Hardware-Efficient Logarithmic Floating-Point Multipliers for Error-Tolerant Applications

Zijing Niu, Tingting Zhang, *Student Member, IEEE,* Honglan Jiang, *Member, IEEE,* Bruce F. Cockburn, *Member, IEEE,* Leibo Liu, *Senior Member, IEEE,* and Jie Han, *Senior Member, IEEE*

***Abstract*—The increasing computational intensity of important new applications poses a challenge for their use in resource-restricted devices. Approximate computing using power-efficient arithmetic circuits is one of the emerging strategies to reach this objective. In this article, five hardware-efficient logarithmic floating-point (FP) multipliers are proposed, which all use simple operators, such as adders and multiplexers, to replace complex and more costly conventional FP multipliers. Radix-4 logarithms are used to further reduce the hardware complexity. These designs produce double-sided error distributions to mitigate error accumulation in complex computations. The proposed multipliers provide superior trade-offs between accuracy and hardware, with up to 30.8% higher accuracy than a recent logarithmic FP design or up to $68\times$ less energy than the conventional FP multiplier. Using the proposed FP logarithmic multipliers in JPEG image compression achieves higher image quality than a recent logarithmic multiplier design with up to 4.7 dB larger peak signal-to-noise ratio. For training in benchmark NN applications, the proposed FP multipliers can slightly improve the classification accuracy while achieving $4.2\times$ less energy and $2.2\times$ smaller area than the state-of-the-art design.**

***Index Terms*—Floating-Point Multiplier, Logarithmic Multiplier, Neural Network, Approximate Computing, JPEG Compression**

I. Introduction

DUE to the scaling of complementary metal–oxide semiconductor (CMOS) technology, the power density has increased significantly in integrated circuits [1]. The substantial increase in power becomes a limiting factor for computation-intensive applications, such as machine learning (ML) and digital signal processing (DSP), in resource-constrained devices [2] [3]. The computations in many important recent applications involve massive multiply-accumulate (MAC) operations that require a significant amount of energy. This challenge motivates the development of more efficient arithmetic circuits for emerging computing systems.

Many computation-intensive applications, such as image processing and deep neural networks (DNNs), can tolerate a degree of computational error [4]. Approximate computing (AC) has emerged as a promising strategy to improve the energy efficiency in such systems [5]. In particular, AC has been exploited extensively in hardware implementations, including approximate arithmetic circuit design, voltage over-scaling and precision scaling techniques [6], [7].

Floating-point (FP) arithmetic is often favored in applications that require sufficient accuracy over a wide dynamic range. Since FP MAC circuits, especially the multipliers therein, dominate the power dissipation and circuit area, efficient FP multiplier design has been of interest [8]–[14]. Nevertheless, FP multipliers are relatively underexplored compared to their fixed-point counterparts and only a few of them have been applied to applications, such as image processing and, in particular, the training process of DNNs. Although the bit width reduction in the FP representation for the training of DNNs increases efficiency [15] [16], precision reduction can cause a significant deterioration in accuracy. Therefore, designs of approximate FP multipliers are promising to further improve the hardware efficiency for error-tolerant applications.

As an alternative to the conventional FP representation, logarithmic representations of FP numbers have been considered for the acceleration of NNs. For example, Lognet shows that logarithmic computation can enable a more accurate encoding of weights and activations that results in higher classification accuracies at low resolutions [17]. Efficient FP logarithmic multipliers (LMs) have become promising for NN training [18]. A state-of-the-art 4-bit training strategy for DNNs has been developed for the logarithmic radix-4 representation [19]. Significant progress has been made in exploiting reduced-precision integers for inference; 8-bit FP representation [20] and logarithmic 4-bit FP representation [19] have shown their effectiveness in the training of DNNs. Lastly, application-driven AC techniques have been developed to achieve domain-specific hardware-efficient NN implementations [21] [22].

In this article, we propose five energy-efficient FP LMs for error-tolerant, computation-intensive applications. Two novel approximation methods are described for the logarithm and anti-logarithm conversions that generate double-sided error distributions. The multiplier designs consist of simple operators, such as adders and multiplexers, which lead to lower power and area costs compared to conventional FP multipliers. Finally, the application-level performance of the proposed FP multipliers is evaluated for an image processing application and several benchmark NNs. It is also found that the classification accuracy can be slightly improved in some cases with a significant reduction in energy consumption.

Some of our preliminary results were reported in [23]. The novel contributions of this article are summarized as follows:

- A new approximation method for logarithm and anti-

Z. Niu, T. Zhang, B. F. Cockburn and J. Han are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada. E-mail: zijing2@ualberta.ca, ttzhang@ualberta.ca, cockburn@ualberta.ca, jhan8@ualberta.ca

H. Jiang is with the Department of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: honglan@sjtu.edu.cn

L. Liu is with the Institute of Microelectronics, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China. E-mail:liulb@tsinghua.edu.cn

logarithm conversions generates a double-sided error distribution to minimize the accumulation of approximation error in sums of products.
- Four new FP LMs exploit circuit optimizations. In particular, a radix-4 logarithm (R4L) approach is developed that further reduces the hardware complexity.
- A detailed and comprehensive error evaluation reveals the relation between the FP precision format and the error behavior of the proposed multipliers.
- The proposed designs improve the hardware efficiency in JPEG image compression and the training of NNs.

The rest of this article is organized as follows: In Section II, technical background and related work are reviewed. Section III introduces the proposed logarithmic approximation methods and logarithmic multiplications. Circuit designs of five proposed FP multipliers and the proposed radix-4 logarithm hardware reduction approach are presented in Section IV. Their performance with respect to accuracy and hardware are given. Section V presents case studies of error-tolerant applications. Finally, conclusions are provided in Section VI.

II. Background

A. Theory of Logarithm Approximation

Let Z be a number in the binary representation $Z = 2^e(1 + k)$, where k $(0 \leq k < 1)$ denotes the fractional part and e is the exponent. Mitchell first proposed a simple method for a logarithmic conversion, given in [24]:

$$\log_2 Z \cong e + k. \tag{1}$$

If $\log_2(1 + k)$ is approximated by l, then Mitchell's antilogarithm approximation can be expressed as:

$$2^l \cong l + 1, \tag{2}$$

where $0 \leq l < 1$.

Mitchell's approximation offers both high speed and low hardware cost. However, it always underestimates the true logarithm, which cause negative errors to accumulate in MAC operations. To produce a double-sided error distribution, a nearest-one logarithmic approximation finds the nearest power of two for Z [25]. When $Z - 2^e < 2^{e+1} - Z$, it uses the same logarithm as Mitchell's method; otherwise, Z is given by $Z = 2^{e+1}(1 - y)$, where $0 \leq y < 0.25$, and thus the logarithm is approximated as:

$$\log_2 Z \cong e + 1 - y. \tag{3}$$

B. Review of Approximate Multipliers

1) Logarithmic Multipliers (LMs): Mitchell's approximation has been the basis for many other LMs. Operand decomposition improves the accuracy of Mitchell's approximation by dividing the two inputs into four to reduce the number of '1's in the inputs [26]. An improved operand decomposition algorithm further reduces the energy consumption in LMs [27]. The iterative LM (ILM) improves the accuracy through a pipelined implementation with an error correction circuit that leads to iterative calculations of compensation terms [28]. A truncated ILM further reduces the hardware complexity [29]. A low-cost two-stage ILM compensates for errors in the addition of approximate logarithms [30].

To further improve the hardware efficiency, Mitchell's algorithm has been combined with other approximation techniques and optimized implementations of components. A customizable signed LM utilizes logarithmic approximation and truncation of operands, in which the one's complement representation is adopted to imprecisely handle negative numbers [31]. A cost-efficient two-stage logarithmic design uses a truncated LM in an iterative structure [30]. In both the non-iterative and iterative LMs, approximate adders are used to add the mantissas to reduce energy and improve accuracy [32]. A set-one adder (SOA) sets the lower significant bits to '1's to compensate the accumulated errors in sums generated in Mitchell's LM. The lower significant sum bits are computed using OR gates in the lower-part-or adder (LOA) and are set as one of the inputs in the approximate mirror adder (MAA3) [33] [34]. A two-stage LM employs different trimming strategies in the least significant parts of the input operands and the mantissas of the trimmed operands, which leads to lower energy and area cost [35]. A dynamic range LM relies on a truncation scheme to dynamically compensate for the accumulated errors generated by Mitchell's approximation [36].

Unlike Mitchell's approximation-based LMs, an improved LM uses the nearest-one logarithmic approximation method to produce double-sided error distributions [25]. A nearest-one detector (NOD) was proposed to first detect the nearest power of two and then produce the approximate logarithm.

2) Approximate Floating-Point Multipliers: Truncation and voltage over-scaling techniques are commonly used for approximate FP multipliers [37]–[41]. A configurable approximate FP multiplier utilizes the K-nearest neighbor (kNN) algorithm to determine the truncated bits of a given input that can minimize energy and area [8]. Using Mitchell's approximation, a logarithmic approximate FP multiplier (LAM) improves the energy efficiency of NN training [14]. Moreover, configurable FP multipliers have been studied to provide different levels of accuracy in real-time [9] [10]. In [9], a configurable floating-point unit avoids multiplication by discarding one mantissa or adding and shifting two mantissas. The multiplication is also replaced with addition in a runtime-configurable FP multiplier, using an accuracy-tuning method [10]. However, neither of these two configurable FP multipliers completely eliminates multiplication since exact multiplication is still required when the error rate exceeds a pre-determined value.

Approximate fixed-point multipliers are required for the mantissa multiplication. For example, an approximate modified Booth encoding (AMBE) algorithm generates the partial products and then uses an inexact 4-2 compressor to optimize the mantissa multiplier [11]. This design also adopts bit truncation in the partial products to generate variable accuracy. Three approximate FP units are developed by proposing an approximate speculative multiplier and using gate-level pruning with an approximate speculative adder [12].

III. Representation, Formulation and Approximation for Multiplication

A. FP Representation and Multiplication

1) IEEE 754 FP Representation: The IEEE 754 standard defines the most commonly used formats for FP numbers. These formats all contain a 1-bit sign S, a w-bit exponent E and a q-bit mantissa M [42]. Fig. 1 shows the IEEE 754 representation of a single-precision FP number.

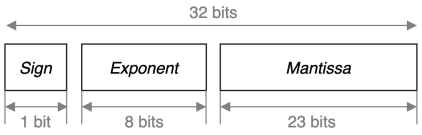


Fig. 1: The IEEE-754 single-precision format.

