A Hardware-Efficient Logarithmic Multiplier with Improved Accuracy

Mohammad Saeed Ansari
University of Alberta
Edmonton, AB, Canada
ansari2@ualberta.ca

Bruce F. Cockburn
University of Alberta
Edmonton, AB, Canada
cockburn@ualberta.ca

Jie Han
University of Alberta
Edmonton, AB, Canada
jhan8@ualberta.ca

***Abstract*—Logarithmic multipliers take the base-2 logarithm of the operands and perform multiplication by only using shift and addition operations. Since computing the logarithm is often an approximate process, some accuracy loss is inevitable in such designs. However, the area, latency, and power consumption can be significantly improved at the cost of accuracy loss. This paper presents a novel method to approximate $log_2 N$ that, unlike the existing approaches, rounds N to its nearest power of two instead of the highest power of two smaller than or equal to N. This approximation technique is then used to design two improved 16×16 logarithmic multipliers that use exact and approximate adders (ILM-EA and ILM-AA, respectively). These multipliers achieve up to 24.42% and 9.82% savings in area and power-delay product, respectively, compared to the state-of-the-art design in the literature with similar accuracy. The proposed designs are evaluated in the Joint Photographic Experts Group (JPEG) image compression algorithm and their advantages over other approximate logarithmic multipliers are shown.**

***Index Terms*—approximate computing, logarithmic multiplier, hardware-efficient multiplier, JPEG image compression**

I. Introduction

Advances in semiconductor technology, besides providing many benefits, have made digital circuits more vulnerable to parameter variations (process, voltage, and temperature) and soft errors. Thus ensuring strictly accurate computing is getting increasingly challenging [1]. On the other hand, there exist many applications, such as multimedia processing, for which fully accurate results are not required and there might be a range of acceptable results rather than a unique result [2]. Approximate computing can be attractive to such applications due to its potentially significant reduction in design costs while still producing sufficiently accurate results.

Multiplication is a key arithmetic operation that is highly optimized in processors [3]. Logarithmic multipliers convert multiplication into only shift and addition operations, thus allowing it to be done faster with smaller power consumption and area overhead [4]. Logarithmic multipliers are inherently approximate designs due to: (1) a limited number of precision bits and (2) the inaccuracy in computing the function $log(x)$ [5]. Using either piece-wise linear approximations over a finely subdivided input domain or iterative techniques can compensate for the accuracy loss in computing $log_2(x)$ [6].

This work was supported financially by the Natural Sciences and Engineering Research Council of Canada (NSERC) (Project No. RES0018685 and RES0025211).

Among the existing approaches for the hardware implementation of logarithmic conversion, piece-wise polynomial approximation is usually the most efficient solution [4], [7]–[9]. The first piece-wise polynomial approximation was likely proposed by Mitchell [10]. Several Mitchell-based methods have been proposed to improve the accuracy. However, the approximation error in the Mitchell-based approaches is always negative (i.e., the approximate result is smaller than the exact value) [6]. This systematic error causes problems in repetitive or iterative operations, such as matrix multiplications, since the errors do not cancel and are accumulated.

This paper proposes a novel approximation method with a double-sided error distribution that can be used as a more accurate baseline design instead of the Mitchell approach. Two improved logarithmic multipliers are constructed using exact and approximate adders, referred to as ILM-EA and ILM-AA, respectively. They are then evaluated in the Joint Photographic Experts Group (JPEG) application.

The remainder of this paper is organized as follows: Section II presents the proposed approximation approach and the resulting logarithmic multipliers. Section III discusses the error and hardware performance of the proposed and state-of-the-art logarithmic multipliers. The JPEG application is considered in Section IV to evaluate the proposed design. Finally, Section V concludes the paper.

II. Proposed Techniques

Let Z be the n-bit binary representation of a positive integer N. Without the loss of generality, let $z_k, k \leq n$, be the most significant '1' in Z. Hence, N can be represented as:

$$N = 2^k(1+x) = 2^{k+1}(1-y), \tag{1}$$

where $\quad 0 \leq x, y \leq 1$.

