A Logarithmic Floating-Point Multiplier for the Efficient Training of Neural Networks

Zijing Niu
University of Alberta
Edmonton, AB, Canada
zijing2@ualberta.ca

Honglan Jiang
Tsinghua University
Beijing, China
honglan@ualberta.ca

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

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

Leibo Liu
Tsinghua University
Beijing, China
liulb@tsinghua.edu.cn

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

ABSTRACT

The development of important applications of increasingly large neural networks (NNs) is spurring research that aims to increase the power efficiency of the arithmetic circuits that perform the huge amount of computation in NNs. The floating-point (FP) representation with a large dynamic range is usually used for training. In this paper, it is shown that the FP representation is naturally suited for the binary logarithm of numbers. Thus, it favors a design based on logarithmic arithmetic. Specifically, we propose an efficient hardware implementation of logarithmic FP multiplication that uses simpler operations to replace complex multipliers for the training of NNs. This design produces a double-sided error distribution that mitigates the accumulative effect of errors in iterative operations, so it is up to 45% more accurate than a recent logarithmic FP design. The proposed multiplier also consumes up to 23.5× less energy and 10.7× smaller area compared to exact FP multipliers. Benchmark NN applications, including a 922-neuron model for the MNIST dataset, show that the classification accuracy can be slightly improved using the proposed multiplier, while achieving up to 2.4× less energy and 2.8× smaller area with a better performance.

CCS CONCEPTS

• **Hardware** → **Combinational circuits**; *Application specific integrated circuits*; • **Computer systems organization** → *Neural networks.*

KEYWORDS

Floating-Point Multiplier; Neural Network; Approximate Computing

ACM Reference Format:
Zijing Niu, Honglan Jiang, Mohammad Saeed Ansari, Bruce F. Cockburn, Leibo Liu, and Jie Han. 2021. A Logarithmic Floating-Point Multiplier for the Efficient Training of Neural Networks. In *Proceedings of the Great Lakes Symposium on VLSI 2021 (GLSVLSI '21), June 22–25, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3453688.3461509

GLSVLSI '21, June 22–25, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8393-6/21/06…$15.00
https://doi.org/10.1145/3453688.3461509

1 INTRODUCTION

Artificial neural networks (NNs) are computational models that possess attractive characteristics of the biological NNs of the brain. NNs have been widely explored and used in many areas such as signal processing, image analysis and medical diagnosis [13]. An artificial NN requires massive multiply-accumulate (MAC) arithmetic computations in both the training and inference phases. With the increasing size of NNs, the amount of MAC computation becomes a limiting factor. This challenge motivates the search for more efficient computation processes and hardware implementations.

Various methods have been explored for accelerating NNs. They include reducing redundancies in structure, optimizations of gradient-based backpropagation and decreasing the computation intensity in the convolution to improve training and inference [18][21]. Taking advantages of the error tolerance in NNs, approximate computing is a promising technique to improve the computational and energy efficiency in deep learning applications [7].

To ensure the accuracy of NN models, a floating-point (FP) representation is usually adopted in the training phase as the wider range of representation leads to more accurate training. Since the FP MAC circuits, especially multiplication, dominate power consumption and circuit area, the design of efficient FP multipliers has been extensively investigated [2, 15, 20]. Low-precision computation in training and inference has recently been pursued. Significant progress has been made in exploiting reduced-precision integers for inference, while 8-bit FP numbers [19] and logarithmic 4-bit FP numbers [17] have been shown to be effective in the training of deep NNs. Hence, finding a balance between accuracy, speed and area in the implementation of NNs is a key challenge as apparently no hardware has been proposed for these low-precision NNs.

In this work, we show that the standard IEEE 754 FP representation is naturally suited for logarithmic arithmetic. A logarithmic FP multiplier is designed using simple operators such as adders and multiplexers. Unlike other approximate FP multipliers, a double-sided error distribution is produced by the proposed design, which minimizes the increase of accumulative errors. The proposed design also improves the energy efficiency of training with only a slight

loss of accuracy. In some cases, it even improves the accuracy of NNs compared to those using exact multipliers.

The rest of this paper is organized as follows: Section 2 describes the research motivation and related work. Section 3 introduces the logarithmic FP representation and formulation for multiplication. Section 4 presents the proposed multiplier with an accuracy and performance evaluation. Section 5 presents NN applications and experimental results. Finally, Section 6 concludes the paper.