In this format, a number N can be expressed in a base-2 scientific notation as follows:

$$N = (-1)^S \cdot 2^{E-bias} \cdot (1+x), \tag{4}$$

where S is either 0 for a positive number or 1 for a negative number. To ensure unsigned integers in the exponent field, a bias $(2^{(w-1)}-1)$, such as 127 for the single-precision, is added to the actual exponent value to get E. Thus $E-bias$ denotes the actual exponent value of N. With the hidden '1', x is the fractional part of the FP number, and hence $0 \leq x < 1$.

2) Floating-Point Multiplication: In the IEEE 754 format, the FP multiplication involves three processes, including an XOR operation for the sign bits, the addition of the exponents, and the multiplication of the mantissa bits. Note that X is used to denote the actual mantissa, $1+x$, in the following formulation. Consider the product $P = A \times B$, which is computed as follows:

$$S_P = S_A \oplus S_B, \tag{5}$$

$$X_{AB} = (1+x_A) \times (1+x_B), \tag{6}$$

$$E_P = \begin{cases} E_A + E_B - bias, & X_{AB} < 2, \\ E_A + E_B - bias + 1, & otherwise, \end{cases} \tag{7}$$

$$X_P = \begin{cases} X_{AB}, & X_{AB} < 2, \\ X_{AB}/2, & otherwise, \end{cases} \tag{8}$$

where the sign bit, exponent and mantissa of A, B and P are, respectively, denoted with the corresponding subscripts. The exponent and mantissa of product P relate to the comparison of the obtained mantissa X_{AB} with 2. Note that (5) is valid for the sign computation of the proposed designs, so it will not be discussed hereafter.

B. Logarithmic Approximation and Mathematical Formulations for Multiplication

Since Mitchell's approximation method computes underestimated products that lead to error accumulation in a MAC summation, two novel approximation methods that produce a double-sided error distribution are proposed.

1) Logarithmic Multiplication 1:

a) Logarithmic FP Representation: The proposed approximation method-1 is based on the conversion of a logarithmic FP representation that differs from the IEEE 754 format. An FP number N is first converted into the format using its nearest power of two, as per (3), and the corresponding mantissa. Since the IEEE 754 FP format provides the largest power of two smaller than N, by comparing the fraction x in (4) with 0.5, the nearest power of two can be determined for N.

If $x \geq 0.5$, then N is closer to $2^{E-bias+1}$ than 2^{E-bias} (or equally away for the equality case). The exponent E is incremented by 1 and, accordingly, the mantissa becomes $\frac{1+x}{2}$. In contrast, the exponent and mantissa of N remain the same as in (4) when $x < 0.5$. Let the converted exponent of N be denoted by $E^{'}$ and the converted mantissa by $X^{'}$. Then, for approximation method-1 $E^{'}$ and $X^{'}$ are given by:

$$E^{'} = \begin{cases} E, & x < 0.5, \\ E+1, & x \geq 0.5, \end{cases} \tag{9}$$

$$X^{'} = 1 + x^{'} = \begin{cases} 1+x, & x < 0.5, \\ \frac{1+x}{2}, & x \geq 0.5, \end{cases} \tag{10}$$

where $0.75 \leq X^{'} < 1.5$.

b) Logarithm and Anti-logarithm Approximation: To better introduce the approximation method and its usage in multiplication, different notations are used. Consider the logarithm of $1+k$, i.e., $\log_2(1+k)$ and the anti-logarithm of l, i.e., 2^l.

Due to the conversion in representation as per (10), the logarithm is applied to $X^{'}$. Thus the range of $1+k$ is $[0.75, 1.5)$ and the logarithm approximation method-1 is applied over the domain $-0.25 \leq k < 0.5$.

Since l is obtained as the sum of the two approximate logarithms, whose range is $[-0.5, 1)$, the anti-logarithm approximation method-1, as described in (2), is applied over the domain region $-0.5 \leq l < 1$.

The logarithm approximation method-1 of the function $\log_2(1+k)$ is shown in Fig. 2, in which it is compared with Mitchell's method and the exact function. Note that method-1 computes the same underestimated results as Mitchell's method when both input operands are closer to 2^{E-bias}. However, the input operands closer to $2^{E-bias+1}$ are converted to overestimated logarithm values. As shown in Fig. 3, the anti-logarithm approximation method-1 produces the same overestimated results as Mitchell's method when l is in the range of $[0, 1)$, whereas it underestimates the results when l is negative. As a result, the proposed approximation method-1 can compute either underestimated or overestimated results, depending on the input operands.

c) Mathematical Formulations: For the logarithmic N using the logarithmic FP representation, the exponent is given as (9) and the logarithm is approximated as follows:

$$\log_2(X^{'}) = \log_2(1+x^{'}) \cong \begin{cases} x, & x < 0.5, \\ \frac{1+x}{2} - 1, & x \geq 0.5. \end{cases} \tag{11}$$

For $P = A \times B$ in the logarithm domain, the exponent is still computed by addition, whereas the multiplication of the

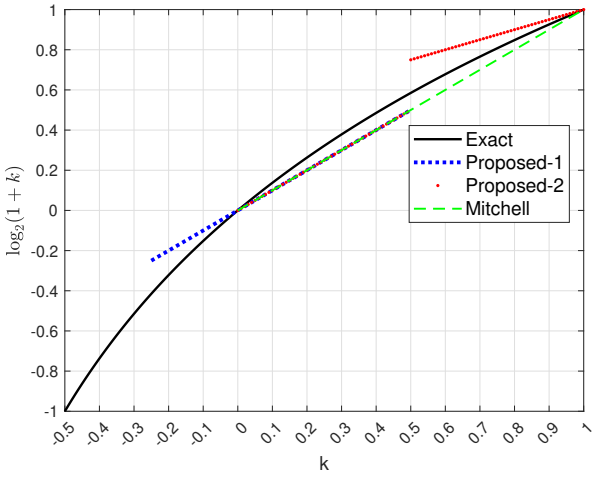


Fig. 2: Approximations of $\log_2(1+k)$.

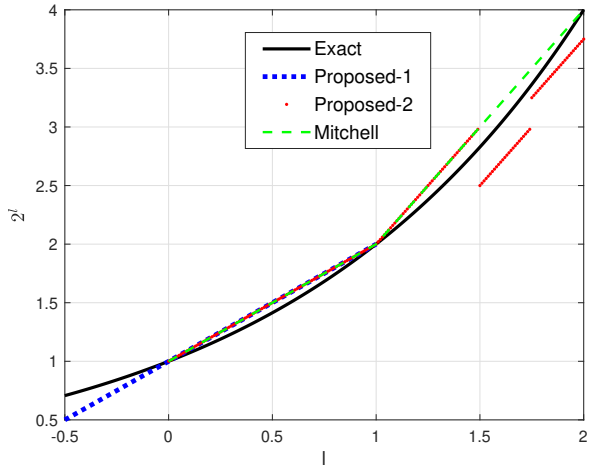


Fig. 3: Approximations of 2^l.

mantissas is converted to addition. Let $X_{AB}^{'} = X_A^{'} \times X_B^{'} = (1+x_A^{'}) \times (1+x_B^{'})$, and hence the logarithm of $X_{AB}^{'}$ is given by:

$$\log_2(X_{AB}^{'}) = \log_2(1+x_A^{'}) + \log_2(1+x_B^{'}) \cong \widehat{X}_A + \widehat{X}_B, \quad (12)$$

where $\widehat{X}_A$ and $\widehat{X}_B$ denote the approximate logarithms obtained using (11).

Using the anti-logarithm approximation method-1, $X_{AB}^{'}$ is obtained as:

$$X_{AB}^{'} \cong 2^{\widehat{X}_A + \widehat{X}_B} \cong 1 + \widehat{X}_A + \widehat{X}_B. \quad (13)$$

According to (11), we obtain $-0.5 \leq \widehat{X}_A + \widehat{X}_B < 1$, and thus, $0.5 \leq X_{AB}^{'} < 2$. When $X_{AB}^{'} < 1$ (or $\widehat{X}_A + \widehat{X}_B < 0$), $X_{AB}^{'}$ cannot be directly represented as the mantissa for the product P. In this case, $X_{AB}^{'}$ is multiplied by 2 and, accordingly, the exponent is reduced by 1.

Finally, the logarithmic FP multiplication in the proposed multiplication-1 method is given by:

$$E_P = \begin{cases} E_A^{'} + E_B^{'} - bias, & \widehat{X}_A + \widehat{X}_B \geq 0, \\ E_A^{'} + E_B^{'} - bias - 1, & otherwise, \end{cases} \quad (14)$$

$$X_P = \begin{cases} 1 + \widehat{X}_A + \widehat{X}_B, & \widehat{X}_A + \widehat{X}_B \geq 0, \\ (1 + \widehat{X}_A + \widehat{X}_B) \times 2, & otherwise. \end{cases} \quad (15)$$

Note that $E_A^{'}$ and $E_B^{'}$ are the converted exponents, and $\widehat{X}_A$ and $\widehat{X}_B$ are the approximate logarithms given in the converted mantissas for A and B, respectively.

2) Logarithmic Multiplication 2:

a) Logarithm and Anti-logarithm Approximation: Consider a given number N, when $x \geq 0.5$, $\log_2 N = (E+1) + (\frac{1+x}{2} - 1)$ according to (11). By cancelling out the '+1' and '-1', we obtain $\log_2 N = E + \frac{1+x}{2}$, which leads to the logarithm approximation method-2 given by (16). Note that the conversions for the original exponent and the mantissa in (9) and (10) are avoided in the approximation.

$$\log_2(1+k) \cong \begin{cases} k, & 0 \leq k < 0.5, \\ \frac{(1+k)}{2}, & 0.5 \leq k < 1, \end{cases} \quad (16)$$

where the range of the approximate logarithm is $[0, 1)$. Thus, the range of l is $[0, 2)$ (note that l is the sum of the two approximate logarithms). Consider using Mitchell's anti-logarithm method, as shown in Fig. 3, when $l \in [1.5, 2)$, larger approximate results will be produced in both the logarithm and anti-logarithm processes, which can lead to relatively large error accumulation. Therefore, we propose to reduce the anti-logarithm value by subtracting certain values to ensure positive errors. The subtracted value is selected to reduce the error accumulation effect as much as possible while introducing relatively small hardware overhead. The trade-off is assessed when determining the subtracted value. Using different subtracted values with excessively small intervals in the range of $[1.5, 2)$ can substantially increase the hardware overhead, whereas subtracting one single value in the range of $[1.5, 2)$ can be ineffectual or aggravate the error accumulation. Based on empirical experiments for computing the overall accumulative error with various subtracted values, 0.5 and 0.25 were selected for the domains $[1.5, 1.75)$ and $[1.75, 2)$, respectively. The anti-logarithm approximation used in this method is expressed as:

$$2^l \cong \begin{cases} l + 1, & l < 1, \\ 2 \times l, & 1 \leq l < 1.5, \\ 2 \times l - 0.5, & 1.5 \leq l < 1.75, \\ 2 \times l - 0.25, & 1.75 \leq l < 2, \end{cases} \quad (17)$$

The logarithm approximation method-2 is shown in Fig. 2. When the mantissas of both input operands are smaller than 0.5, method-2 computes the same underestimated logarithm approximation as Mitchell's method; otherwise, it computes overestimated results. As shown in Fig. 3, the same overestimated anti-logarithm approximation as Mitchell's method is produced when $l \in [0, 1.5)$, whereas underestimated anti-logarithm approximations are generated when $l \in [1.5, 2)$.

b) Mathematical Formulations: Using the proposed logarithm approximation method-2, as per (16) and the anti-logarithm approximation, as per (17), X_{AB} is obtained. When $\widehat{X}_A + \widehat{X}_B \geq 1$, X_{AB} is divided by 2 to fit in the range of $[1, 2)$ and, accordingly, a carry to $E_A + E_B$ is added to ensure the correct result.