A. Proposed approximation approach

The conventional approximation approaches use the highest power of two smaller than the given number N. Instead, we propose the approximation given in Algorithm 1. Note that $2^k \leq N < 2^{k+1}$. Let d_1 and d_2 denote the differences $N - 2^k$ and $2^{k+1} - N$, respectively. As shown in Algorithm 1, when $d_1 < d_2$ we underestimate the value of $log_2 N$; otherwise, we overestimate it.

The exact, Mitchell, and the proposed methods for computing $log_2 N$ are plotted in Fig. 1. Note that the proposed approximation results in more than 6× smaller average error (over the range [1, 255]) than the Mitchell method (0.0088 vs. 0.0568), which is due to the double-sided error distribution of the proposed approach.

Algorithm 1 Proposed approximation for $log_2 N$

```
1:  $N = 2^k(1+x) = 2^{k+1}(1-y)$ where $0 \le x, y \le 1$
2:  $d_1 = N - 2^k$                ▷ error in underestimate
3:  $d_2 = 2^{k+1} - N$            ▷ error in overestimate
4:  if $d_1 < d_2$ then            ▷ use underestimate
5:      $x = d_1/2^k$
6:      $log_2 N \approx k + x$
7:  else                           ▷ use overestimate
8:      $y = d_2/2^{k+1}$
9:      $log_2 N \approx k + 1 - y$
10: end if
```

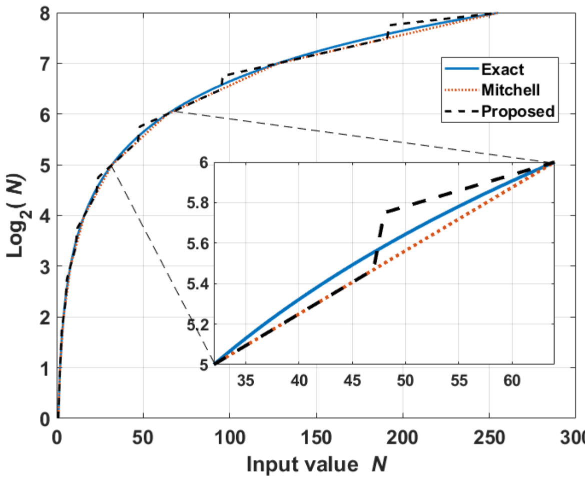


Fig. 1: Approximation of $log_2 N$.

B. Proposed multiplier designs

1) High-level description: The proposed multiplier first transforms the multiplicand α and multiplier β to the closest powers of two plus an additional term. Hence, the multiplication can be rewritten as:

$$\alpha = m_1 + q_1, \text{ where } m_1 = 2^{k_1} \tag{2}$$

$$\beta = m_2 + q_2, \text{ where } m_2 = 2^{k_2} \tag{3}$$

$$\alpha \times \beta \approx (2^{k_1+k_2} + q_2 2^{k_1} + q_1 2^{k_2}) + \cancel{q_1 q_2}. \tag{4}$$

As shown in (4), the three most significant terms are all multiplications by powers of two that can be easily performed as left-shift operations in hardware. In our design, the least significant term ($q_1 q_2$) is ignored and left as the approximation error. A more detailed description of the proposed approximate multiplier is provided in Algorithm 2, where NOD, PE and DEC stand for the nearest-one detector, the priority encoder, and the decoder. Detailed descriptions of these three components are given in the following subsection.

Algorithm 2 Proposed approximate multiplier

```
1:  procedure M($\alpha$, $\beta$)
2:      $\alpha$, $\beta$: inputs, $\gamma$: approximate output
3:      $m_1 \leftarrow$ NOD($\alpha$),
4:      $k_1 \leftarrow$ PE($m_1$),
5:      $q_1 \leftarrow \alpha - m_1$,              ▷ for steps 3-5 see (2)
6:      $m_2 \leftarrow$ NOD($\beta$),
7:      $k_2 \leftarrow$ PE($m_2$),
8:      $q_2 \leftarrow \beta - m_2$,               ▷ for steps 6-8 see (3)
9:      $q_1 2^{k_2} \leftarrow q_1 << k_2$,
10:     $q_2 2^{k_1} \leftarrow q_2 << k_1$,
11:     $2^{k_1+k_2} \leftarrow$ DEC($k_1 + k_2$),
12:     $\gamma \leftarrow 2^{k_1+k_2} + q_2 2^{k_1} + q_1 2^{k_2}$. ▷ for steps 9-12 see (4)
```