2 MOTIVATION AND RELATED WORK

The training phase, which plays a key role in achieving a high NN accuracy, requires more arithmetic computation than inference due to the use of the iterative gradient descent algorithm for updating weights and biases. Since inference is less sensitive to precision reduction with simpler computations, fixed-point arithmetic units dealing with fixed-point numbers are applied in the inference phase. Many approximate designs have been developed to perform fixed-point multiplication, such as truncated multipliers and logarithm-based multipliers [9]. However, with a wider range of representation, the FP arithmetic unit, especially the FP multiplier, permits greater training accuracy at the cost of higher power consumption and larger area usage.

Hence, hardware-efficient FP multipliers have mostly been explored for the training of NNs. A short-bit-length FP format is considered in [8] for the training of convolutional NNs. An approximate computing technique called Tunable Floating-Point (TFP) adjusts the precisions for different operations to lower the power consumption [5]. Both of the above proposals depend on bit width reduction of the FP representation to increase the efficiency, similar to most of the other studies focused on bit-width scaling [5]. In [15], mantissa multiplication is converted to addition of the input operands; however, exact multiplication is required when the error rate exceeds a pre-determined value. These methods do not completely eliminate the multiplication.

As an energy-efficient 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 more accurate encoding of weights and activations that results in higher classification accuracies at low resolutions [11]. A state-of-the-art 4-bit training strategy for deep NNs is based on the logarithmic radix-4 format [17]. Hence, efficient logarithm-based FP multipliers have become promising for the training of NNs.

Recently, a logarithmic approximate multiplier (LAM) was proposed to improve the power efficiency of NN training by implementing FP multiplication with fixed-point addition [4]. However, the LAM always underestimates the product, so approximation errors are accumulated in the training process. Different from the conventional logarithmic approximation, which uses the highest power of two smaller than the given number, a nearest-one detector (NOD) is proposed in [1] to find the nearest power of two to a given input. Since the improved logarithmic multiplier is designed for fixed-point numbers, the NOD evaluates from the most significant bit to the least significant bit of the inputs. Taking advantage of the IEEE 754 FP format, however, the nearest power of two for a number in the FP representation can readily be determined without using a NOD, as shown next.

3 LOGARITHMIC FLOATING-POINT REPRESENTATION AND FORMULATION FOR MULTIPLICATION

3.1 FP Representation and Multiplication

The IEEE 754 standard defines the most commonly used formats for representing FP numbers. The IEEE 754 FP formats contain a 1-bit sign S, a p-bit exponent E and a q-bit mantissa M. Fig. 1 shows the IEEE 754 representation of a single-precision FP number.

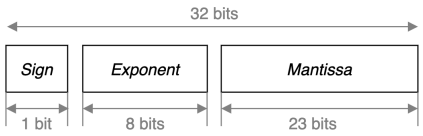


Figure 1: The IEEE-754 single-precision format.

An FP number N in base-2 scientific notation is expressed as:

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

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, such as 127 for the single-precision, is added to the actual exponent value. $E - bias$ denotes the actual exponent value of N. With the hidden '1', x is the fractional part of the FP number, represented by the mantissa M, and hence $0 \le x < 1$. Note that X is used to denote the actual mantissa, $1 + x$, in the following content.

In the IEEE 754 formats, the FP multiplication can be calculated by three processes, including the XOR operation for the sign bits, the addition of the exponents, and the multiplication of the mantissa bits. Consider $P = A \times B$, which is computed as follows:

$$S_P = S_A \oplus S_B, \tag{2}$$

$$X_{AB} = (1 + x_A) \times (1 + x_B), \tag{3}$$

$$E_P = \begin{cases} E_A + E_B - bias, & X_{AB} < 2, \\ E_A + E_B - bias + 1, & otherwise, \end{cases} \tag{4}$$

$$X_P = \begin{cases} X_{AB}, & X_{AB} < 2, \\ X_{AB}/2, & otherwise, \end{cases} \tag{5}$$

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 (2) is valid for the sign computation of the proposed design as well, so it will not be discussed in the following content.