Finally, logarithmic FP multiplication in the proposed multiplication-2 method is given by:

$$E_P = \begin{cases} E_A + E_B - bias + 1, & \widehat{X}_A + \widehat{X}_B \geq 1 \\ E_A + E_B - bias, & otherwise, \end{cases} \tag{18}$$

$$X_P = \begin{cases} 1 + \widehat{X}_A + \widehat{X}_B, & \widehat{X}_A + \widehat{X}_B < 1, \\ \widehat{X}_A + \widehat{X}_B, & 1 \leq \widehat{X}_A + \widehat{X}_B < 1.5, \\ \widehat{X}_A + \widehat{X}_B - 0.25, & 1.5 \leq \widehat{X}_A + \widehat{X}_B < 1.75, \\ \widehat{X}_A + \widehat{X}_B - 0.125, & 1.75 \leq \widehat{X}_A + \widehat{X}_B < 2. \end{cases} \tag{19}$$

C. Theoretical Error Analysis

The two proposed FP logarithmic multiplication methods introduce approximation errors in both the logarithm conversion and anti-logarithm conversion. The error distance is analyzed with the exponent ignored for simplicity.

According to (5)-(8), the result of exact multiplication is given by $P_{ex} = (1+x_A)\times(1+x_B) = 1+x_A+x_B+x_Ax_B$. The products for the proposed multiplication-1 and multiplication-2 algorithms, denoted as P_{ap1} and P_{ap2}, are given by:

$$P_{ap1}\begin{cases} = 1 + x_A + x_B, & x_A, x_B < 0.5, & \text{(20a)} \\ 1 + 2x_A + x_B, & x_A < 0.5, x_B \geq 0.5, & \text{(20b)} \\ 1 + x_A + 2x_B, & x_A \geq 0.5, x_B < 0.5, & \text{(20c)} \\ 2x_A + 2x_B, & x_A, x_B \geq 0.5, & \text{(20d)} \end{cases}$$

$$P_{ap2}\begin{cases} = 1 + x_A + x_B, & x_A, x_B < 0.5, & \text{(21a)} \\ \dfrac{3 + 2x_A + x_B}{2} \text{ or } 1 + 2x_A + x_B, & x_A < 0.5, x_B \geq 0.5, & \text{(21b)} \\ \dfrac{3 + x_A + 2x_B}{2} \text{ or } 1 + x_A + 2x_B, & x_A \geq 0.5, x_B < 0.5, & \text{(21c)} \\ 1.5 + x_A + x_B \text{ or } 1.75 + x_A + x_B, & x_A, x_B \geq 0.5. & \text{(21d)} \end{cases}$$

By comparing the equations under different conditions, the error, $Err = P_{ex} - P_{ap}$, can be derived as follows:

$$Err_1 = \begin{cases} x_Ax_B, & x_A, x_B < 0.5, & \text{(22a)} \\ x_A(x_B - 1), & x_A < 0.5, x_B \geq 0.5, & \text{(22b)} \\ x_B(x_A - 1), & x_A \geq 0.5, x_B < 0.5, & \text{(22c)} \\ (1 - x_A)(1 - x_B), & x_A, x_B \geq 0.5, & \text{(22d)} \end{cases}$$

$$Err_2 = \begin{cases} x_Ax_B, & x_A, x_B < 0.5, & \text{(23a)} \\ \dfrac{x_B + 2x_Ax_B - 1}{2} \text{ or } x_A(x_B - 1), & x_A < 0.5, x_B \geq 0.5, & \text{(23b)} \\ \dfrac{x_A + 2x_Ax_B - 1}{2} \text{ or } x_B(x_A - 1), & x_A \geq 0.5, x_B < 0.5, & \text{(23c)} \\ x_Ax_B - 0.5 \text{ or } x_Ax_B - 0.75, & x_A, x_B \geq 0.5. & \text{(23d)} \end{cases}$$

The largest positive error Err_1 is 0.25 when both x_A and x_B are 0.5, while the negative error Err_1 that has the maximum absolute value is close to -0.25 when either one of x_A or x_B approaches 0.5 while the other one equals 0.5. The largest positive Err_2 is very close to 0.5. The negative Err_2 has the largest absolute value of -0.5 when both x_A and x_B are 0.5. Therefore, the maximum $|Err|$ is 0.25 and 0.5 for multiplication-1 and multiplication-2, respectively. In comparison, the LAM [14], which uses Mitchell's approximation method has the largest absolute error of 0.25 and always underestimates the product. However, the average errors for multiplication-1 and multiplication-2 can be reduced since the positive errors and negative errors tend to cancel each other in sum-of-products calculations.

IV. Circuit Designs of Floating-Point Logarithmic Multipliers

In this section, a generic circuit architecture is first introduced for the proposed FP LMs. The circuit design for these multipliers is then presented in detail. Lastly, the radix-4 logarithm (R4L) is considered to further reduce the hardware cost of the proposed designs.

A. A Generic Circuit Architecture

Fig. 4 presents the generic circuit architecture for the proposed FP LMs according to their mathematical formulations. Taking advantage of the IEEE 754 FP format, the sign S, the exponent E, and the mantissa M of the FP number can be obtained directly. The w-bit E is given by $E[w-1]E[w-2]\cdots E[0]$. $1.M$ is used to denote the actual mantissa. Here, M contains q bits, e.g., 23 bits for single-precision, after the binary point and the hidden '1', given as $M[q-1]M[q-2]\cdots M[1]M[0]$. Note that $1.M$ represents $1+x$ in (4).

The sign bit of the product, S_P, is obtained using an XOR gate for the two input signs, S_A and S_B. For the multiplier that uses approximation method-1, the exponents (E_A and E_B) and the mantissas ($1.M_A$ and $1.M_B$) are converted into the nearest-one FP representation, whereas for approximation method-2, the conversion is not required. The converted exponents or original exponents serve as the inputs of the adder, denoted by E'_A and E'_B, and the sum is denoted by E'_P. The converted mantissas are propagated to the logarithm approximation block to compute the approximate logarithms, denoted as M'_A and M'_B. Since the multiplication is replaced by the addition operation in the logarithm domain, M'_A and M'_B are added to obtain the sum, denoted by M'_P, which is then used in the anti-logarithm approximation block to compute its anti-logarithm. In the following adjustment block, the approximate anti-logarithm will be normalized to the range of $[1, 2)$ if needed, and E'_P will be adjusted accordingly, resulting in E_P and M_P. E_P is added with the bias to comply with the IEEE standard and any exception (such as overflow, underflow, and "not a number") is reported. The two inputs are checked for exceptions at the beginning of the computation. Note that the rounding unit is not required in the inexact design since it is already inherently imprecise.

For the proposed multiplier designs, instead of being implemented directly according to the equations, the circuits are simplified to reduce the hardware complexity and latency. Specifically, some arithmetic operations are replaced with simpler operations. By merging multiple computations, the circuit blocks in the same color, as shown in Fig. 4, are integrated into a single block. In particular, the representation conversion blocks of the exponent and the mantissa are shown in dashed boxes since the conversion is not always required.

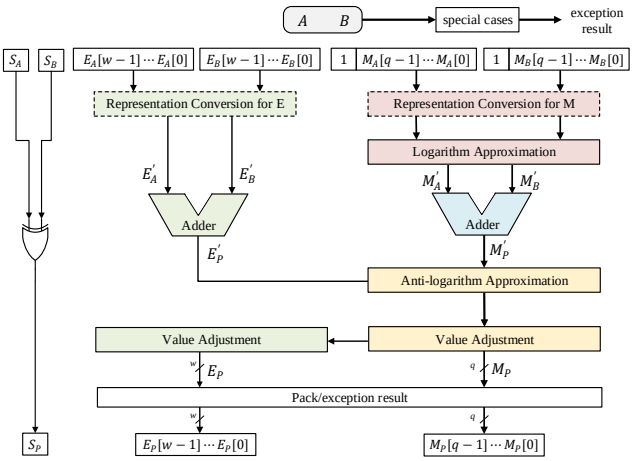


Fig. 4: A generic circuit architecture for the proposed designs.

B. FP Logarithmic Multiplier-1

1) Logarithm approximation and the addition of approximate logarithms: As shown in Fig. 5, two given mantissas, M_A and M_B are the inputs for the two FP logarithm estimators (FP-LEs), which compute the approximate logarithm of the mantissa. For FPLM-1, both the representation conversion and logarithm approximation for the mantissa are implemented by an FP-LE. Simple wire routing is used to implement the FP-LE, as shown in Fig. 5. The nearest power of two for each FP number can be determined by simply checking the leading bit of the explicit mantissa (without the hidden 1), $M[q-1]$, and hence a 2-to-1 multiplexer is used to obtain the approximate logarithm, M'. According to (11), the left shifter for implementing the division by 2 is replaced by wire routing. Therefore, $(1+x)/2$ is implemented as $0.1M[q-1]\cdots M[1]$. When $M[q-1]=1$, the approximate logarithm, $(1+x)/2-1$, is obtained as a negative number in 2's complement, i.e., $M' = 1.1M[q-1]\cdots M[1]$; otherwise, M' is $0.M[q-1]\cdots M[0]$. In the case of $M[q-1]=1$, the LSB of M', $M[0]$, is discarded to keep M' in $q+1$ bits.

M'_A and M'_B are then summed using a $q+1$-bit adder.

