A Brief Review of Logarithmic Multiplier Designs

Tingting Zhang
Department of Electrical & Computer Engineering
University of Alberta
Edmonton, Canada
ttzhang@ualberta.ca

Zijing Niu
Department of Electrical & Computer Engineering
University of Alberta
Edmonton, Canada
zijing2@ualberta.ca

Jie Han
Department of Electrical & Computer Engineering
University of Alberta
Edmonton, Canada
jhan8@ualberta.ca

***Abstract*—The base-2 logarithmic arithmetic converts multiplication to hardware-efficient shift and addition. Aimed for a good trade-off between circuit complexity and accuracy, a logarithmic multiplier inherently introduces inevitable approximation into the computation. This paper briefly reviews logarithmic multiplier designs from different perspectives on improving performance and accuracy, followed by a summary of their applications in error-tolerant systems.**

***Index Terms*—Approximate computing, logarithmic multiplier, neural network**

I. Introduction

The logarithmic number system (LNS) is an alternative to conventional fixed-point (FxP) and floating-point (FP) number representations in a digital system. It benefits from a wider numerical range than FxP [1], a lower round-off noise than FP [2] and a milder switching activity [3]. Recent researches show that the LNS with a reduced bit-width of numbers achieves a high output quality when implementing impulse response filters [4], [5] and training neural networks (NNs) [6]–[8].

Approximate computing has emerged as a low power technique by exploiting the inherent error resiliency in applications such as digital signal processing and NNs [9], [10]. As a fundamental arithmetic operation, multiplication often dominates the performance of circuits and systems. Therefore, designs of approximate multipliers have extensively been investigated, including logarithmic multipliers (LMs) [11].

Without using a complex procedure, the base-2 logarithmic arithmetic converts multiplication to simple additions and bit-shifting, therefore achieving a significant improvement in hardware efficiency [12], [13]. In this paper, we review both FxP and FP LMs from different design perspectives, as well as their error tolerant applications.

The remainder of this paper is organized as follows. Section II presents the related work. LMs are reviewed in Section III. Sections IV summarizes the applications. Section V concludes this paper and discusses future prospects.

This work was partly supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (RES0048688 and RES0054326), MITACS (RES0058253), and Alberta Innovates (RES0053965). T. Zhang is supported by a Ph.D. scholarship from the China Scholarship Council (CSC).

978-1-6654-5707-1/22/$31.00 ©2022 IEEE

II. Related Work

Approximate multipliers have extensively been studied to achieve a good trade-off between accuracy and hardware efficiency for error tolerant applications [11]. The accuracy is evaluated by using a number of error metrics [14], such as the error rate, the error distance, and the root-mean-square error.

Approximate non-logarithmic multiplier designs mainly focus on the simplification in partial product (PP) generation and accumulation for signed and unsigned multiplications. Approximation in the PP generation, for example, includes the use of an approximate 2×2 multiplier to construct larger unsigned multipliers [15], [16], approximate encoding to reduce the delay of calculating the triple multiplicands for the radix-8 Booth encoding algorithm [17], [18], and approximate encoders for the radix-4 Booth encoding algorithm [19]–[21]. Approximation in the PP accumulation includes the use of truncation [22], approximate PP trees [23], [24], and approximate compression using approximate adders [25], [26] or compressors [27]–[30].

LMs deliver a mathematically elegant solution by replacing multiplication with addition and shifting. Mitchell first proposed a simple approximation method for binary logarithmic conversion [31]. However, its error characteristic of always underestimating the product leads to a dramatic error accumulation in circuits and systems [11]. Thus, more hardware-efficient LM designs have recently been pursued for a higher accuracy in intensive computations [32]–[34].

III. Logarithmic Multiplier Designs

A. Preliminaries

Let N be an m-bit number in the binary representation, $N = \sum_{i=0}^{m-1} 2^i n_i$, where n_i denotes the bit value at the ith position. Let k denotes the position of the leading '1' bit, we obtain $N = 2^k + \sum_{i=0}^{k-1} 2^i n_i$. The binary (base-2) logarithm of N is given by:

$$\log_2 N = \log_2(2^k(1+x)) = k + \log_2(1+x), \qquad (1)$$

where x ($= \sum_{i=0}^{k-1} 2^{i-k} n_i$ and $0 \leq x < 1$)) represents the fractional part and k indicates the exponent. If $P = AB$, $\log_2 P$ is computed as:

$$\log_2 P = k_A + \log_2(1+x_A) + k_B + \log_2(1+x_B), \qquad (2)$$

where the exponents and fractional parts of A and B are respectively denoted with the corresponding subscripts. Mitchell's method approximates $\log_2(1+x)$ by x, leading to [31]:

$$\log_2 P \cong k_A + k_B + x_A + x_B. \tag{3}$$

The final product P is obtained by using an antilogarithm function to compute $2^{\log_2 P}$. In Mitchell's method, P is approximately given by [31]:

$$P \cong \begin{cases} 2^{k_A+k_B}(1+x_A+x_B), & x_A+x_B<1, \\ 2^{k_A+k_B+1}(x_A+x_B), & x_A+x_B \geq 1. \end{cases} \tag{4}$$

The basic architecture of an FxP/FP LM is presented in Fig. 1. It is implemented in three stages: (1) the logarithmic conversion of the input operands, (2) the addition of the logarithms, and (3) computing the antilogarithm and converting it to the standard number representation. Consider signed FxP multiplication in two's complement; a signed to unsigned (S2U) converter is particularly used to convert the inputs at Stage 1. Different from an FP LM, an FxP LM requires a leading one detector (LOD) and a priority encoder (PE) to obtain the mantissa and the exponent for the logarithm conversion. The LOD finds the most significant '1' in each input operand, whose bit position is then obtained by the PE. The log converter implements a logarithm approximation. At Stage 2, the multiplication is performed in the logarithm domain by using two adders. At Stage 3, the sum of two logarithms is approximately converted to its antilogarithmic value. The results are then decoded to obtain the final product. Lastly, the signs of the inputs, s_A and s_B, are XOR-ed to obtain the sign of the product P.

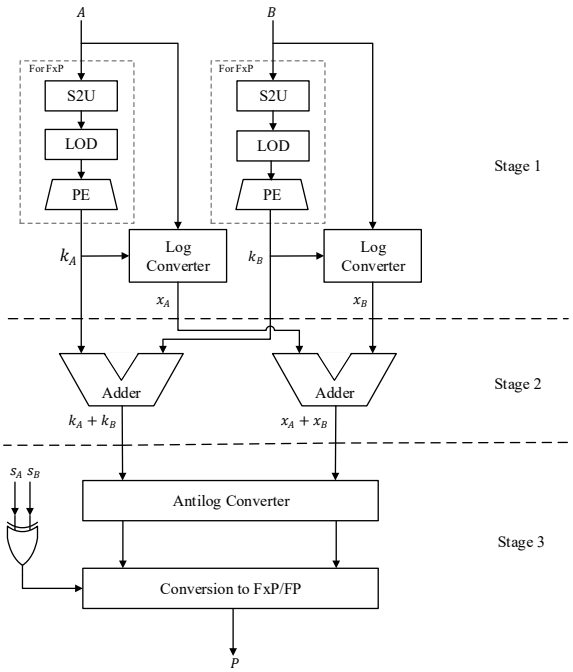

Fig. 1: A Logarithmic Multiplier Architecture.

B. A Review

In this section, LMs are reviewed from different perspectives in the design process.

1) Computing Architecture: The iterative LM (ILM) improves the accuracy of Mitchell's LM by compensating errors through an iterative procedure [32]. The general architecture for an ILM is shown in Fig. 2. The basic block computes the approximate product of two inputs using the LM, followed by an error term calculator (ETC) to generate two errors. Then, the LM serves as the error compensation block (ECB) by multiplying these two errors to obtain the error compensation term, which is then added to the approximate product as the final product. The use of additional ECBs leads to a higher accuracy at a cost of a larger circuit.

To reduce the critical path delay in the basic block, pipelining is used to implement the ILM in [32] for a higher level of parallelism. However, for a higher performance, the ILM neglects the comparison between x_A+x_B and 1 in (4), thus at a cost of accuracy. The truncated ILM in [35] compensates this error by considering the comparison prior to the completion of the current iteration and utilizes truncation to reduce hardware. A low-cost two-stage ILM further compensates errors in the addition and uses a truncated LM [36].