3.2 Logarithmic FP Representation

An FP number N is first converted into the format of its nearest power of two and the corresponding mantissa. Since the IEEE 754 FP format provides the highest power of two smaller than N, by comparing the fraction x with 0.5, the nearest power of two can be determined for N. This process is further simplified to checking whether the leading bit of the explicit mantissa is '1' in the circuit implementation.

If $x \ge 0.5$, N is closer to $2^{E-bias+1}$ than 2^{E-bias} (or equally away for the equal sign). The exponent E is then incremented by 1 and, accordingly, the mantissa becomes $\frac{1+x}{2}$. In contrast, the exponent

and mantissa of N remain the same when $x < 0.5$. Let the converted exponent of N be denoted as E' and the converted mantissa as X'. Then E' and X' are given by:

$$E' = \begin{cases} E, & x < 0.5, \\ E+1, & x \geq 0.5, \end{cases} \tag{6}$$

$$X' = 1 + x' = \begin{cases} 1+x, & x < 0.5, \\ \frac{1+x}{2}, & x \geq 0.5. \end{cases} \tag{7}$$

3.3 Mathematical Formulation for the Proposed Design

The logarithm approximation used in the proposed design is based on $\log_2(1+k) \cong k$ from Mitchell [12]. This logarithm approximation and its anti-logarithm approximation are both applied over the region of $0 \leq k < 1$ in most logarithmic multipliers. Hence, for the logarithmic N using the nearest one representation, the exponent is given in (6) and $\log_2(X')$ is approximated as follows:

$$\log_2(X') \cong \begin{cases} x, & x < 0.5, \\ \frac{1+x}{2} - 1, & x \geq 0.5. \end{cases} \tag{8}$$

Note that the logarithm approximation, $\log_2(1+k) \cong k$, is applied over the region $-0.25 \leq k < 0.5$ in (8).

For $P = A \times B$ in the logarithm domain, the exponent is still computed by addition, whereas the multiplication for the mantissa is converted to addition. Let $X'_{AB} = (1+x'_A) \times (1+x'_B)$, and hence the logarithm of X'_{AB} is given by:

$$\log_2(X'_{AB}) = \log_2(1+x'_A) + \log_2(1+x'_B) \cong x'_A + x'_B. \tag{9}$$

Using the anti-logarithm approximation, $2^k \cong k+1$, applied over the region $-0.5 \leq k < 1$, X'_{AB} is computed as:

$$X'_{AB} \cong 2^{x'_A + x'_B} \cong 1 + x'_A + x'_B. \tag{10}$$

According to (8), we obtain $-0.5 \leq x'_A + x'_B < 1$, and thus, $0.5 \leq X'_{AB} < 2$. When $X'_{AB} < 1$ (or $x'_A + x'_B < 0$), X'_{AB} cannot be directly adopted as the mantissa of product P. Therefore, in this case, X'_{AB} is multiplied by 2 and, accordingly, the exponent is reduced by 1.

Finally, the logarithmic FP multiplication is given by:

$$E_P = \begin{cases} E'_A + E'_B - bias, & x'_A + x'_B \geq 0, \\ E'_A + E'_B - bias - 1, & otherwise, \end{cases} \tag{11}$$

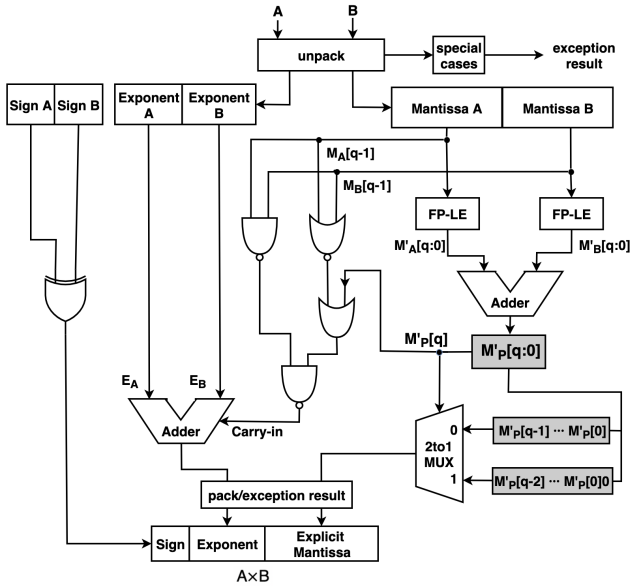
$$X_P = \begin{cases} 1 + x'_A + x'_B, & x'_A + x'_B \geq 0, \\ (1 + x'_A + x'_B) \times 2, & otherwise. \end{cases} \tag{12}$$

Note that E'_A and E'_B are the converted exponents, and x'_A and x'_B are the approximate logarithms given in the converted mantissas, for A and B, respectively.