2) Anti-logarithm approximation and value adjustment: The anti-logarithm approximation and value adjustment are implemented together using a multiplexer, as shown in Fig. 5. The $(q+1)$-bit M'_P obtained from the adder is the input for this block. For FPLM-1, as per (15), since $\widehat{X}_A+\widehat{X}_B$ can be negative in some cases, the MSB of M'_P, i.e., $M'_P[q]$ is the sign bit of M'_P in 2's complement. When $\widehat{X}_A+\widehat{X}_B<0$, $M'_P[q]=1$; otherwise, $M'_P[q]=0$. Therefore, $M'_P[q]$ is used as the selection signal for the multiplexer. $1+\widehat{X}_A+\widehat{X}_B$ is implemented as $0.M'_P[q-1]\cdots M'_P[0]$ or $1.M'_P[q-1]\cdots M'_P[0]$ when $M'_P[q]=1$ or $M'_P[q]=0$, respectively. According to (15), $-0.5\le\widehat{X}_A+\widehat{X}_B<1$, meaning that $M'_P[q-1]=1$ when $M'_P[q]=1$. Therefore, in the case of $M'_P[q]=1$, $1.M'_P[q-2]\cdots M'_P[0]0$ is obtained by performing the $\times 2$ operation on $0.M'_P[q-1]\cdots M'_P[0]$.

3) Addition of exponents and value adjustment: The circuits for the exponent conversion, addition and value adjustment, shown as the green blocks in Fig. 4, are implemented together by integrating and simplifying multiple computations. For FPLM-1, according to (9), the exponents of the two operands are converted first, depending on the values of $M_A[q-1]$ and $M_B[q-1]$. Then the converted exponents are added subsequently to obtain E'_P, which is decremented by 1 if $M'_P[q]=1$, as per (14). The required circuits to implement these operations are simplified to one adder with a carry-in bit, $Carry\_E$, that is determined by the modified value of E_A and E_B. When $M_A[q-1]M_B[q-1]$ are '00' or '11', $M'_P[q]$ can only be '0' or '1', respectively; otherwise, $M'_P[q]$ can be '0' or '1' in either case. According to (11) and (14), when both mantissa values are smaller than 0.5 ($M_A[q-1]M_B[q-1]=00$), $\widehat{X}_A+\widehat{X}_B\ge 0$ ($M'_P[q]=0$), which means $E'_A+E'_B$ is not modified; hence, $Carry\_E=0$. When both mantissas are greater than or equal to 0.5 ($M_A[q-1]M_B[q-1]=11$), $\widehat{X}_A+\widehat{X}_B<0$ ($M'_P[q]=1$), which means that $E'_A+E'_B$ is decremented by 1, i.e., $E_A+1+E_B+1-1=E_A+E_B+1$; therefore, $Carry\_E=1$.

Therefore, the $Carry\_E$ for FPLM-1 is obtained as:

$$Carry\_E = \frac{}{(\,M'_P[q]+\overline{(\,M_A[q-1]+M_B[q-1]\,)}\,)\cdot\overline{(\,M_A[q-1]\cdot M_B[q-1]\,)}} \quad (24)$$

As shown in Fig. 5, four logic gates are used to generate $Carry\_E$.

C. FP Logarithmic Multiplier-2

1) Logarithm approximation and addition of approximate logarithms: As shown in Fig. 6, for FPLM-2, the FP-LE implements the logarithm approximation for the mantissa. As per (16), $M[q-1]$ is used to determine the boundary point for computing the approximate logarithm, M'. When $M[q-1]=1$, $(1+x)/2$ is implemented as $0.1M[q-1]\cdots M[1]$; otherwise, x is obtained as $0.M[q-1]\cdots M[0]$. In both cases, the obtained hidden bit is 0, which has no effect on the sum. Thus, M' is obtained using M_A and M_B as the inputs of FP-LE without prefixing the hidden bit. Accordingly, a q-bit adder is used for the addition of M'_A and M'_B with a carry-out signal, C_{out}, as shown in Fig. 6. In the case of $M[q-1]=1$, the LSB of M', $M[0]$, is discarded to keep M' as q bits.

2) Anti-logarithm approximation and value adjustment: The q-bit M'_P and C_{out} obtained from the adder are used to implement the anti-logarithm approximation and value adjustment, as shown in Fig. 6. $\widehat{X}_A+\widehat{X}_B$ in (19) is obtained as $C_{out}.M'_P[q-1]\cdots M'_P[0]$. The four boundary points in (19) are implemented by using C_{out}, $M'_P[q-1]$ and $M'_P[q-2]$, which are then used to implement the select

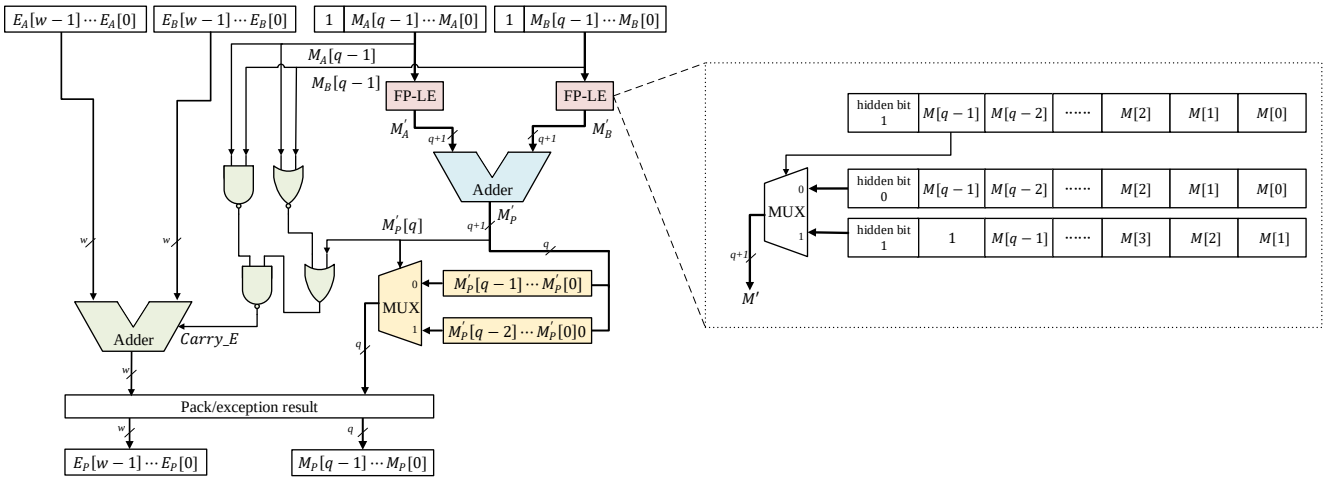


Fig. 5: The circuit design of the first proposed floating-point logarithmic multiplier, FPLM-1.

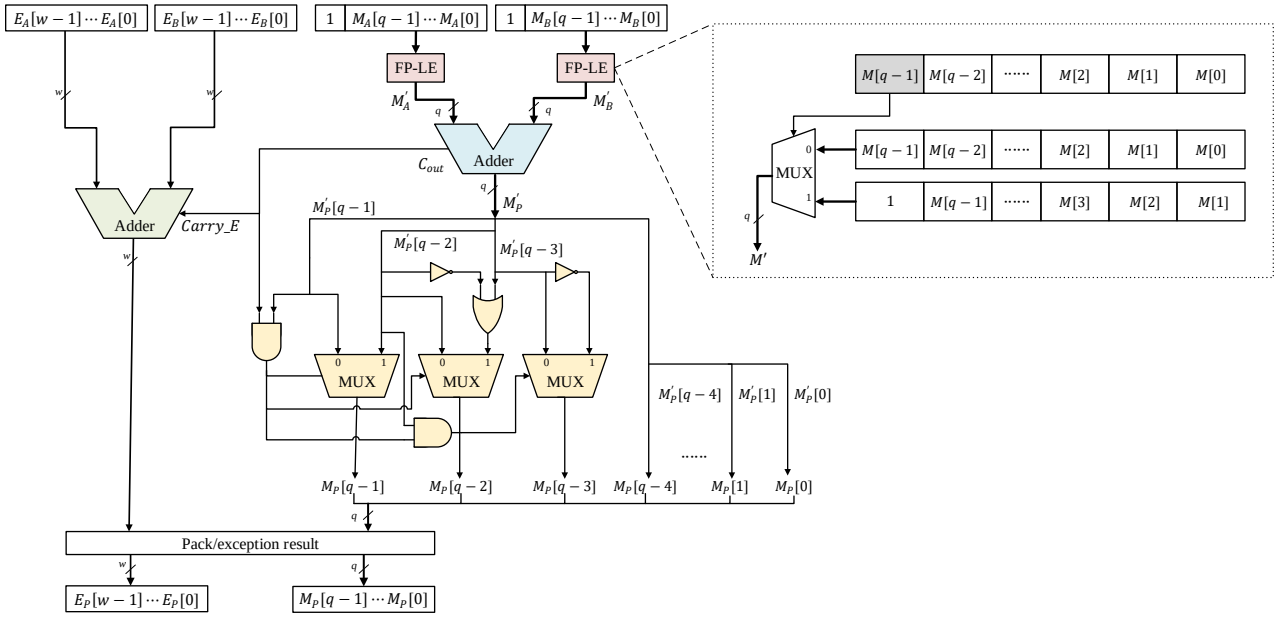


Fig. 6: The circuit design of the second proposed floating-point logarithmic multiplier, FPLM-2.

signals for the three multiplexers. As per (19), the first sub-function is determined by $C_{out} = 0$, and the mantissa, $1 + \widehat{X}_A + \widehat{X}_B$, is then obtained as $1.M'_P[q-1]\cdots M'_P[0]$. When $1 \leq \widehat{X}_A + \widehat{X}_B < 1.5$, meaning that $C_{out}M'_P[q-1] = 10$, $\widehat{X}_A + \widehat{X}_B$ is also obtained as $1.M'_P[q-1]\cdots M'_P[0]$. In these two cases, $C_{out} \cdot M'_P[q-1]$, which is always 0, is used as the select signal to pass M'_P to the output. The third sub-function is determined by $C_{out}M'_P[q-1]M'_P[q-2] = 110$, and $\widehat{X}_A + \widehat{X}_B - 0.25$ is implemented by $1.01M'_P[q-3]\cdots M'_P[0]$. The fourth scenario occurs when $C_{out}M'_P[q-1]M'_P[q-2] = 111$. $\widehat{X}_A + \widehat{X}_B - 0.125$ is obtained as $1.101M'_P[q-4]\cdots M'_P[0]$ if $M'_P[q-3]$ is 0 or $1.110M'_P[q-4]\cdots M'_P[0]$ if $M'_P[q-3]$ is 1. In these two cases, $C_{out} \cdot M'_P[q-1] = 1$ selects the output for $M_P[q-1]M_P[q-2]$ and $C_{out} \cdot M'_P[q-1] \cdot M'_P[q-2]$ is used to select the output for $M_P[q-3]$. The logic gates for computing $M_P[q-1]\cdots M_P[q-2]$ are shown in Fig. 6 and $M'_P[q-4]\cdots M'_P[0]$ is directly passed to the output as the remaining bits.