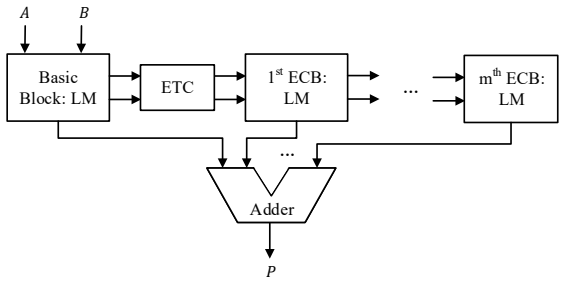

Fig. 2: An Iterative Logarithmic Multiplier Architecture.

2) Input Preprocessing: Some LM designs first preprocess the input operands to simplify the circuit or to improve the accuracy of Mitchell's logarithmic approximation.

a) Hardware-oriented: As a common technique, truncation has been considered in iterative and noniterative LMs to generate smaller bit-width operands for processing [35]–[39]. Especially, Yin et al. proposed a dynamic range LM for signed and unsigned multiplication by dynamically truncating input bits and setting the least significant bit of the truncated operand to '1' to compensate for the negative errors introduced by Mitchell's approximation [38]. Pilipović et al. split the input operands into less and more significant sections, and trimmed the less significant bits when the more significant section contains at least one non-zero bit; otherwise, the more significant section is trimmed [40]. Instead of using the S2U units, Kim et al. approximated two's complement with one's complement for signed multiplication to reduce computation complexity [39].

b) Accuracy-oriented: Mahalingam and Ranganathan decomposed the two input operands to four for decreasing the

probability that a bit '1' occurs in the decomposed inputs, thus reducing the switching power [41]. The hybrid-encoding LMs developed in [42] and [43] first decompose the input operands into two sections, and then apply the logarithm-based and radix-4 Booth multiplication, respectively, to generate less and more significant bits in the product.

3) LOD Designs: In FxP LMs, the LOD unit at Stage 1 accounts for about 49% and 54% of the area and energy, respectively [44]. It detects the left-most '1' and generates a one-hot encoded output. For hardware efficiency, the LM in [45] achieves a $3\times$ speed up by first grouping the input bits to sets of four, and then using OR gates and 4-bit LODs to detect the leading '1'. Approximate 32-bit LOD designs are constructed by 4-bit LOD units in [44] by using two approaches. The first approach is to approximate 4-bit LODs with a single fixed bias or dynamic biases determined by a PE-controlled multiplexer. The second is to approximately simplify a 32-bit LOD design by using an exact 16-bit LOD. It partitions the 32-bit inputs into two parts. The more significant half is processed by using an exact 16-bit LOD if it contains at least one '1'; otherwise, the less significant half is processed. To reduce large errors for small inputs, a scaling scheme is further adopted by using bit-shift operations [44].

4) Logarithm Conversion: Mitchell's method generates a single-sided error distribution that leads to error accumulation. To produce a double-sided error distribution, a nearest-one logarithmic approximation finds the nearest power of two for N [33]. When $N - 2^k < 2^{k+1} - N$, it uses the same logarithm as Mitchell's method; otherwise, N is represented as $N = 2^{k+1}(1 - y)$, where $0 \leq y < 1$, and the logarithm is approximated by $\log_2 N \cong k + 1 - y$. In a circuit implementation, instead of using LODs, a nearest-one detector (NOD) was designed to detect the nearest power of two. Inspired by this method, the logarithmic design in [34] generates a double-sided error distribution for FP LMs.

5) Addition Units: To further improve hardware efficiency, approximate designs have been considered for the mantissa addition, including three types of approximate adders utilized in [46]: the lower-part-OR adder (LOA) [47], the approximate mirror adder-A3 (MAA3) [48] and the set-one adder (SOA) [46]. The LOA calculates less significant bits by using OR gates. In the MAA3, one of the two inputs is approximately considered as less significant bits in the sum for addition. The SOA sets the less significant bits to '1's. Ansari et al. used a modified SOA, which sets the less significant bits as alternating '1's and '0's [33], [44].

IV. Applications

LMs have been considered to improve the hardware efficiency of error tolerant systems, such as those for image processing and machine learning.

Applications in image processing include multiplication [43], [46], sharpening [37], [42], compression [43], smoothing [40], [43], [49], and matching [32]. Two main evaluation metrics for image processing are the peak signal-to-noise ratio and the structural similarity. Compared with exact multiplication, the 16-bit signed FxP LM in [43] produces a similar output for image multiplication and smoothing, and results in a slight quality loss for image compression.

