



# **International Journal of Engineering Research and Science & Technology**

**[www.ijerst.org](http://www.ijerst.org)**

**ISSN : 2319-5991**

**Vol. 22 No. 1(2) (2026)**



**[ijerst.editor@gmail.com](mailto:ijerst.editor@gmail.com)  
[editor@ijerst.com](mailto:editor@ijerst.com)**

## Research Paper

# Design Of Low-Cost And High-Accurate 8-Bit Logarithmic Floating-Point Arithmetic Circuits

Mr. Dr. U. Sreenivasulu<sup>1</sup>, P. Jhansi<sup>2</sup>, B. Manoj<sup>3</sup>, CH. Rajesh<sup>4</sup>, P. Mallikarjuna Reddy<sup>5</sup>

<sup>1</sup>ASSOCIATE PROFESSOR, <sup>2,3,4,5</sup>STUDENTS

Department Of Electronics and Communication Engineering,

N.B.K.R INSTITUTE OF SCIENCE AND TECHNOLOGY

## ABSTRACT:

This thesis presents the design and implementation of a low-cost and high-accuracy 8-bit Logarithmic Floating-Point (LFP) arithmetic unit, intended for efficient computation in low-power and high-speed embedded systems. Conventional IEEE-754 floating-point units (FPUs) involve high hardware cost due to the need for normalization, rounding, and complex multiplier hardware. In contrast, the proposed LFP architecture performs operations in the logarithmic domain, replacing multiplication and division with addition and subtraction operations. The complete design is modelled in Verilog HDL and synthesised using the Xilinx Vivado tool. Simulation results demonstrate up to 25% delay reduction and 20% area savings when compared to existing architectures. This work demonstrates that logarithmic arithmetic can significantly enhance computational efficiency without substantial accuracy loss, thereby making it highly suitable for next-generation IoT and DSP processors.

## I. INTRODUCTION

Recent research proves that most parameters of the deep learning models fall in low-value bins. This distribution of weights and activations could lead to an ineffective usage of the dynamic range for the quantized models based on the low bit-width fixed-point (FXP) format (e.g., INT8) [1] and more accuracy loss [2]. In addition, the hardware cost gap between the floating-point (FP) arithmetic units and FXP arithmetic units are narrowed with the reduced bit-width. For example, the area of FP and FXP arithmetic units, consisting of adders, multipliers, and multiply-accumulate (MAC) units across different bit-widths using TSMC 7 nm technology, are compared in [3]. As shown in Fig. 1, compared with the FXP arithmetic units, the FP multiplier and FP divider require less area

and the FP adder requires more area. In the case of 8-bit, the area for FP8 and INT8 MAC units are almost identical. This observation paves the way for quantizing deep learning models as the low bit-width FP formats (e.g., FP8), which can be efficiently supported with NVIDIA H100 (E4M3: 4-bit exponent, 3-bit mantissa; and E5M2) [4] and Intel Gaudi2 (E4M3, E5M2) [5]. On the other hand, the FPGA design of the low bit-width FP multipliers is also studied. For example, Xilinx uses  $9.5 \times$  LUT6s to perform a 7-bit FP multiplication (E3M3) [6]. Wu etc. uses  $1 \times$  DSP and  $20 \times$  LUT6s to calculate  $4 \times$  FP multiplications (E3M4) [7].

In addition to above standard 8-bit FP formats (the size of mantissa is different), academia also proposes many novel low bit-width FP formats. For example, the block minifloat (BM), the adaptive floating-point (AdaptFloat), the fixed-length adaptive numerical data type (ANT), the self-gating float (SG-Float), and the dynamic bit (DyBit)-precision formats are proposed in [8], [9], [10], [11], and [12], respectively. However, these methods require to adjust the data format dynamically according to the distribution of model parameters. Therefore, this brief prefers to use standard 8-bit FP format. In addition, the normalized Top-1 accuracy of ResNet-50 with different FP formats in [6] proves that E3M4 achieves the best model accuracy. Three different FP8 formats (E5M2, E4M3, and E3M4) for deep learning applications are carefully discussed in [13] and [17] that suggest E4M3 is suited for the natural language processing model and E3M4 is better on computer vision task. Therefore, this brief focuses on the ASIC design of arithmetic units based on the 8-bit FP formats (e.g., E4M3 and E3M4).