3) Addition of exponents and value adjustment: For FPLM-2, according to (18), E_A and E_B are summed and increased by 1 if $C_{out} = 1$. Therefore, as shown in Fig. 6, C_{out} is directly used to generate $Carry\_E$.

D. Designs using the Radix-4 Logarithm

The radix-4 logarithm (R4L) was introduced to reduce the hardware complexity of the FP logarithm multipliers by exploiting the relation between the base 2 and base 4 logarithms. The base 2 logarithm of a given number N, is two times of its base 4 logarithm, i.e., $\log_2 N = 2\log_4 N$. This indicates that the base 4 logarithm can be stored with a 1-bit smaller bit-width than the base 2 logarithm in a hardware implementation. Therefore, we propose to convert the approximate logarithm

of the mantissa to a radix-4 logarithm in an intermediate computation process to reduce the hardware cost of an FP LM. Radix-4 logarithm conversion is only used for the mantissa since applying to the exponent leads to large errors.

The R4L approach is specified in Algorithm 1. The approximate logarithms for the two mantissas, denoted by m_A and m_B, are obtained by applying the logarithm approximation to the mantissas. Note that depending on the requirement of $\alpha()$, the mantissa here can be prefixed with the hidden '1'. m_A and m_B are converted to the radix-4 logarithm by simply discarding the LSB, resulting in $m_A[q:1]$ and $m_B[q:1]$. The bit-width of the resulting approximate logarithm is reduced, which leads to a smaller hardware cost for the adder. Then the result, sum, is concatenated with '0' to ensure the correctness of the anti-logarithm approximation. When q is 1, the addition can be implemented with just one OR gate. The output, M_P, is obtained after normalizing the anti-logarithm result to make it lie in the range of $[0, 1)$.

Algorithm 1 A Radix-4 Logarithm (R4L) Approach

Input: Mantissas: M_A and M_B; Bit-width of mantissa: q; Logarithm approximation method: $\alpha()$; Anti-logarithm approximation method: $\beta()$;
Output: Mantissa of the product: M_P.

if $q \geq 1$ **then**
  $m_A \leftarrow \alpha(M_A)$
  $m_B \leftarrow \alpha(M_B)$
  if $q > 1$ **then**
    $sum \leftarrow ADD(m_A[q:1],\ m_B[q:1])$
  else
    $sum \leftarrow OR(m_A[1],\ m_B[1])$
  end if
  $h \leftarrow \beta(sum\ \&\ \text{'0'})$
  $M_P \leftarrow NORM(h)$
else
  $M_P \leftarrow 0$
end if

The R4L approach was applied to the proposed FPLM-1, FPLM-2, and the conventional logarithmic multiplier (CLM) that uses Mitchell's approximation [24] to reduce their hardware complexity. The obtained multiplier designs are denoted as FPLM-1-r4, FPLM-2-r4, and CLM-r4.

V. Performance Evaluation

In this section, the performance of the five proposed multipliers (FPLM-1, FPLM-2, FPLM-1-r4, FPLM-2-r4, and CLM-r4) is evaluated by comparing them with the conventional FP multiplier (FPM), LAM [14], CFPU2 [43] and FPmul-T9 [44], with respect to accuracy and circuit efficiency. Note that FPmul-T9 is evaluated in half-precision since it is a half-precision design. The CFPU2, which has two tuning bits, is used due to its good trade-off between the accuracy and the hardware cost.

A. Accuracy Evaluation

1) Error Assessment: Two standard error metrics are considered to evaluate the error characteristics of the proposed FP LMs.

- The mean relative error distance (MRED) is the average value of all possible relative absolute error distances.
- The average error (AE) is the average difference, which can be positive or negative, between the exact and approximate products.

Four FP precision levels are considered for the evaluation of each multiplier: 32-bit single-precision, 16-bit half-precision, Brain FP (bfloat16) and 8-bit FP (FP8) format. The first three precisions are respectively in the form of (1, 8, 23), (1, 5, 10) and (1, 8, 7) bits for the sign, exponent and mantissa, respectively. The FP8 is chosen in the form of (1, 5, 2) bits since it was found to perform the best with respect to classification accuracy after the simulation of using various FP8 formats [20]. The product obtained by the single-precision exact FP multiplier is used as the benchmark since the truncation of mantissa bits inherently introduces errors for the 16-bit and 8-bit implementations. A sample of 10^7 random cases from the two most common general distributions, i.e., uniform and standard normal distributions, of input operands were generated to obtain the results in Table I. Note that the uniformly distributed random cases are generated over the interval of $[1, 2)$ to assess the errors introduced in the mantissa computation. Due to the small bit-width of the FP8 format, all the input combinations are enumerated and considered in the uniformly distributed cases.

The results in Table I show that, for single-precision, FPLM-1 is the most accurate multiplier design for both two distributions, with the lowest MRED and $|\mathrm{AE}|$, followed by FPLM-1-r4. For the bfloat16 format, FPLM-2-r4 achieves the smallest $|\mathrm{AE}|$, while FPLM-1 is the most accurate for MRED, which is 39.6% more accurate than CFPU2 for the standard normal distribution. Compared to the LAM, FPLM-1 performs up to 30.8% more accurately for the bfloat16 format with respect to MRED, and achieves a smaller $|\mathrm{AE}|$ by up to 2.58×10^3 times for single-precision. The smaller $|\mathrm{AE}|$s show the advantage of the double-sided error distribution obtained for the proposed designs. Note that for the 32-bit precision, the error metrics that are shown as identical have a difference of less than 10^{-6}. For the 16-bit precision, FPLM-1-r4 produces larger errors than FPLM-1, whereas FPLM-2-r4 is slightly more accurate than FPLM-2. This is due to the relation between its error distribution and the bit-width, which is explained in the next subsection. For FP8, all of the multipliers produce large errors. FPLM-2 is the most accurate multiplier, achieving up to a 16.1% smaller MRED and up to 45.9% smaller $|\mathrm{AE}|$ than LAM.

2) The Relation between FP Precisions and Error Behavior: According to Table I, the comparison results are the same for 32- and 16-bit precisions, while they are different for the FP8 format. Therefore, we investigated the relation between the mantissa width and the error behavior of the proposed designs. Fig. 7 shows the MREDs obtained when varying the mantissa width for FP LMs and the FPM with the single-precision. The MREDs for widths decreasing from 10 bits to 2 bits are presented due to the relatively small drop in accuracy going from 23 bits to 11 bits. The lines with the marker show the MREDs for the FPM and all FP LMs, respectively. For the FPM, by considering single-precision as the baseline format,

TABLE I: Error Metrics for the Multipliers

Multipliers	Uniform Distribution		Standard Normal Distribution	
Single-Precision				
	MRED	|AE|	MRED	|AE|($\times 10^{-5}$)
FPLM-1	**0.0288**	$\mathbf{3.2 \times 10^{-5}}$	**0.0288**	**0.81**
FPLM-2	0.0368	0.0416	0.0373	1.09
FPLM-1-r4	0.0288	3.2×10^{-5}	0.0288	0.83
FPLM-2-r4	0.0368	0.0416	0.0373	1.09
CLM-r4	0.0384	0.0833	0.0381	1.24
LAM	0.0384	0.0833	0.0381	1.24
CFPU2	0.0439	2.4×10^{-5}	0.0468	2.28
Half-Precision				
	MRED	|AE|	MRED	|AE|($\times 10^{-5}$)
FPLM-1	0.0289	0.0021	0.0298	0.82
FPLM-2	0.0365	0.0399	0.0394	1.10
FPLM-1-r4	0.0290	0.0043	0.0311	0.83
FPLM-2-r4	0.0362	0.0382	0.0392	1.10
CLM-r4	0.0397	0.0862	0.0416	1.27
LAM	0.0391	0.0847	0.0409	1.26
CFPU2	0.0442	0.0032	0.0484	2.29
FPmul-T9	**0.002**	**0.0002**	**0.0012**	**0.05**
Bfloat16				
	MRED	|AE|	MRED	|AE|($\times 10^{-5}$)
FPLM-1	**0.0302**	0.0175	**0.0300**	0.74
FPLM-2	0.0348	0.0280	0.0361	0.98
FPLM-1-r4	0.0330	0.0351	0.0326	1.16
FPLM-2-r4	0.0341	**0.0143**	0.0359	**0.73**
CLM-r4	0.0488	0.1066	0.0485	1.45
LAM	0.0436	0.0950	0.0433	1.32
CFPU2	0.0465	0.0181	0.0497	2.30
FP8				
	MRED	|AE|	MRED	|AE|($\times 10^{-5}$)
FPLM-1	0.2311	0.5626	0.2159	4.83
FPLM-2	**0.1626**	**0.3750**	**0.1586**	**2.61**
FPLM-1-r4	0.4367	1.0000	0.4232	8.69
FPLM-2-r4	0.3201	0.7500	0.3078	5.74
CLM-r4	0.3201	0.7500	0.3078	5.74
LAM	0.1914	0.4375	0.1891	3.71
CFPU2	0.1652	0.3795	0.1598	3.54

the reduction of the bit-width leads to accuracy loss in the product. As shown in Fig. 7, the FPM produces insignificant errors when the mantissa width is from 10 bits to 7 bits, whereas its MRED greatly increases when the mantissa width is smaller than 7 bits. The MRED rapidly increases for all the LMs with further decreases in the mantissa width from 6 bits to 2 bits. The FPLM-1-r4 has the largest error increase and the FPLM-2 becomes the most accurate, which produces smaller errors than the FPM when the mantissa width is lower than 5 bits.

The average multiplication results of 10^4 uniformly distributed random numbers within $[1, 2)$ are plotted with respect to the reduced mantissa widths in Fig. 8 for 7 bits and 4 bits to 2 bits (chosen to fit space limits). To clarify the trends, the average values of products for each of the 50 samples were computed after arranging the products in ascending order. As shown in Fig. 8, with decreasing mantissa width, the FPM generates increasingly smaller products (see the blue plus sign) compared to the single-precision baseline results (see the black dots). Meanwhile, since the LAM always produces smaller products (see the lime green diamonds) than the FPM, the error distances between the LAM and the baseline result are increased with decreasing mantissa width. However, multipliers FPLM-1 (see the green circles) and FPLM-2 (see the orange squares) can produce both underestimated and overestimated products.

When the mantissa width is lower than 7-bit, the overestimated products have smaller error distances to the baseline products compared to the underestimated products, which eventually reduces the overall error distances. This trend is prominent especially when the mantissa width is from 4-bit to 2-bit, as shown in Fig. 8(b) to Fig. 8(d), where the overestimated products are closer to the baseline products. FPLM-2 is more accurate when the mantissa width is from 5-bit to 2-bit since it produces a larger number of larger overestimated products than FPLM-1.