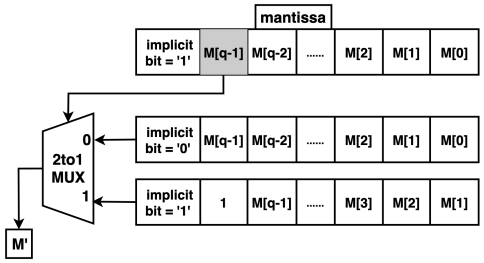
4 FLOATING-POINT LOGARITHMIC MULTIPLIER DESIGN AND EVALUATION

4.1 Hardware Implementation

The design of the proposed FP logarithmic multiplier, denoted as FPLM, is shown in Fig. 2. Taking advantage of the IEEE 754 FP format, the exponent and the explicit mantissa of the FP number can directly be obtained. Thus, for the hardware implementation, $1.M$ is used to denote the actual mantissa of the given FP number. Here, M contains q bits, e.g., 23 bits in the single-precision, after the binary point and the implicit '1', given as $M[q-1]M[q-2]\cdots M[1]M[0]$ in Fig. 2. Note that $1.M$ is equivalent to $1+x$ in (1).



(a) The proposed logarithmic floating-point multiplier.



(b) The proposed floating-point logarithm estimator (FP-LE).

Figure 2: The design of the proposed logarithmic floating-point multiplier.

An FP logarithm estimator (FP-LE) is further proposed to compute the approximate logarithm of the mantissa based on the nearest power of two for the FP number. Simple arithmetic operations are used to implement the FP-LE, as shown in Fig. 2(b). The nearest power of two for each FP number can simply be determined by checking the leading bit of the explicit mantissa, $M[q-1]$, and hence a 2-to-1 multiplexer is used to obtain the approximate logarithmic value x', implemented by M'. When $M[q-1] = 1$, according to (8), the approximate logarithm, $x' = (1+x)/2 - 1$, is obtained as a negative number in 2's complement, i.e., $M' = 1.1M[q-1]\cdots M[1]$. Otherwise, $x' = x$; M' is obtained as $0.M[q-1]\cdots M[0]$.

As shown in Fig. 2(a), M'_A and M'_B obtained from the FP-LEs are then summed to compute the explicit mantissa. In this addition, the addend, 1, for $1 + x'_A + x'_B$ in (10), is omitted as the hidden bit and it has no effect on the explicit mantissa of the sum. Let M'_P denote the sum of M'_A and M'_B in the form of $M'_P[q].M'_P[q-1]\cdots M'_P[0]$ with

$q+1$ bits. As per (12), when $x'_A + x'_B < 0$, which means $M'_P[q] = 1$ in 2's complement, the mantissa, $(1 + x'_A + x'_B) \times 2$, is obtained as $1.M'_P[q-2] \cdots M'_P[0]0$. Otherwise, the mantissa, $1 + x'_A + x'_B$, is obtained as $1.M'_P[q-1] \cdots M'_P[0]$.

According to (6), the exponents of the two operands are modified first, practically depending on $M_A[q-1]$ and $M_B[q-1]$. Then the modified exponents are summed subsequently to obtain the exponent of the product, which is subtracted by 1 if $M'_p[q] = 1$, as per (11). The circuits that are required to implement the above operations can be simplified to only one adder with a carry-in bit that determines the modified value of $E_A + E_B$. As shown in Fig. 2(a), four logic gates are used to generate the carry-in bit based on the truth table obtained with $M_A[q-1]$, $M_B[q-1]$ and $M'_p[q]$ as inputs. For example, when both $M_A[q-1]$ and $M_B[q-1]$ are '1', $E_A + E_B$ is added with 2 as per (6), but a 1 is subtracted since $M'_P[q] = 1$ as in (11). Therefore, in this case, the carry-in bit is '1'. The exponent is then added with the bias to be stored in the IEEE standard format.