For NN-based applications, inference is less sensitive to precision loss, so FxP LMs are usually applied for inference [33], [36], [38]–[40], [42], [44]. The use of a 16-bit LM for convolutional NN-based classification on MNIST, CIFAR-10, and ImageNet ILSVRC2012 datasets achieves nearly the same accuracy as that by using the exact design [36]. Interestingly, compared to using the exact FxP multiplier, the use of an 8-bit FxP LM increases the classification accuracy of the AlexNet with a smaller PDP [33]. Due to a wider range of representation, hardware-efficient FP multipliers have been used for the training of NNs [34], [50]. The FP LM was evaluated for classifying MNIST at four precision levels. It slightly improves the classification accuracy and uses less energy for implementing a neuron in the single-precision representation [34].

K-means clustering is an unsupervised machine learning algorithm, which is used to partition data points into groups. LMs can be applied to calculate the squared deviation between points belonging to different clusters [38], [46]. The F-measure value is commonly used to evaluate the quality of clustering. Compared to the exact FxP multiplier, a 16-bit FxP LM provides better F-measure values for the selected University of California Irvine (UCI) benchmark datasets [46]. A dynamic range LM leads to similar clustering results [38].

V. Conclusions and Prospects

Logarithmic multipliers significantly improve the hardware efficiency of error-tolerant applications by converting multiplication to addition and shifting operations. In this paper, logarithmic multipliers are briefly reviewed from different design perspectives.

Compared with non-logarithmic approximate multipliers, Mitchell's logarithmic multiplier benefits from its simple structure, but with a large accuracy loss due to the underestimated product. Aimed for an optimized accuracy, the iterative logarithmic algorithm suffers from a large circuit area and a long delay required for error compensation. Hybrid logarithmic multipliers that collaboratively use Mitchell's approximation and a more accurate computation method for respectively encoding less and more significant input bits, worth further investigation for a better trade-off between hardware and accuracy. Moreover, dedicated approximate circuits can be designed to compensate errors due to the logarithmic approximation for improvements in both hardware and accuracy.

For emerging NN applications, a state-of-the-art 4-bit training strategy based on a logarithmic radix-4 format shows a great potential of using the LNS with a small bit width for a significant hardware improvement. There is also some evidence that the use of logarithmic multipliers is likely to improve accuracy for classification. However, it is a challenge to generalize this result. An analysis on the relationship between the error characteristics of LMs and the various features of

NN applications may help understand this important issue. The verification and test of LMs remain open for future research.

REFERENCES

[1] N. G. Kingsbury and P. J. Rayner, "Digital filtering using logarithmic arithmetic," *Electron. Lett.*, vol. 7, no. 2, pp. 56–58, 1971.

[2] B. Parhami, "Computing with logarithmic number system arithmetic: Implementation methods and performance benefits," *Comput. Electr. Eng.*, vol. 87, p. 106800, 2020.

[3] V. Paliouras and T. Stouraitis, "Logarithmic number system for low-power arithmetic," in *PATMOS*, 2000, pp. 285–294.

[4] C. Basetas, I. Kouretas, and V. Paliouras, "Low-power digital filtering based on the logarithmic number system," in *PATMOS*, 2007, pp. 546–555.

[5] S. A. Alam and O. Gustafsson, "Design of finite word length linear-phase FIR filters in the logarithmic number system domain," *VLSI Des.*, vol. 2014, 2014.

[6] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal *et al.*, "Ultra-low precision 4-bit training of deep neural networks," *NIPS*, vol. 33, pp. 1796–1807, 2020.

[7] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani *et al.*, "Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks," *NIPS*, vol. 32, 2019.

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

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

[10] M. Traiola, A. Virazel, P. Girard, M. Barbareschi, and A. Bosio, "A survey of testing techniques for approximate integrated circuits," *Proc. IEEE*, vol. 108, no. 12, pp. 2178–2194, 2020.

[11] 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.

[12] V. Paliouras and T. Stouraitis, "Low-power properties of the logarithmic number system," in *ARITH*, 2001, pp. 229–236.

[13] O. Gustafsson and N. Hellman, "Approximate floating-point operations with integer units by processing in the logarithmic domain," in *ARITH*, 2021, pp. 45–52.

[14] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *TC*, vol. 62, no. 9, pp. 1760–1771, 2012.

[15] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSID*, 2011, pp. 346–351.