FPLM-1-r4, FPLM-2-r4, and CLM-r4 show similar error behavior and smaller products compared to FPLM-1, FPLM-2, and LAM, respectively. In particular, when the mantissa width is larger than 6-bit, FPLM-2-r4 is more accurate than FPLM-2. Compared to FPLM-2, for the same inputs, FPLM-2-r4 produces a smaller underestimated product or a smaller overestimated product, as shown in Fig. 8(a). The overestimated products computed by FPLM-2-r4 have smaller error distances to the baseline products than those computed by FPLM-2 and a larger number of them are produced than the underestimated products, therefore reducing the overall error distances of FPLM-2-r4. However, when the mantissa width is smaller than 7-bit, the error caused by the width truncation becomes dominant, leading to a larger number of underestimated products compared to the baseline results. When the mantissa width is 2-bit, a significant error increase for FPLM-1 occurs. The limited mantissa width causes additional errors, which increases the approximation error and leads to large underestimated errors when the input mantissa is larger than 1.5. As shown in Fig. 8(d), the products of FPLM-1 have a larger error distance to the baseline product for the sample sets from 160 to 200.

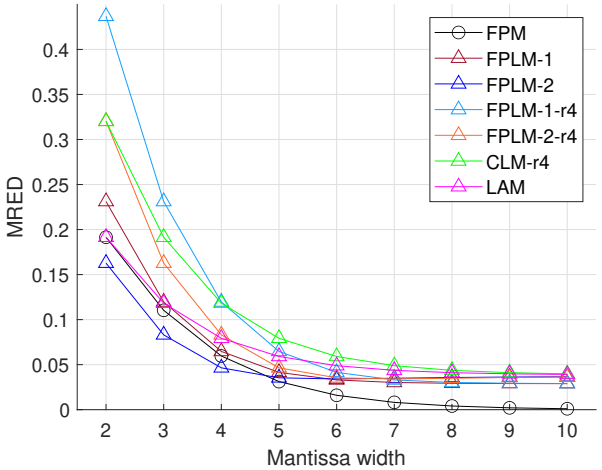
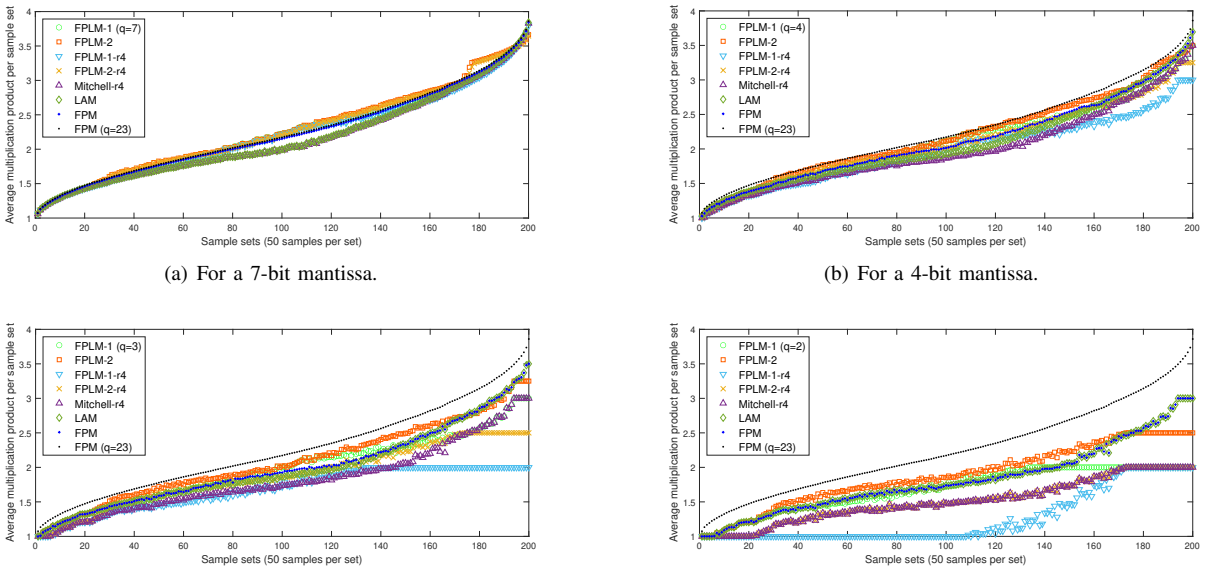


Fig. 7: MREDs of the multipliers as a function of the mantissa width, with respect to the single-precision conventional FP multiplier.

B. Hardware Evaluation

The five proposed FPLMs and the comparative designs, i.e., LAM [14], CFPU2 [43] and FPmul-T9 [44], were implemented in Verilog and an FPM was obtained using the



(a) For a 7-bit mantissa.

(b) For a 4-bit mantissa.

(c) For a 3-bit mantissa.

(d) For a 2-bit mantissa.

Fig. 8: Average product of each sample set (50 samples per set) for the five proposed multipliers, LAM [14], and FPM with reduced mantissa widths (7-, 4-, 3-, and 2-bit) compared to the single-precision FPM (with a mantissa width of 23 bits).

Synopsys DesignWare IP library (DW_fp_mult). All of the designs were synthesized using the Synopsys Design Compiler (DC) for STM's CMOS 28-nm process assuming a supply voltage of $1.0\,V$ and a die temperature of $25°C$. The "ultra compile" option was used in the synthesis for optimization. All of the designs in the four precision formats were evaluated at a clock frequency of 250 MHz. The critical path delay, area and power dissipation are reported by the Synopsys DC. Note that the evaluation does not include layout parasitics.

As shown in Table II, for the 32-bit and 16-bit implementations, the proposed CLM-r4 is the most energy-efficient and smallest design. It incurs a $68\times$ smaller power-delay product (PDP) and a $18\times$ smaller area compared to the FPM for the single-precision implementation. The other four proposed designs are more hardware-efficient and more accurate than CFPU2; they have a larger hardware cost compared to the LAM while being more accurate. The proposed designs are not as accurate as the FPmul-T9 but with $6\times$ to $13\times$ energy savings. For the FP8 format, FPLM-1-r4 achieves a 19% smaller PDP and a 6.8% smaller area compared to LAM. The FPLM-2 requires slightly smaller power and area compared to LAM while being 16% more accurate. FPLM-2-r4 and CLM-r4 are also more energy-efficient and smaller than LAM. It is important to note that, according to [40], the power of the accurate FP multiplier is dominated by the mantissa multiplication, accounting for over 80%, and the rounding unit for nearly 18%. Therefore, the reduction in power and area can be largely attributed to the elimination of the mantissa multiplier and the rounding unit in the proposed designs.

TABLE II: Circuit Measurements of the FP Multipliers

	Power (μW)	Area (μm^2)	Delay (ns)	PDP (fJ)
Single-Precision				
FPM	643.4	2666	3.54	2277.6
FPLM-1	30.8	240.5	2.36	72.8
FPLM-2	25.8	211.9	2.20	56.9
FPLM-1-r4	29.9	234.1	2.27	67.9
FPLM-2-r4	22.8	198.6	2.11	48.2
CLM-r4	**17.3**	**146.7**	**1.92**	**33.2**
LAM	17.7	149.3	1.98	35.0
CFPU2	372.7	1888	3.56	1326.8
Half-Precision				
FPM	157.5	868.8	3.03	477.2
FPLM-1	15.1	119.9	1.25	18.9
FPLM-2	13.4	108.3	1.05	14.1
FPLM-1-r4	14.2	113.4	1.18	16.7
FPLM-2-r4	12.7	103.1	0.98	12.4
CLM-r4	**9.5**	**78.6**	**0.91**	**8.7**
LAM	9.9	81.2	0.97	9.6
FPmul-T9	58.7	333.1	1.98	116.2
CFPU2	83.8	540.6	3.19	267.3
Bfloat16				
FPM	107.8	724.6	2.90	312.6
FPLM-1	15.0	123.3	1.23	18.5
FPLM-2	14.1	115.7	1.05	14.8
FPLM-1-r4	14.1	116.8	1.16	16.3
FPLM-2-r4	13.4	110.4	0.98	13.1
CLM-r4	**11.2**	**93.8**	**0.91**	**10.2**
LAM	11.6	96.4	0.97	11.2
CFPU2	45.2	404.5	2.59	117.1
FP8				
FPM	41.0	327.5	2.10	86.14
FPLM-1	7.25	55.1	0.49	3.55
FPLM-2	7.14	54.1	0.48	3.42
FPLM-1-r4	**6.73**	**50.9**	**0.40**	**2.69**
FPLM-2-r4	6.81	52.0	0.40	2.72
CLM-r4	6.81	52.0	0.40	2.72
LAM	7.23	54.6	0.46	3.32
CFPU2	12.2	128.2	1.13	13.78

VI. Application Evaluations

A. JPEG Compression

1) Experimental Setup: The five proposed FP LMs were evaluated using JPEG compression for the four precision lev-

els. The required matrix multiplications in the discrete cosine transform (DCT) and inverse DCT (IDCT) were implemented using the FP LMs. The quality of the compressed image was assessed by comparing it with the original image using the peak signal noise ratio (PSNR). The JPEG compression of 256×256-pixel standard test images including "Lena" and "cameraman" were performed with the standard quantization matrix. The compression quality factor was set to 50 in the experiment.

2) Evaluation Results: The average PSNR values obtained with the two test images, using different FP multipliers for the four precision levels, are shown in Table III.

Among the FP multipliers, FPLM-1 achieves the highest PSNR for the 32-bit and 16-bit precisions; FPLM-2 performs the best for the FP8 format, followed by FPLM-1. For 32-bit and 16-bit precisions, images processed by FPLM-1, FPLM-2, FPLM-1-r4, and FPLM-2-r4 show similar qualities as the accurate result, while images compressed using the LAM and CLM-r4 show significant losses in quality. This suggests that the double-sided error distributions produced by FPLM-1, FPLM-2, FPLM-1-r4, and FPLM-2-r4 can effectively reduce error accumulation in the multiple matrix multiplications of both the DCT and IDCT. For the FP8 format, all reconstructed images show considerably lower quality due to the relatively large errors produced by using FP multipliers. Overall, compared with LAM, the proposed FPLM-1, FPLM-2, FPLM-1-r4, and FPLM-2-r4 multipliers produce higher image qualities with a larger PSNR by up to 4.7 dB; CLM-r4 offers higher energy and area savings with a very close image quality. It is also shown that the obtained PSNRs follow the error analysis results for the FP multipliers.