Any exception result (such as overflow, underflow, and not a number) is reported by detecting the two operands and the final result. Note that the rounding unit is not required in the inexact design since it is inherently imprecise.

4.2 Performance of the FP Logarithmic Multiplier

The performance of the proposed FPLM is evaluated by comparing it with the exact FP multiplier (EFM) and a recent multiplier, LAM [4], in terms of accuracy and circuit performance. The multiplier in [1] is not applicable for this comparison since it was designed for fixed-point numbers.

4.2.1 Approximation error analysis. The approximations of the function $\log_2(1+k)$ used in the two multipliers are shown in Fig. 3, in which they are compared to the exact function.

As shown in Fig. 3, the LAM always underestimates the multiplication result since the approximation method $\log_2(1+k) \cong k$ is adopted over the interval $(0, 1)$ for k. FPLM computes the same underestimated results as the LAM when both input operands are closer to 2^{E-bias}. However, the input operands closer to $2^{E-bias+1}$ are overestimated when converted to the logarithmic domain. In FPLM, when one of the input operands is closer to 2^{E-bias} and the other one is closer to $2^{E-bias+1}$, the multiplication error can be reduced due to the aforementioned overestimation and underestimation of the inputs. For a logarithm multiplier, the antilogarithm approximation, $2^k \cong (1+k)$, can offset some errors in the last step since the approximation direction is opposite to that of $\log_2(1+k) \cong k$.

4.2.2 Multiplication accuracy evaluation. The accuracy of the multiplication is assessed by using a sample of 10^3 uniformly distributed random input combinations, as shown in Fig. 4. Since the proposed FPLM computes the accurate exponent of the product, the random numbers are generated over the interval of $[1, 2)$ by ignoring the exponent for simplicity. As can be seen, the FPLM produces double-sided errors.

The mean relative error distance (MRED) is the average value of all possible relative (absolute) error distances and the average

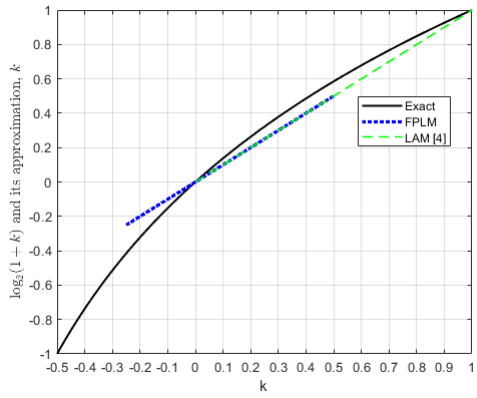


Figure 3: Approximations of $\log_2(1+k)$.

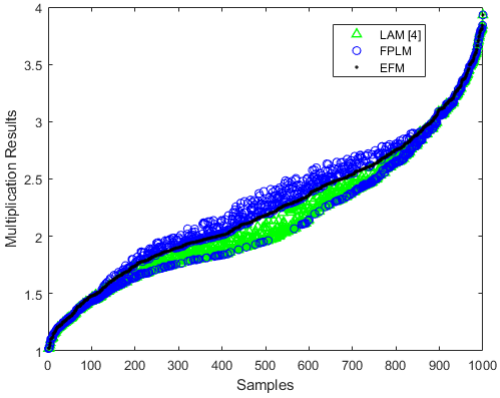


Figure 4: Multiplication accuracies of FPLM, LAM [4] and EFM.

error (AE) is the average (signed) difference between the exact and approximate products. Both metrics and four FP precision formats: 32-bit single-precision, 16-bit half-precision, Brain Floating Point (Bfloat16) format and 8-bit FP (FP8) format in the form of (1, 5, 2) bits for the sign, exponent and mantissa, are considered for the evaluation of each multiplier. The FP8 (1, 5, 2) format is chosen since it performs best with respect to the classification accuracy. It has been adopted to an application of NN training [19]. A sample of 10^7 random cases of uniform and standard normal distributions were generated respectively to obtain the results in Tables 1 and 2. It is shown that, for a normal distribution, the FPLM performs up to 31% more accurately in 32- and 16-bit precisions with respect to MRED, and has lower AE, compared to LAM. For a uniform distribution, the FPLM is more accurate in the single-precision, half-precision and Bfloat16 formats with up to 45% smaller MRED and over 10^3 $\times$ smaller AE. For the FP8 format, both of the multipliers produce significantly larger errors; the FPLM is slightly less accurate than LAM.

