This section presents our new method of implementing log_2, let X be a binary number

$$X = X_{N-1}X_{N-2}..........X_1X_0 \qquad (5)$$

Where X_k is the binary digit, $X_k = \{0, 1\}$. Assume that the binary digits $X_{N-1}X_{N-2}....X_{k+1}$ of X are all zeros and X_k is 1. Then the leading one is X_k, thus we get:

$$X = \sum_{i=0}^{k} x_i 2^i = 2^k + \sum_{i=0}^{k-1} x_i 2^i \qquad (6)$$

Since the logarithmic value of X can be expressed in terms of m and e as in (1), so our method tries to calculate $log_2(X)$ directly by getting the integer part then add the fraction part.

Our proposed method based on that presented in [5] which creates a dependence between accuracy and number of iterations, but here we calculate $log_2(X)$ directly independent of the number of iterations; getting the floor of log_2 of the input obtaining the integer part then adding log_2 of the mantissa that presents the fraction part. This operation is described as follows:

1) We calculate $log_2(X)$ by searching for the leading one (i.e., position of X_k) to get the integer part of $log_2(X)$.
2) We can get the approximated value for $log_2(mantissa)$ using (7), this operation is described in Fig. 5, noting that we must have the same length N for data after shifting the decimal point [6].

$$log_2(m) = \frac{x - 2^{\lfloor(log_2(x))\rfloor}}{2^{\lfloor log_2(x)\rfloor}} \qquad (7)$$
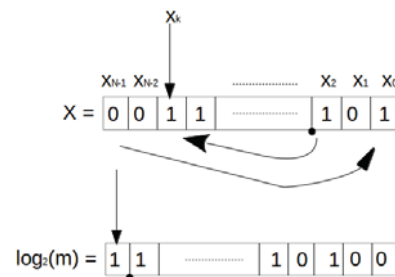

Fig. 5. Floor shift method operation

If we apply (7) directly, we will get a maximum error between exact and calculated values of 0.086, but I we

World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:8, No:10, 2014

combine a simple LUT for certain values range and adding 0.04 as bias to center the error around zero for other values, this modification will minimize the error to half. The next section will present more improvements to this method for calculating mantissa with more accuracy depending on Taylor series.

Ex: $log_2(0.1)$: integer part = $log_2(0.1) = -4$ and fraction part calculation using (7) = 0.6, so $log_2(0.1)$ using this method = (equals) -3.4.

VI. Taylor Series Based Improved Method

This method used Taylor series expansion to calculate log_2 of the mantissa and then add its value to the floor of log_2 of the input as shown in the last section, here we use (8) to get the fraction part using Taylor series as in (9,10). We will demonstrate how error depends on the point of expansion at x=1 and x=1.5.

$$log_2(m) = \frac{X}{2^{\lfloor log_2(X) \rfloor}} \quad (8)$$

A. Taylor Series Expansion Centered at x=1

Taylor series expansion can be written as follows:

$$ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}(x-1)^n \quad (9)$$

Also the conversion equation between $ln(x)$ or $log_e(x)$ to $log_2(x)$ can be rewritten as:

$$log_2(x) = \frac{ln(x)}{ln(2)} = \frac{log_e(x)}{log_e(2)} \quad (10)$$

Fig. 6, shows $log_2(m)$ using Taylor series when taking 3rd, 4th terms and taking the average of both will produce approximated value for exact $log_2(m)$, noting that the accuracy of output value increases as number of terms increases, also noting here the performance of taking average of both 3rd and 4th terms is better than using 5th terms for this case.
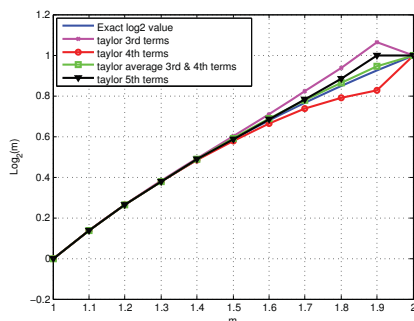
Fig. 6. Exact and calculated values for Taylor series algorithm