TABLE III: Average PSNR (dB) for JPEG Compression using FP Multipliers with Different Precisions

	Single-Precision	Half-Precision	Bfloat16	FP8
FPM	32.27	32.26	32.05	15.26
FPLM-1	**30.10**	**30.08**	**29.67**	15.23
FPLM-2	30.01	30.01	29.61	**15.31**
FPLM-1-r4	30.10	30.07	29.20	13.40
FPLM-2-r4	30.01	30.00	29.12	13.63
CLM-r4	24.91	24.85	23.87	13.63
LAM	24.91	24.86	24.32	15.26
CFPU2	28.67	28.64	28.11	14.98

B. Neural Network Applications

1) Experimental Setup: The FP multipliers were used in the training phase of a multi-layer perceptron (MLP) and a convolutional neural networks (CNN) by using the Pytorch framework [45]. The same multiplier is used both in the training phase and in the inference engine. It is noted that the training process is affected by many factors, which are explored for determining the training procedure for our experiments as discussed below.

a) Datasets and Network Models: Four classification datasets, fourclass [46], HARS [47], MNIST [48] and CIFAR-10 [49], were used for the evaluation. A small MLP network was used in training for the fourclass dataset. The MLP networks used for the HARS and MNIST are (561, 40, 6) and (784, 128, 10) models, respectively. AlexNet was used to classify the CIFAR-10 dataset.

b) Hyperparameter Configurations: To fairly evaluate the effect of approximate multiplication on training, for each dataset the same training procedure (e.g., the same optimizer) is used for all the models using different FP multipliers. In all cases, the mini-batch training strategy [50] is used; the batch size for the CIFAR-10, MNIST, HARS and fourclass are set to 64, 128, 300 and 100, respectively. In addition, five trials were done using different random initialization for each training simulation.

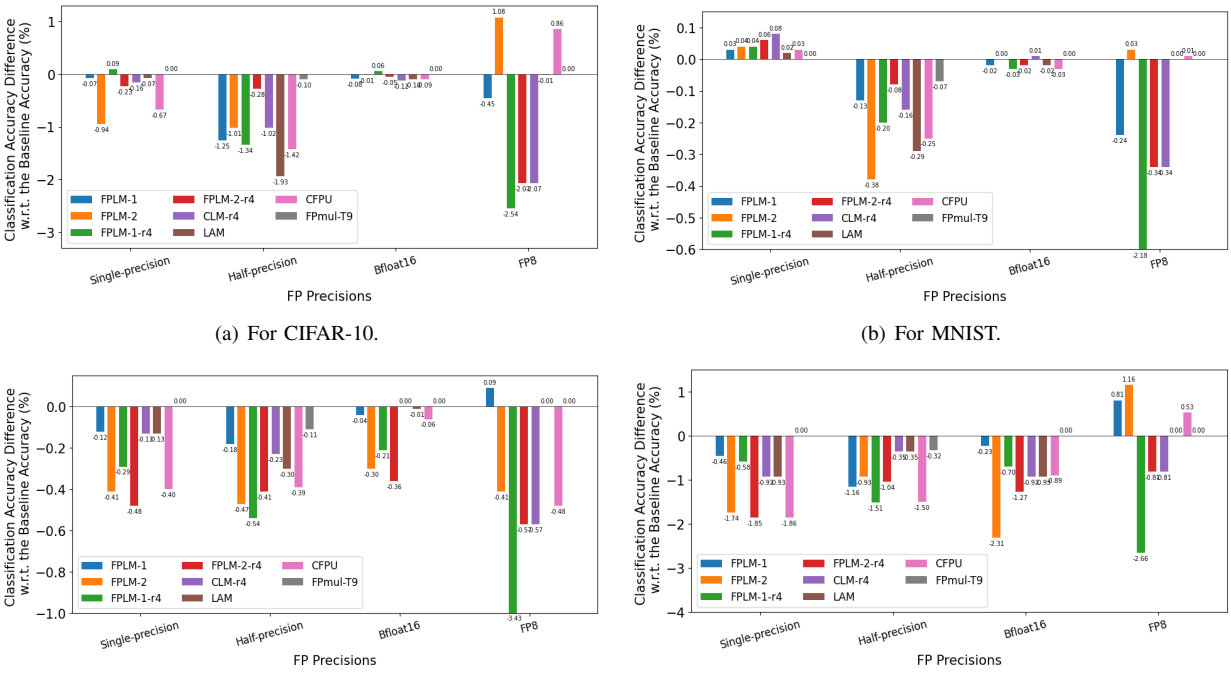
c) Early Stopping Criterion: The stopping criterion has an important impact on obtaining the trained parameters, i.e., weights and biases, and thereby it affects the classification accuracy [51]. In order to obtain optimal generalization performance and to avoid overfitting, an early stopping strategy terminates the training if the validation loss does not improve for a number of (t) epochs or the set maximum number of epochs is reached. t is set to 100, 100, 200, and 1000 for CIFAR-10, MNIST, HARS, and fourclass, respectively. By doing so, a desirable trade-off is achieved between the training time and validation loss. The trained parameters are then obtained at the epoch that achieves the lowest validation loss. The maximum number of training epochs is set to 30k; for each epoch, the entire training set is shuffled.

2) Evaluation Results:

a) Classification Accuracy Analysis: The comparison of the average classification accuracies of five trials for using different FP multipliers for the four precisions is shown in Fig. 9. Note that for a better visualization, the percentage shown in Fig. 9 is the difference between the accuracy of using an approximate multiplier and that of using FPM at the same precision level.

The classification accuracy fluctuates for each specific multiplier regarding different datasets, FP precision formats, and random initialization. However, the differences are subtle, especially for 32-bit and 16-bit precisions. For each trial, the difference in classification accuracy is $< 1\%$ in most cases for these multipliers. Considering the average results, as shown in Fig. 9, the difference is even smaller ($< 0.4\%$) for MNIST and HARS. This average difference is slightly larger for CIFAR-10 ($< 2\%$) and fourclass ($< 2.4\%$). For the FP8 format, an average accuracy loss of 3% is observed by using the FPM compared to the using FPM in 32-bit precision. The classification accuracy of using FPLM-1-r4 degrades the most across all the datasets ($> 2\%$) since FPLM-1-r4 has a large MRED and AE in the FP8 format.

For the CIFAR-10 and MNIST datasets, FPLM-1-r4 and CLM-r4 achieve the highest average classification accuracies in the single-precision and bfloat16 formats, respectively; FPmul-T9 performs the best in the half-precision, followed by the FPLM-2-r4. It is interesting to observe that all of the multipliers slightly improve the classification accuracy in many cases. For HARS and fourclass, the FPLM-1 is the best in the single-precision. For the FP8 format, the FPLM-2 outperforms the other multipliers for CIFAR-10, MNIST and fourclass while improving the classification accuracy; FPLM-1 achieves the best for HARS with an improved accuracy.



(a) For CIFAR-10.

(b) For MNIST.

(c) For HARS.

(d) For fourclass.

Fig. 9: Comparison of the average classification accuracy of four datasets with approximate multipliers: a negative percentage means a decrease and a positive percentage means an increase in the accuracy with respect to using FPM.

Overall, the evaluation result indicates that NNs using the five proposed multipliers and the other FP multipliers can lead to very close classification accuracies for CIFAR-10, MNIST and HARS with the 32-bit and 16-bit precision formats, while the accuracies of classifying a small dataset, i.e., fourclass, or using the FP8 format have slightly larger differences. Compared to LAM and CFPU2, FPLM-1 performs better in most cases and FPLM-2 is better in the FP8 format; in the cases that other proposed designs achieve a lower accuracy, the energy reduction is the trade-off. Our designs achieve better hardware efficiency at the cost of a larger accuracy loss in half-precision compared to FPmul-T9. The average difference for the half-precision and FP8 format is larger than the other two precisions due to the smaller bit-width of the exponent.

The classification accuracy is not strictly consistent with the error analysis results for the FP multipliers. These results indicate that a smaller multiplication error does not always lead to a higher classification accuracy. However, for the FP8 format, the performance of using different multipliers loosely follows the error analysis results for the multipliers in Table I. This is possible since the iterative computation in the training process, including both forward and backward propagation, depends on many different factors. These factors vary for each dataset, network structure and random initialization.

b) Hardware Evaluation: Since the proposed designs perform well for classification with low energy in the bfloat16 format, an artificial neuron in the bfloat16 format was implemented to assess the overall hardware cost of NNs. The FP adder used in the neuron was obtained using the Synopsys DesignWare IP library (DW_fp_add).

The simulation results in Table IV were obtained at a clock frequency of 125 MHz. It was shown that a neuron using the proposed CLM-r4 dissipates the least energy with the smallest area, which is up to $4.2\times$ smaller in energy with an area up to $2.2\times$ smaller than the neuron using FPMs.

TABLE IV: Circuit Assessment of the Artificial Neuron

	Power(μW)	Area(μm^2)	Delay(ns)	PDP(fJ)
FPM	144.2	1075.1	6.5	941.6
FPLM-1	59.1	537.1	4.4	263.3
FPLM-2	57.9	522.4	4.2	247.3
FPLM-1-r4	57.6	526.3	4.3	252.6
FPLM-2-r4	56.9	514.8	4.2	239.6
CLM-r4	**54.1**	**483.23**	**4.1**	**223.7**
LAM	54.8	486.1	4.2	230.2
CFPU2	78.6	1052.8	6.1	479.4

VII. Conclusions

In this article, FP logarithmic multipliers are proposed for error-tolerant computation-intensive applications by using two novel approximation methods that benefit from double-sided error distributions in the logarithm and anti-logarithm conversions. The radix-4 logarithm is used to further reduce the hardware complexity. For the 32-bit and 16-bit precisions, the proposed FPLM-1 multiplier is the most accurate design with up to 30.8% smaller MRED and $10^3\times$ smaller average error compared to a recent FP design, LAM [14]. CLM-r4 is the most energy-efficient multiplier and the smallest design with up to $68\times$ smaller PDP and up to $18\times$ smaller area compared to the conventional FP multiplier. For the FP8 format, FPLM-1-r4 is the most energy-efficient and the smallest design. From the experimental results, the double-sided error distribution benefits from error cancellation in low-precision computations.

Using the proposed FP logarithmic multipliers in JPEG image compression achieves higher image quality than LAM, with a larger PSNR of up to 4.7 dB. Compared to NNs implemented using a conventional FP multiplier, the evaluation for benchmark NNs using proposed designs shows up to $4.2\times$ and $2.2\times$ reductions in energy and area, respectively, while achieving similar classification accuracies. Particularly, the proposed FPLM-1 achieves higher classification accuracy in most cases and FPLM-2 performs better in FP8 format, compared to LAM. Moreover, higher classification accuracies can be obtained by using the proposed designs compared to the use of the conventional FP multiplier. We also found that no single FP logarithmic multiplier performed the best across all the datasets with different precisions. Therefore, the impact of FP logarithmic multipliers on the training process of NNs remains an important topic for future research.

ACKNOWLEDGMENT

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326) and Alberta Innovates (Project Number: RES0053965). T. Zhang is supported by a Ph.D. scholarship from the China Scholarship Council (CSC). We also thank CMC Microsystems for providing access to the simulation tools.