4.2.3 Hardware evaluation. The proposed FPLM and the LAM in [4] were implemented in Verilog and an EFM was obtained using the Synopsys DesignWare IP library (DW_fp_mult). All of the designs were synthesized using the Synopsys Design Compiler (DC) for

Table 1: Mean Relative Error Distance of Multipliers

	Single-Precision	Half-Precision	Bfloat16	FP8
Uniform Distribution				
FPLM	**0.0289**	**0.0289**	**0.0302**	0.2311
LAM [4]	0.0385	0.0391	0.0437	**0.1915**
Normal Distribution				
FPLM	**0.0288**	**0.0311**	**0.0300**	0.2160
LAM [4]	0.0382	0.0410	0.0434	**0.1891**

Table 2: Average Error of Multipliers

	Single-Precision	Half-Precision	Bfloat16	FP8
Uniform Distribution				
FPLM	$\mathbf{3.2\times10^{-5}}$	$\mathbf{2.2\times10^{-3}}$	**0.0176**	0.5630
LAM [4]	0.0833	0.0848	0.0950	**0.4380**
Normal Distribution				
FPLM	$\mathbf{8.3\times10^{-6}}$	$\mathbf{8.2\times10^{-6}}$	$\mathbf{7.4\times10^{-6}}$	$\mathbf{3.5\times10^{-5}}$
LAM [4]	1.2×10^{-5}	1.3×10^{-5}	1.3×10^{-5}	4.8×10^{-5}

STM's CMOS 28-nm process with a supply voltage of 1.0 V and a temperature of 25°C. The same process and the same optimization option were used to ensure a fair comparison. The evaluation results are shown in Table 3. All of the designs are evaluated with a 500-MHz clock frequency.

As shown in Table 3, the FPLM consumes 20.8× less PDP and 10.7× smaller area compared to the EFM in the single-precision format, while it consumes 23.5 × less energy and 6.5× smaller area in the FP8 format. Also, the FPLM achieves a shorter delay compared to the LAM [4] in the single-precision format. It is interesting to observe that the FPLM consumes less energy in Bfloat16 compared to the consumption in the half-precision format, while LAM is the opposite. Although LAM has the smallest PDP, its accuracy is lower as the trade-off. It is necessary to point out that, according to [6], the power consumption of the accurate FP multiplier is dominated by the mantissa multiplication for over 80% and the rounding unit for nearly 18%. Therefore, the power and area cost can be largely reduced due to the elimination of the mantissa multiplier and the rounding unit in the proposed design.

5 NEURAL NETWORK APPLICATIONS

5.1 Experimental Setup

The proposed logarithmic FP multiplier is used in the arithmetic unit in a multi-layer perceptron (MLP) to illustrate their performance in NNs with respect to the classification accuracy and hardware performance. It is evaluated against the EFM considered as the baseline arithmetic unit, and the LAM in [4]. In the experiments, the exact multiplication is replaced with approximate designs in the training phase by using the Pytorch framework [14]. To fairly evaluate the effect of approximate multiplication on training, the multiplier used in the inference engine is an exact FP multiplier with the same precision as the approximate one. An NN employing approximate multipliers of four precisions was trained using the same number of epochs and each training was repeated five times using random weight initializations. Since the code quality and GPU performance affect the acceleration of training, the training time is not eligible for comparison.

Table 3: Circuit Assessment of the FP Multipliers

	FPLM	LAM [4]	EFM
Single-Precision			
Power (μW)	67.2	39.9	1366.9
Delay (ns)	1.66	1.69	1.70
Area (μm^2)	270.4	138.8	2910.3
PDP (fJ)	111.5	67.5	2323.7
Half-Precision			
Power (μW)	29.5	16.1	375.7
Delay (ns)	1.26	0.92	1.70
Area (μm^2)	116.8	76.5	1057.5
PDP (fJ)	37.2	14.8	638.6
Bfloat16			
Power (μW)	30.4	18.0	252.2
Delay (ns)	1.22	0.92	1.70
Area (μs^2)	127.1	89.9	887.6
PDP (fJ)	37.1	16.5	428.7
FP8			
Power (μW)	13.9	9.6	94.5
Delay (ns)	0.49	0.42	1.70
Area (μm^2)	58.5	48.1	385.6
PDP (fJ)	6.8	4.0	160.6

Three classification datasets, the fourclass [3], the HARS [16] and the MNIST [10], were used for the evaluation. A small MLP network was used for training the fourclass. The MLP networks used for the HARS and MNIST are (561, 40, 6) and (784, 128, 10) models, respectively. The activation functions for the hidden layer and output layer are the Rectified Linear Unit (ReLU) function and sigmoid function, respectively.