[16] W. Liu, T. Zhang, E. McLarnon, M. O'Neill, P. Montuschi, and F. Lombardi, "Design and analysis of majority logic-based approximate adders and multipliers," *TETC*, vol. 9, no. 3, pp. 1609–1624, 2019.

[17] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 Booth multipliers for low-power and high-performance operation," *TC*, vol. 65, no. 8, pp. 2638–2644, 2015.

[18] H. Waris, C. Wang, and W. Liu, "Hybrid low radix encoding-based approximate Booth multipliers," *TCAS II*, vol. 67, no. 12, pp. 3367–3371, 2020.

[19] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 Booth multipliers for error-tolerant computing," *TC*, vol. 66, no. 8, pp. 1435–1441, 2017.

[20] W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. Swartzlander, and F. Lombardi, "Design and analysis of approximate redundant binary multipliers," *TC*, vol. 68, no. 6, pp. 804–819, 2018.

[21] T. Zhang, H. Jiang, H. Mo, W. Liu, F. Lombardi, L. Liu, and J. Han, "Design of majority logic-based approximate Booth multipliers for error-tolerant applications," *TNANO*, vol. 21, pp. 81–89, 2022.

[22] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *TVLSI*, vol. 24, no. 10, pp. 3105–3117, 2016.

[23] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power-and area-efficient approximate wallace tree multiplier for error-resilient systems," in *ISQED*, 2014, pp. 263–269.

[24] S. Hashemi, R. I. Bahar, and S. Reda, "Drum: A dynamic range unbiased multiplier for approximate applications," in *ICCAD*, 2015, pp. 418–425.

[25] H. Jiang, C. Liu, F. Lombardi, and J. Han, "Low-power approximate unsigned multipliers with configurable error recovery," *TCAS I*, vol. 66, no. 1, pp. 189–202, 2018.

[26] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *DATE*, 2014, pp. 1–4.

[27] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *TC*, vol. 64, no. 4, pp. 984–994, 2014.

[28] S. Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," *TVLSI*, vol. 25, no. 5, pp. 1782–1786, 2017.

[29] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "Low-power approximate multipliers using encoded partial products and approximate compressors," *JETC*, vol. 8, no. 3, pp. 404–416, 2018.

[30] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," *TCAS I*, vol. 65, no. 12, pp. 4169–4182, 2018.

[31] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, no. 4, pp. 512–517, 1962.

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

[33] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *TC*, vol. 70, no. 4, pp. 614–625, 2020.

[34] 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," in *GLSVLSI*, 2021, pp. 65–70.

[35] S. E. Ahmed, S. Kadam, and M. Srinivas, "An iterative logarithmic multiplier with improved precision," in *ARITH*, 2016, pp. 104–111.

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

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

[38] 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," *TSUSC*, vol. 6, no. 4, pp. 612–625, 2020.

[39] M. S. Kim, A. A. Del Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell's approximate log multipliers for convolutional neural networks," *TC*, vol. 68, no. 5, pp. 660–675, 2018.

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

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

[42] R. Pilipović and P. Bulić, "On the design of logarithmic multiplier using radix-4 Booth encoding," *IEEE Access*, vol. 8, pp. 64 578–64 590, 2020.

[43] U. Lotrič, R. Pilipović, and P. Bulić, "A hybrid radix-4 and approximate logarithmic multiplier for energy efficient image processing," *Electron.*, vol. 10, no. 10, p. 1175, 2021.

[44] M. S. Ansari, S. Gandhi, B. F. Cockburn, and J. Han, "Fast and low-power leading-one detectors for energy-efficient logarithmic computing," *IET Comput. Digit. Tech.*, vol. 15, no. 4, pp. 241–250, 2021.

[45] K. H. Abed and R. E. Siferd, "VLSI implementations of low-power leading-one detector circuits," in *SECON*, 2006, pp. 279–284.

[46] 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," *TCAS I*, vol. 65, no. 9, pp. 2856–2868, 2018.

[47] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *TCAS I*, vol. 57, no. 4, pp. 850–862, 2009.

[48] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *TCAD*, vol. 32, no. 1, pp. 124–137, 2012.

[49] D. Nandan, J. Kanungo, and A. Mahajan, "An error-efficient Gaussian filter for image processing by using the expanded operand decomposition logarithm multiplication," *JAIHC*, pp. 1–8, 2018.

[50] 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.