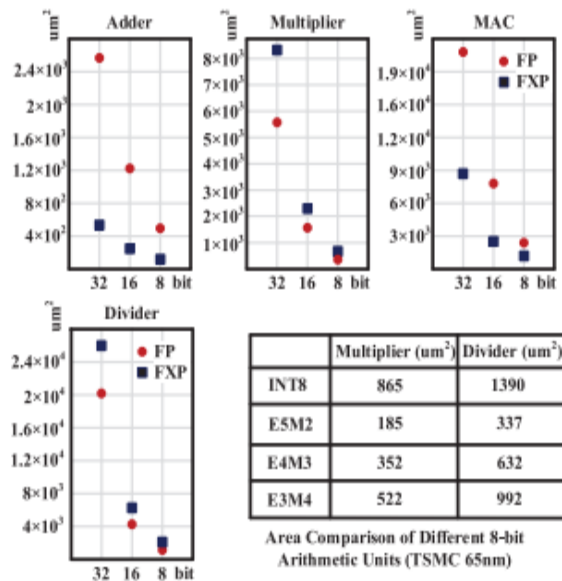


Fig. 1. Area comparison (TSMC 65 nm) of different FP and FXP arithmetic units across different bit-widths, where FP is E5M2 format.

The logarithmic number system (LNS) not only performs the multiplication or division with addition or subtraction but also simplifies the exponential functions [14]. The challenge is the hardware implementation of the logarithmic (Log) and anti-logarithmic (Antilog) functions that generally are realized by the piecewise approximation methods because the Look-up table methods and digital recurrence methods suffer from the large area and long delay, respectively [15], [16]. However, this brief demonstrates that the Log and Antilog converters can be implemented with a low-cost method in the case of the 8-bit FP format. The contributions of this brief are as follows.

- 1) In accordance with the truth table of Log and Antilog converters, this brief finds the 3-bit and 4-bit Log and Antilog converters can be efficiently realized by  $\{x, x+1\}$  and  $\{x, x-1\}$ , respectively, which can significantly simplify the design of FP arithmetic units based on E3M4 and E4M3 formats.

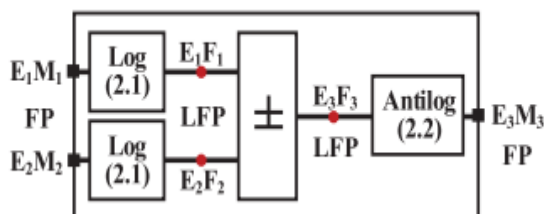


Fig. 2. Multiplication and division in the logarithmic domain.

- 2) Compared to the standard E3M4 and E4M3 multipliers, the proposed logarithmic FP (LFP) multipliers can reduce the area by 40% and 32%, and reduce the delay by 24% and 4%. In addition, compared to the standard E3M4 and E4M3 dividers, the proposed LFP dividers can reduce the area by 67% and 61%, and reduce the delay by 79% and 68%.

- 3) The proposed LFP multiplier (E3M4) can be used in the convolution core. The cell area of proposed method is slightly larger than the INT6-based design, and reduces the cell area by 33% compared to the INT8-based design. On the other hand, the accuracy loss of quantized ResNet-50, MobileNet, and ViT-B are  $-0.12\%$ ,  $0.38\%$ , and  $0.8\%$ , which are better than the INT8 and E4M3-based design.

- 4) The proposed LFP divider also can be used in the image change detection. Compared with the FP divider (E3M4), the false rate is slightly reduced from  $2.97\%$  to  $2.95\%$ .

The rest of this brief is organized as follows. Section II introduces the design methodology of 3-bit and 4-bit Log and Antilog converters. Section III discusses the hardware design of multipliers, dividers, convolution core, and image change detector using the proposed Log and Antilog converters. The comparison results prove the advantages of proposed methods.