5.2 Evaluation Results

5.2.1 Classification Accuracy. The comparison of the classification accuracy is shown in Fig. 5 to indicate the relative approximation error of the logarithmic FP multipliers. Since the benchmark NNs using EFM for training are considered as baseline models, the baseline accuracies for the four precisions are listed in Fig. 5 in the order of the single-precision, half-precision, Bfloat16 and FP8 formats.

The proposed FPLM produces a higher accuracy than LAM for the three datasets in the single-precision, half-precision, and Bfloat16 format, while it degrades more in the FP8 format. For the MNIST, it is interesting to observe that the FPLM slightly improves the classification accuracy in the single-precision and Bfloat16 formats. However, the classification accuracy for LAM based training shows degradation in all the four precisions, with 2.068% in the half-precision. The FPLM shows up to 0.21% and 2.19% higher accuracy than LAM for the HARS and fourclass, respectively. It also indicates that the FPLM performs more accurately in the classification of larger dataset due to the offset effect of the double-sided errors.

5.2.2 Hardware Evaluation. The circuit of an artificial neuron with two inputs was measured to indicate the hardware cost of the implemented NNs. The FP adder used in the neuron was obtained using the Synopsys DesignWare IP library (DW_fp_add). The Bfloat16

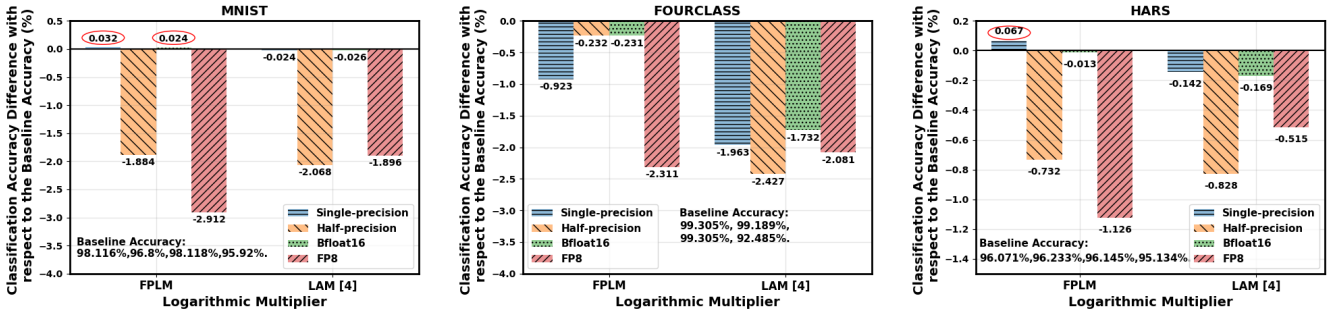


Figure 5: Comparison of the classification accuracy of three datasets with LMs for four precision levels: a negative percentage means a decrease and a positive percentage means an increase in accuracy from using accurate multipliers.

was selected since the two designs performed very well for the classification (Fig. 5) with a low energy consumption (Table 3).

The simulation results in Table 4, obtained at a 200-MHz clock frequency, show that although the neuron using the LAM is more hardware efficient, the neuron using the FPLM consumes 2.4× less energy and is 2.8× smaller compared to the EFM based neuron. In general, the hardware improvements are smaller than that in Table 3 since they are limited by the FP adder.