REFERENCES

[1] M. Neisser, "International roadmap for devices and systems lithography roadmap," *Journal of Micro/Nanopatterning, Materials, and Metrology*, vol. 20, no. 4, pp. 044 601–044 601, 2021.

[2] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[3] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019.

[4] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," *Proceedings of the 50th Annual Design Automation Conference*, pp. 1–9, 2013.

[5] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," *ETS*, pp. 1–6, 2013.

[6] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proc. IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[7] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," *Proc. 32nd Int. Conf. on Machine Learning*, vol. 37, pp. 1737–1746, 2015.

[8] M. Yan, Y. Song, Y. Feng, G. Pasandi, M. Pedram, and S. Nazarian, "knn-cam: A k-nearest neighbors-based configurable approximate floating point multiplier," *ISQED*, pp. 1–7, 2019.

[9] D. Peroni, M. Imani, and T. S. Rosing, "Runtime efficiency-accuracy tradeoff using configurable floating point multiplier," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 2, pp. 346–358, 2020.

[10] M. Imani, R. Garcia, S. Gupta, and T. Rosing, "Rmac: Runtime configurable floating point multiplier for approximate computing," *Proc. Int. Symposium on Low Power Electronics and Design*, pp. 1–6, 2018.

[11] P. Yin, C. Wang, W. Liu, E. E. Swartzlander, and F. Lombardi, "Designs of approximate floating-point multipliers with variable accuracy for error-tolerant applications," *J. Signal Process. Syst.*, vol. 90, pp. 641–654, 2018.

[12] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," *ESSCIRC*, pp. 465–468, 2016.

[13] C. Yan, X. Zhao, T. Zhang, J. Ge, C. Wang, and W. Liu, "Design of high hardware efficiency approximate floating-point fft processor," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[14] T. Cheng, Y. Masuda, J. Chen, J. Yu, and M. Hashimoto, "Logarithm-approximate floating-point multiplier is applicable to power-efficient neural network training," *Integration*, vol. 74, pp. 19–31, 2020.

[15] S. O'uchi, H. Fuketa, T. Ikegami, W. Nogami, T. Matsukawa, T. Kudoh, and R. Takano, "Image-classifier deep convolutional neural network training by 9-bit dedicated hardware to realize validation accuracy and energy efficiency superior to the half precision floating point format," *ISCAS*, pp. 1–5, 2018.

[16] M. Franceschi, A. Nannarelli, and M. Valle, "Tunable floating-point for artificial neural networks," *ICECS*, pp. 289–292, 2018.

[17] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," *ICASSP*, pp. 5900–5904, 2017.

[18] T. Zhang, Z. Niu, and J. Han, "A brief review of logarithmic multiplier designs," in *LATS*. IEEE, 2022, pp. 1–4.

[19] X. Sun, N. Wang, C. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. E. Maghraoui, V. Srinivasan, and K. Gopalakrishnan, "Ultra-low precision 4-bit training of deep neural networks," *NeurIPS*, pp. 1796–1807, 2020.

[20] N. Wang, J. Choi, D. Brand, C. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," *NeurIPS*, p. 7685–7694, 2018.

[21] B. Liu, Z. Wang, X. Wang, R. Zhang, A. Xue, Q. Shen, N. Xie, Y. Gong, Z. Wang, J. Yang *et al.*, "An efficient BCNN deployment method using quality-aware approximate computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4217–4228, 2022.

[22] B. Liu, H. Cai, Z. Zhang, X. Ding, Z. Wang, Y. Gong, W. Liu, J. Yang, Z. Wang, and J. Yang, "More is less: Domain-specific speech recognition microprocessor using one-dimensional convolutional recurrent neural network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1571–1582, 2021.

[23] Z. Niu, H. Jiang, M. S. Ansari, B. F. Cockburn, L. Liu, and J. Han, "A logarithmic floating-point multiplier for the efficient training of neural networks," *GLSVLSI*, no. 6, p. 65–70, 2021.

[24] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. on Electronic Computers*, vol. 11, no. 4, pp. 512–517, 1962.

[25] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE TC*, vol. 70, no. 4, pp. 614–625, 2021.

[26] V. Mahalingam and N. Ranganathan, "Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition," *IEEE TC*, vol. 55, no. 12, pp. 1523–1535, 2006.

[27] D. Nandan, J. Kanungo, and A. Mahajan, "An efficient VLSI architecture design for logarithmic multiplication by using the improved operand decomposition," *Integration*, vol. 58, pp. 134–141, 2017.

[28] Z. Babić, A. Avramović, and P. Bulić, "An iterative logarithmic multiplier," *Microprocessors and Microsystems*, vol. 35, no. 1, pp. 23–33, 2011.

[29] S. E. Ahmed and M. B. Srinivas, "An improved logarithmic multiplier for media processing," *J. Signal Process. Syst.*, vol. 91, no. 6, pp. 561–574, 2019.

[30] H. Kim, M. S. Kim, A. A. Del Barrio, and N. Bagherzadeh, "A cost-efficient iterative truncated logarithmic multiplication for convolutional neural networks," *ARITH*, pp. 108–111, 2019.

[31] M. S. Kim, A. A. D. Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell's approximate log multipliers for convolutional neural networks," *IEEE TC*, vol. 68, no. 5, pp. 660–675, 2019.

[32] 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 TCAS-I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[33] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2009.

[34] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[35] R. Pilipović, P. Bulić, and U. Lotrič, "A two-stage operand trimming approximate logarithmic multiplier," *IEEE TCAS-I: Regular Papers*, vol. 68, no. 6, pp. 2535–2545, 2021.

[36] P. Yin, C. Wang, H. Waris, W. Liu, Y. Han, and F. Lombardi, "Design and analysis of energy-efficient dynamic range approximate logarithmic

multipliers for machine learning," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 612–625, 2020.
[37] J. Y. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 3, pp. 273–286, 2000.
[38] F. Fang, T. Chen, and R. A. Rutenbar, "Floating-point bit-width optimization for low-power signal processing applications," *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. III–3208–III–3211, 2002.
[39] J. Eilert, A. Ehliar, and D. Liu, "Using low precision floating point numbers to reduce memory cost for MP3 decoding," *IEEE 6th Workshop on Multimedia Signal Processing*, pp. 119–122, 2004.
[40] A. Gupta, S. Mandavalli, V. J. Mooney, K. Ling, A. Basu, H. Johan, and B. Tandianus, "Low power probabilistic floating point multiplier design," *ISVLSI*, pp. 182–187, 2011.
[41] H. Zhang, W. Zhang, and J. Lach, "A low-power accuracy-configurable floating point multiplier," *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pp. 48–54, 2014.
[42] "IEEE Standard for Binary Floating-point Arithmetic," *ANSI/IEEE Std 754-1985*, pp. 1–20, 1985.
[43] M. Imani, D. Peroni, and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," *Proceedings of the 54th Annual Design Automation Conference*, pp. 1–6, 2017.
[44] J. Ge, C. Yan, X. Zhao, K. Chen, B. Wu, and W. Liu, "An energy-efficient approximate floating-point multipliers for wireless communications," *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 1–5, 2022.
[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, p. 8026–8037, 2019.
[46] C. Chang and C. Lin, "Fourclass," 1996.
[47] U. M. L. Repository, "Human activity recognition using smartphones data set," 2012, Center for Machine Learning and Intelligent Systems, University of California, Irvine.
[48] Y. LeCun, C. Cortes, and C. Burges, "MNIST Handwritten Digit Database," 2001, National Institute of Standards and Technology, Gaithersburg, MD, USA.
[49] V. N. A. Krizhevsky and G. Hinton, "The CIFAR-10 Dataset," 2014.
[50] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," *Proc. of the 20th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pp. 661–670, 2014.
[51] L. Prechelt, "Early stopping-but when?" *Neural Networks: Tricks of the trade*, pp. 55–69, 1998.

Zijing Niu received the B.Sc. in Microelectronic Science and Engineering from the Sichuan University, Chengdu, China, in 2018. She is working toward the M.Sc. degree in the Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada, since Sep. 2018. Her research interests include approximate computing, hardware designs for accelerating deep learning applications.

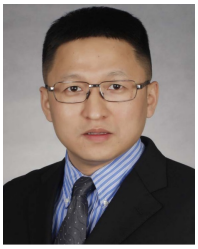
Tingting Zhang (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in the College of Electronic and Information Engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2016 and 2019, respectively. She is working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada, since Sep. 2019. Her research interests include approximate computing, Ising computing, combinatorial optimization and nanoelectronic circuits and systems.

Honglan Jiang (Member, IEEE) received the B.Sc. and master's degrees in instrument science and technology from the Harbin Institute of Technology, Harbin, Heilongjiang, China, in 2011 and 2013, respectively, and the Ph.D. degree in integrated circuits and systems from the University of Alberta, Edmonton, AB, Canada, in 2018. From 2018 to 2021, she worked as a Post-Doctoral Fellow with the School of Integrated Circuits, Tsinghua University, Beijing, China. She is currently an Associate Professor with the Department of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai, China. Her research interests include approximate computing, reconfigurable computing, and stochastic computing.

Bruce F. Cockburn (S'86-M'90) received the B.Sc. degree in Engineering Physics in 1981 from Queen's University, Kingston, ON, Canada. In 1985 and 1990 he received the M.Math. and Ph.D. degrees, respectively, in Computer Science from the University of Waterloo, Waterloo, ON.

He is presently a Professor in the Department of Electrical and Computer Engineering at the University of Alberta, Edmonton, AB, Canada. From 1981 to 1983 he was a Test Engineer and Software Designer at Mitel Corporation in Kanata, ON. In 2001 he was a sabbatical visitor at Agilent Technologies Inc. in Santa Clara, CA, USA. From 2014 to 2015 he was a sabbatical visitor at the University of British Columbia in Vancouver, BC, Canada. His research interests include the testing and verification of integrated circuits, application-specific hardware accelerators, novel applications of high-level synthesis and field-programmable gate arrays, heterogeneous parallel computing, stochastic and approximate computing, and genetic data processing algorithms and hardware accelerators.

Dr. Cockburn is also a member of the Association for Computing Machinery and is registered as a Professional Engineer with the Association of Professional Engineers and Geoscientists of Alberta.



Leibo Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering and the Ph.D. degree from the School of Integrated Circuits, Tsinghua University, Beijing, China, in 1999 and 2004, respectively. He is currently a Professor with the School of Integrated Circuits, Tsinghua University. His current research interests include reconfigurable computing, mobile computing, and VLSI digital signal processing.

Jie Han (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Delft University of Technology, Delft, The Netherlands, in 2004. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, and novel computational models for nanoscale and biological applications.