## II. RELATED WORK

### “FP8 quantization: The power of the exponent,”

A. Kuzmin, M. Van Baalen, Y. Ren, M. Nagel, J. Peters, and T. Blankevoort.

When quantizing neural networks for efficient inference, low-bit integers are the go-to format for efficiency. However, low-bit floating point numbers have an extra degree of freedom, assigning some bits to work on an exponential scale instead. This paper in-depth investigates this benefit of the floating point format for neural network inference. We detail the choices that can be made for the FP8 format, including the important choice of the number of bits for the mantissa and exponent, and show analytically in which settings these choices give better performance. Then we show how these findings translate to real networks, provide an efficient implementation for FP8 simulation, and a new algorithm that enables the

learning of both the scale parameters and the number of exponent bits in the FP8 format. Our chief conclusion is that when doing post-training quantization for a wide range of networks, the FP8 format is better than INT8 in terms of accuracy, and the choice of the number of exponent bits is driven by the severity of outliers in the network. We also conduct experiments with quantization-aware training where the difference in formats disappears as the network is trained to reduce the effect of outliers.

#### **“Low-precision floatingpoint arithmetic for high-performance FPGA-based CNN acceleration”**

C. Wu, M. Wang, X. Chu, K. Wang, and L. He.

Low precision data representation is important to reduce storage size and memory access for convolutional neural networks (CNNs). Yet, existing methods have two major limitations: (1) requiring re-training to maintain accuracy for deep CNNs, and (2) needing 16-bit floating-point or 8-bit fixed-point for a good accuracy. In this paper, we propose a low precision (8-bit) floating-point (LPFP) quantization method for FPGA-based acceleration to overcome the above limitations. Without any re-training, LPFP finds an optimal 8-bit data representation with negligible top-1/top-5 accuracy loss (within 0.5%/0.3% in our experiments, respectively, and significantly better than existing methods for deep CNNs). Furthermore, we implement one 8-bit LPFP multiplication by one 4-bit multiply-adder (MAC) and one 3-bit adder, and therefore implement four 8-bit LPFP multiplications using one DSP slice of Xilinx Kintex 7 family (KC705 in this paper) while one DSP can implement only two 8-bit fixed-point multiplications. Experiments on six typical CNNs for inference show that on average, we improve throughput by 64.5x over Intel i9 CPU and by 1.5x over existing FPGA accelerators. Particularly for VGG16 and YOLO, compared to six recent FPGA accelerators, we improve average throughput by 3.5x and 27.5x and improve average throughput per DSP by 4.1x and 5x, respectively. To the best of our knowledge, this is the first in-depth study to simplify one multiplication for CNN inference to one 4-bit MAC and implement four multiplications within one DSP while maintaining comparable accuracy without any re-training.

#### **“High-performance FPGA-based CNN accelerator with block-floating-point arithmetic,”**

X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji.

Convolutional neural networks (CNNs) are widely used and have achieved great success in computer

vision and speech processing applications. However, deploying the large-scale CNN model in the embedded system is subject to the constraints of computation and memory. An optimized block-floating-point (BFP) arithmetic is adopted in our accelerator for efficient inference of deep neural networks in this paper. The feature maps and model parameters are represented in 16-bit and 8-bit formats, respectively, in the off-chip memory, which can reduce memory and off-chip bandwidth requirements by 50% and 75% compared to the 32-bit FP counterpart. The proposed 8-bit BFP arithmetic with optimized rounding and shifting-operation-based quantization schemes improves the energy and hardware efficiency by three times. One CNN model can be deployed in our accelerator without retraining at the cost of an accuracy loss of not more than 0.12%. The proposed reconfigurable accelerator with three parallelism dimensions, ping-pong off-chip DDR3 memory access, and an optimized on-chip buffer group is implemented on the Xilinx VC709 evaluation board. Our accelerator achieves a performance of 760.83 GOP/s and 82.88 GOP/s/W under a 200-MHz working frequency, significantly outperforming previous accelerators.