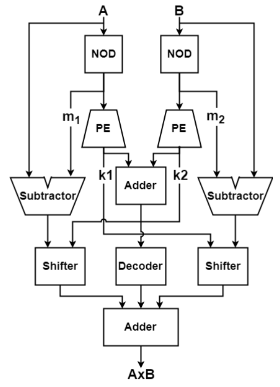
2) Hardware implementation: The proposed approximate multipliers can be implemented by either: (1) implementing the logic to calculate the nearest powers of two or (2) using look-up tables (LUTs). We decided not to use LUTs for the following two reasons:

- The algorithm for finding the two closest powers of two in the proposed approximate multiplier is simple and can be easily done in hardware.
- Memory usage is a serious bottleneck for many applications, such as in neural networks [11], [12]. Hence, avoiding LUTs helps the proposed design to be applicable for a broader range of applications.

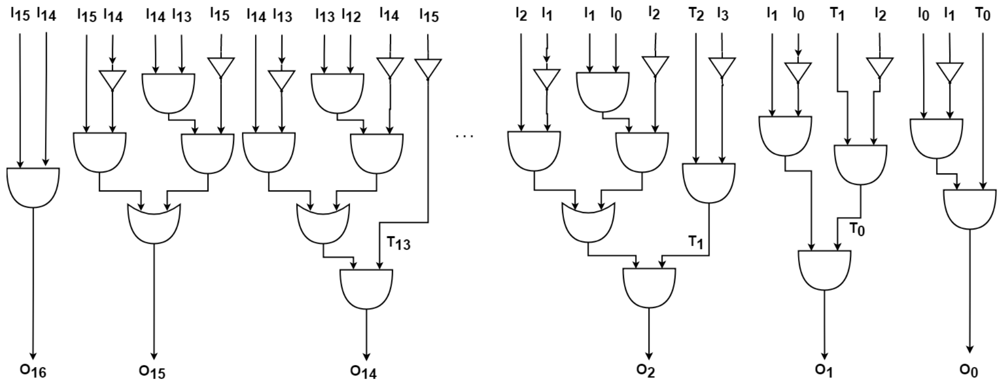
The block diagram of the proposed approximate multiplier is given in Fig. 2(a). The nearest-one detector (NOD) circuit (Fig. 2(b)) is based on a leading-one detector (LOD) circuit. However, unlike the LOD, NOD finds the nearest power of two to its given input. Note that similar to some existing LODs [13], [14], the proposed NOD evaluates from the MSB to the LSB.

The priority encoder (PE) then determines the number of required shifts based on the NOD's output. The two residue terms q_1 and q_2 are also calculated and shifted according to the k_2 and k_1 values, respectively, and a decoder generates the most significant term, $2^{k_1+k_2}$. Finally, the three resulting terms are summed up to obtain the approximate product.

As mentioned before, the Mitchell multiplier always underestimates the actual product. To address this issue, the authors in the recent paper [6] used a set-one-adder (SOA) with k approximation bits, SOA-k. The SOA-k puts 1 for the k LSBs and therefore, always overestimates the actual result. This is used in [6] to improves the accuracy of the Mitchell multiplier with less hardware cost than the Mitchells'. Our proposed design, on the other hand, might round up or down, depending on the input operands and, therefore, it has a double-sided error distribution. This is one major benefit of our proposed design compared to the existing designs in the literature.



(a) Block diagram of the proposed logarithmic multiplier.



(b) Proposed 16-bit nearest-one detector (NOD) circuit.

Fig. 2: The proposed approximate logarithmic multiplier design.

To further improve the hardware efficiency, we used approximate adders to accumulate the base-2 logarithms of the input operands. We used a modified SOA-k adder in which instead of setting all of the k LSBs to '1' they are set to '1' and '0' alternatively. For instance, for SOA-7 we use "0101010" instead of "1111111". By doing so, the resulting adder sometimes overestimates and sometimes underestimates the exact result. Therefore, the double-sided error distribution property in the proposed approximate multiplier ILM-AA is also preserved.