The error in this case of algorithm (x= 1) can be minimized if we add certain values to the diverging o.
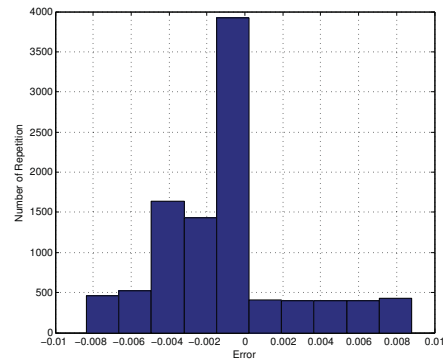Ex: add 0.0119 to values greater than 1.6, the error distribution is shown in Fig. 7.

Fig. 7. Difference between exact and calculated values for Taylor series algorithm with average Taylor terms with modification

B. Taylor Series Expansion Centered at x=1.5

Taylor series expansion can be written as follows:

$$ln(x) = ln(1.5) + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n*(1.5)^n}(x-1.5)^n \quad (11)$$

The fraction part here of $(log_2(m))$ is calculated by using Taylor series expansion taking 4-terms as in (12):

$$log_2(m) = \frac{ln(1.5)}{ln(2)} + \frac{1}{ln(2)} \sum_{n=1}^{4} \frac{(-1)^{n-1}}{n*(1.5)^n}(m-1.5)^n \quad (12)$$

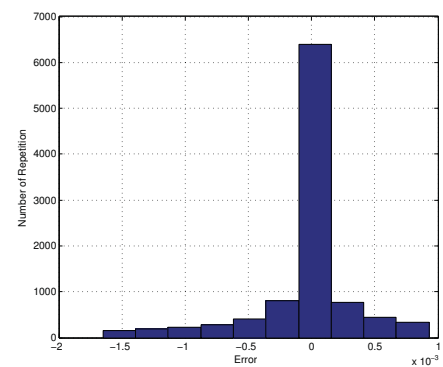This case of algorithm does not need any modification or biasing as shown in the error distribution in Fig. 8.

Fig. 8. Difference between exact and calculated values for Taylor series algorithm with 4 terms Taylor series centered at x = 1.5

VII. Comparison

Fig. 9, presents the error analysis of each algorithm as a function in SQNR and shows the maximum error that can be obtained. It is clear that as SQNR increases the error decreases and vice versa, also we can observe the curves are overlapping together at specific SQNR values giving the same error for all algorithms. If we increase the number of bits, the additional bits will have different effect in reducing the error and increasing SQNR of each algorithm. For example the gain of increasing SQNR of CORDIC method is less than Taylor which is also less than both, the Iterative and LUT methods, also the gain of CORDIC method is higher than floor shift

World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:8, No:10, 2014

method. Table II presents the maximum value of SQNR that we can obtain to get minimum error for each algorithm.
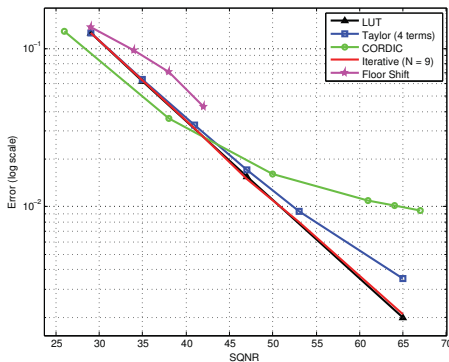

Fig. 9. SQNR Vs Error of each algorithm

Fig. 10, shows that the error distribution at maximum value of SQNR for CORDIC and Iterative algorithms has a uniform distribution around zero, while the mean of the errors in Taylor and LUT algorithms are at zero with a little variance. Floor-Shift algorithm has a uniform distribution of error for the whole range.