Table 4: Circuit Assessment of the Artificial Neuron

	FPLM	LAM [4]	EFM
Power (μW)	124.0	113.8	263.5
Delay (ns)	4.10	4.66	4.69
Area (μm^2)	674.8	606.9	1919.8
PDP (fJ)	508.4	416.5	1235.8

6 CONCLUSION

With the increasing size of neural networks, the amount of arithmetic computation is getting larger and thus computation must become more efficient to be practical in energy-constrained systems. In this paper, a floating-point logarithmic multiplier (FPLM) is proposed for the complex FP multiplication with simple arithmetic operations. The evaluation results show that the FPLM consumes up to 23.5× less PDP and 10.7× smaller area compared to the EFM, while being up to 45% more accurate than a recent FP design. The FPLM-based NN achieves higher classification accuracy in 32- and 16-bit FP precisions compared to the recent design. It consumes 2.4× less energy and it is 2.8× smaller compared to the EFM based neuron in the FP8 format. Interestingly, using FPLM slightly increases the classification accuracy in some cases. In future work, the proposed design will be applied to the training of convolutional NNs and efficient strategies will be investigated to improve the low-precision training.

ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Number: RES0048688). We thank CMC Microsystems for providing access to the simulation tools.

REFERENCES

[1] M. S. Ansari, B. F. Cockburn, and J. Han. 2021. An Improved Logarithmic Multiplier for Energy-Efficient Neural Computing. *IEEE TC* 70, 4, 614–625.
[2] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak. 2016. Approximate 32-bit floating-point unit design with 53% power-area product reduction. In *ESSCIRC*. 465–468.
[3] C. Chang and C. Lin. 1996. Fourclass. https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/binary.html
[4] T. Cheng, Y. Masuda, J. Chen, J. Yu, and M. Hashimoto. 2020. Logarithm-approximate floating-point multiplier is applicable to power-efficient neural network training. *Integration* 74, 19–31.
[5] M. Franceschi, A. Nannarelli, and M. Valle. 2018. Tunable Floating-Point for Artificial Neural Networks. In *ICECS*. 289–292.
[6] A. Gupta, S. Mandavalli, V. J. Mooney, K. Ling, A. Basu, H. Johan, and B. Tandianus. 2011. Low Power Probabilistic Floating Point Multiplier Design. In *ISVLSI*. 182–187.
[7] J. Han and M. Orshansky. 2013. Approximate computing: An emerging paradigm for energy-efficient design. In *ETS*. 1–6.
[8] S. i. O'uchi, H. Fuketa, T. Ikegami, W. Nogami, T. Matsukawa, T. Kudoh, and R. Takano. 2018. 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. In *ISCAS*. 1–5.
[9] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han. 2020. Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. *Proc. IEEE* 108, 12, 2108–2135.
[10] Y. LeCun, C. Cortes, and C. Burges. 2001. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist
[11] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong. 2017. LogNet: Energy-efficient neural networks using logarithmic computation. In *ICASSP*. 5900–5904.
[12] J. N. Mitchell. 1962. Computer Multiplication and Division Using Binary Logarithms. *IRE Trans. on Electronic Computers* 11, 4, 512–517.
[13] A. R. Omondi and J. C. Rajapakse (Eds.). 2010. *FPGA Implementations of Neural Networks* (1st. ed.). Springer Publishing Company, Incorporated.
[14] 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. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8026–8037.
[15] D. Peroni, M. Imani, and T. S. Rosing. 2020. Runtime Efficiency-Accuracy Tradeoff Using Configurable Floating Point Multiplier. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 39, 2, 346–358.
[16] Uci Machine Learning Repository. 2012. Human Activity Recognition Using Smartphones Data Set. https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones
[17] X. Sun, N. Wang, C. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. E. Maghraoui, V. Srinivasan, and K. Gopalakrishnan. 2020. Ultra-Low Precision 4-bit Training of Deep Neural Networks. In *NeurIPS*. 1796–1807.
[18] V. Sze, Y. Chen, T. Yang, and J. S. Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* 105, 12, 2295–2329.
[19] N. Wang, J. Choi, D. Brand, C. Chen, and K. Gopalakrishnan. 2018. Training Deep Neural Networks with 8-Bit Floating Point Numbers. In *NeurIPS*. 7685–7694.
[20] P. Yin, C. Wang, W. Liu, E. E. Swartzlander, and F. Lombardi. 2018. Designs of Approximate Floating-Point Multipliers with Variable Accuracy for Error-Tolerant Applications. *J. Signal Process. Syst.* 90, 641–654.
[21] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu. 2019. Recent advances in convolutional neural network acceleration. *Neurocomputing* 323, 37–51.