III. Performance Evaluation

A. Accuracy metrics

The three accuracy metrics (1) error rate (ER), (2) the mean relative error distance (MRED), and (3) the normalized mean error distance (NMED, the mean error distance normalized by the maximum output of the accurate design) [2], [15] were calculated for 16×16 logarithmic multipliers and the results are given in Table I.

TABLE I: Error characteristics of logarithmic multipliers.

Multiplier Type	MRED	ER(%)	NMED
LM [10]	0.0384	99.77	0.0092
ALM-SOA-11 [6]	0.0330	**98.97**	0.0080
This work: ILM-AA	0.0290	99.99	0.0072
This work: ILM-EA	**0.0289**	99.95	**0.0069**

Since the exhaustive simulation is too time consuming, we generated a sample of 10^7 random cases to obtain the results in Table I. Moreover, the effect of the number of approximation bits on the accuracy of the 16×16 logarithmic multipliers is evident in Fig. 3. According to Fig. 3, using more than eleven bits for approximation significantly reduces the accuracy of ALM-SOA, while the proposed ILM-AA is more robust. However, we also used eleven bits for approximation in our analysis for a fair comparison.

The results in Table I show that the proposed ILM-EA (using exact adders) and ILM-AA (using approximate adders) are the most accurate with respect to MRED and NMED.

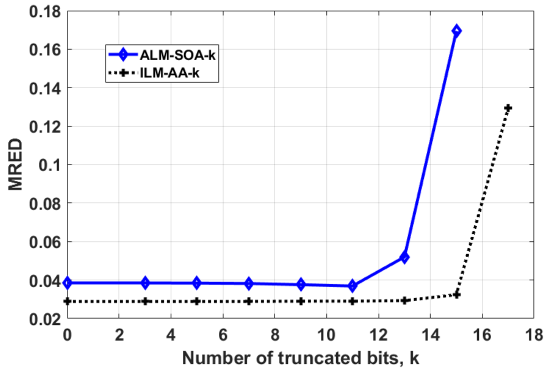


Fig. 3: Effects of the number of truncated bits on the accuracy of 16×16 logarithmic multipliers.

B. Hardware metrics

We implemented all of the designs in VHDL and then synthesized them using the Synopsys Design Compiler (DC) for ST Micro's CMOS 28-nm process. Note that the default settings for DC were used for all the simulations to ensure a fair comparison.

As shown in Table II, both the fastest and smallest design is ILM-AA, which is 41.48% faster and 40.77% smaller than the base Mitchell design, while being 24.48% more accurate than it. With respect to power, ALM-SOA consumes 3.62% less power than the proposed ILM-AA, while ILM-AA has the lowest PDP value among the four considered designs.

TABLE II: Hardware characteristics analysis.

Multiplier Type	Power (μW)	Delay (nS)	Area (μm^2)	PDP (fJ)
LM [10]	94.97	2.29	357.41	217.48
ALM-SOA-11 [6]	**67.97**	1.54	281.35	104.65
ILM-AA	70.43	**1.34**	**211.67**	**94.37**
ILM-EA	140.00	2.26	387.43	316.40

Table III compares the PDP-MRED product for the four considered multipliers. The designs with lower MRED and smaller PDP and PDP-MRED product are, consequently, preferable. Hence, the proposed ILM-AA is the most hardware-efficient design.

TABLE III: Comparison of PDP-MRED products for the logarithmic multipliers.

Multiplier Type	PDP × MRED
LM [10]	8.35
ALM-SOA-11 [6]	3.45
ILM-AA	**2.73**
ILM-EA	9.14

IV. JPEG Application

JPEG was an early and still important image compression standard [16]. The first step in the JPEG algorithm is to partition the pixels of an image I into 8×8 blocks [17]. Then, the Discrete Cosine Transform (DCT) is applied to each block and the resulting DCT coefficients are quantized in order to minimize the high-frequency content. The quantization is accomplished by dividing the matrix of coefficients element-wise by the quantization matrix. Note that we used the quantization matrix (Q) in [18] corresponding to quality factor QF=10. Decompression can be done by performing all of these steps in the reverse order. First, we multiply (by using approximate multipliers) the obtained quantized matrix from the previous step by Q to generate the matrix of DCT coefficients. The inverse DCT is then used to recover the original component matrix. Finally, all of the 8×8 blocks are reassembled to form an image, I', of the same size as the original image I. In our experiment, the image '$Lena$' is compressed by using the JPEG algorithm with the standard quantization matrix (Q) [18].