Based on error distribution curves in Fig. 10, and the values of SQNR in Table II. We chose the LUT, CORDIC, Taylor and Iterative as the best algorithms for our next clarifying comparison shown in Fig. 11.

TABLE II
MAXIMUM SQNR

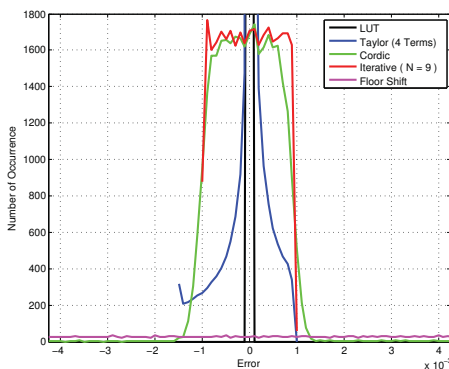| Algorithm	LUT	Iterative N=9	CORDIC	Floor Shifting	Taylor
SQNR	107	71	67	42	75

Fig. 10. Error distribution of each algorithm
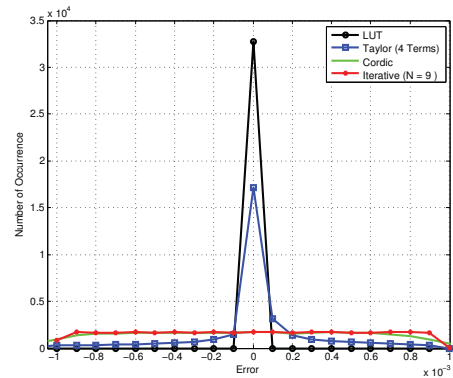
Fig. 11. Error distribution of efficient algorithms

TABLE III
VARIANCE OF EACH ALGORITHM

Algorithm	LUT	Iterative N=9	CORDIC	Floor Shifting	Taylor
Variance	7.60E-011	3.20E-007	9.80E-007	2.80E-004	1.50E-007

VIII. CONCLUSION

This paper presents the most common algorithms that are used for the calculation of log_2. It also proposes a new technique for log_2 calculation that exhibits good trade-off between error, SQNR and accuracy. It is similar to the LUT in that the accuracy is configurable, viz. more bits in LUT provides the same effect on variance as adding terms to the Taylor series or increase number of iterations. The analysis shows that the LUT algorithm is the fastest one but it is not area and power efficient. The Iterative has average speed, area and power but its latency is high compared to the LUT and Taylor. The CORDIC has average speed, but it is a power design.

REFERENCES

[1] H. Hassler and N. Takagi, Function evaluation by table look-up and addition, in Proc.12th Symp.on Computer Arithmetic, pp. 10-16, Jul.1995.
[2] D. DasSarma, D.W. Matula, Measuring the Accumcy of ROM Reciprocal Tables, IEEE 11th Symp.on Computer Arithmetic, pp.932-940, Aug.1994.
[3] Pramod K. Meher, Javier Valls, Tso-Bing Juang, K. Sridharan and Koushik Maharatna, 50 Years of CORDIC: Algorithms, Architectures and Applications, Circuits and Systems I: Regular Papers, IEEE Transactions on (Volume:56, Issue: 9).
[4] Liu Bangqiang, He Ling, Yan Xiao, Base-N Logarithm Implementation on FPGA for the Data with Random Decimal Point Positions, (2013 IEEE 9th International Colloquium on Signal Processing and its Applications, 8-10 Mac. 2013, Kuala Lumpur, Malaysia)
[5] Kostopoulos, D.K, An algorithm for the computation of binary logarithms, Computers, IEEE Transactions on (Volume:40 , Issue: 11).
[6] Tropea, S.E, FPGA Implementation of Base-N Logarithm, Programmable Logic, 2007. SPL07. 2007 3rd Southern Conference.

Table III shows that the Taylor algorithm performance with variance = 1.5e-07 and can be reduced if we increase the number of terms, also the CORDIC algorithm has a variance = 9.8e-07 compared to the variance of the LUT algorithm which is 7.6e-11.