The quality of the five decompressed images, one using an exact multiplier and four using logarithmic multipliers, are compared by using the structural similarity (SSIM) measure [19]. As shown in Table IV, the proposed designs ILM-EA and ILM-AA produce the highest SSIM values next to the exact multiplier.

TABLE IV: SSIM values for decompressed images using exact and logarithmic multipliers.

Multiplier Type	SSIM
Exact multiplier	0.7743
LM [10]	0.7618
ALM-SOA-11 [6]	0.7615
ILM-AA	0.7631
ILM-EA	0.7631

V. Conclusion

This work proposes a novel approximation method to efficiently compute log_2N. Using this method, two improved logarithmic multipliers are designed, ILM-EA and ILM-AA. Both designs are highly accurate and have the smallest MRED values compared to other logarithmic designs in the literature. The less accurate ILM-AA multiplier is 25% smaller than the recent design in [6] and has the smallest PDP and latency. In fact, it is 9.82% more energy-efficient, 12.98% faster, and 12.42% more accurate than the logarithmic multipliers in [6]. Finally, JPEG image compression was considered as an application, for which the proposed designs show a higher output image quality than the other designs.

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *18th IEEE European Test Symposium (ETS)*, pp. 1–6, 2013.

[2] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "Low-power approximate multipliers using encoded partial products and approximate compressors," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 404–416, 2018.

[3] M. de la Guia Solaz, W. Han, and R. Conway, "A flexible low power DSP with a programmable truncated multiplier," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2555–2568, 2012.

[4] J. Y. L. Low and C. C. Jong, "Unified Mitchell-based approximation for efficient logarithmic conversion circuit," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1783–1797, 2015.

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

[6] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[7] D. De Caro, N. Petra, and A. G. Strollo, "Efficient logarithmic converters for digital signal processing applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 10, pp. 667–671, 2011.

[8] T.-B. Juang, S.-H. Chen, and H.-J. Cheng, "A lower error and rom-free logarithmic converter for digital signal processing applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 12, pp. 931–935, 2009.

[9] B.-G. Nam, H. Kim, and H.-J. Yoo, "Power and area-efficient unified computation of vector and elementary functions for handheld 3D graphics systems," *IEEE Transactions on Computers*, vol. 57, pp. 490–504, 2008.

[10] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, no. 4, pp. 512–517, 1962.

[11] G. Srinivasan, P. Wijesinghe, S. S. Sarwar, A. Jaiswal, and K. Roy, "Significance driven hybrid 8t-6t SRAM for energy-efficient synaptic storage in artificial neural networks," in *Design, Automation & Test in Europe Conference Exhibition (DATE)*, pp. 151–156, 2016.

[12] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 2, p. 16, 2018.

[13] K. H. Abed and R. E. Siferd, "VLSI implementations of low-power leading-one detector circuits," in *Proceedings of the IEEE SoutheastCon*, pp. 279–284, IEEE, 2005.

[14] K. Kunaraj and R. Seshasayanan, "Leading one detectors and leading one position detectors-an evolutionary design methodology," *Canadian Journal of Electrical and Computer Engineering*, vol. 36, no. 3, pp. 103–110, 2013.

[15] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 4, p. 60, 2017.

[16] G. K. Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[17] T. Suzuki and T. Yoshida, "Lower complexity lifting structures for hierarchical lapped transforms highly compatible with JPEG XR standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2652–2660, 2016.

[18] J. Yang, G. Zhu, and Y.-Q. Shi, "Analyzing the effect of JPEG compression on local variance of image intensity," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2647–2656, 2016.

[19] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.