#### **“Algorithm-hardware co-design of adaptive floatingpoint encodings for resilient deep learning inference,”**

T. Tambe et al.

Conventional hardware-friendly quantization methods, such as fixed-point or integer, tend to perform poorly at very low precision as their shrunken dynamic ranges cannot adequately capture the wide data distributions commonly seen in sequence transduction models. We present an algorithm-hardware co-design centered around a novel floating-point inspired number format, AdaptivFloat, that dynamically maximizes and optimally clips its available dynamic range, at a layer granularity, in order to create faithful encodings of neural network parameters. AdaptivFloat consistently produces higher inference accuracies compared to block floating-point, uniform, IEEE-like float or posit encodings at low bit precision ( $\leq 8$ -bit) across a diverse set of state-of-the-art neural networks, exhibiting narrow to wide weight distribution. Notably, at 4-bit weight precision, only a 2.1 degradation in BLEU score is observed on the AdaptivFloat-quantized Transformer network compared to total accuracy loss when encoded in the above-mentioned prominent datatypes. Furthermore, experimental results on a deep neural network (DNN) processing element (PE), exploiting AdaptivFloat logic in its computational datapath, demonstrate per-

operation energy and area that is  $0.9\times$  and  $1.14\times$ , respectively, that of an equivalent bit width NVDLA-like integer-based PE.

### III. PROPOSED METHODOLOGY

#### A. Logarithmic Number Representation

The FP format and the proposed LFP format are defined as

$$\begin{cases} (-1)^S \times 2^{E-BIAS} \times (1 + M), & \text{FP} \\ (-1)^S \times 2^{E-BIAS} \times 2^F, & \text{LFP} \end{cases} \quad (1)$$

where S is sign, E is exponent, and M and F are mantissa of FP and LFP numbers, respectively. In consistent with the FPGA implementation of small FP multipliers proposed by Xilinx [6], the bias of the exponent is  $0.5 \times (\max(E) + 1)$ . For example, the bias is 4 or 8 if the exponent field is 3- or 4-bit. All numbers with an exponent of zero are treated as zero. That is, the subnormal numbers are removed to simplify the hardware cost. For example, a binary number (E3M4: 0-100-1000) in FP and LFP format is  $20 \times (1+0.5)$  and  $20 \times 2^{0.5}$ , respectively. The relationship between F and M in (1) is

$$\begin{cases} F = \text{Log}_2(1 + M), & (2.1) \\ M = 2^F - 1, & (2.2). \end{cases} \quad (2)$$

Fig. 2 shows that two input FP numbers  $\{2^{E1-Bias} \times (1 + M1), 2^{E2-Bias} \times (1 + M2)\}$  should be transformed as two LFP numbers  $\{2^{E1-Bias} \times 2^{F1}, 2^{E2-Bias} \times 2^{F2}\}$  using Log converter. Therefore, the multiplication and division can be easily performed with the addition and subtraction in the logarithmic domain. The result is denoted as  $2^{E3-Bias} \times 2^{F3}$ , which should be transformed as an FP number  $2^{E3-Bias} \times (1 + M3)$ . The highlight of LFP format is that the multiplication and division can be efficiently implemented by the addition and subtraction with the Log (2.1) and Antilog (2.2) converters. In addition, to avoid the potential overflow (is easy to happen for the narrow exponent field), the bit-width of the exponent of the result should be extended by 1-bit that is consistent with [6]. In other words, the bit-width of E3 is larger than the bit-width of E1 and E2 with 1-bit.

TABLE I  
TRUTH TABLE OF 3-BIT LOG AND ANTILOG CONVERTERS

| Input x | Log | $\Delta$ | Antilog | $\Delta$ |
|---------|-----|----------|---------|----------|
| 000     | 000 | 0        | 000     | 0        |
| 001     | 001 | 0        | 001     | 0        |
| 010     | 011 | +1       | 010     | 0        |
| 011     | 100 | +1       | 010     | -1       |
| 100     | 101 | +1       | 011     | -1       |
| 101     | 110 | +1       | 100     | -1       |
| 110     | 110 | 0        | 101     | -1       |
| 111     | 111 | 0        | 111     | 0        |

Another advantage of the proposed method is that the Log or Antilog converter could be removed in some applications that can further reduce the hardware cost and conversion error. For example, for convolution operations in Section III-C, we can remove a Log converter because the weights can be quantized as the LFP format in advance. For image change detection in Section III-D, we can remove the Antilog converter because the result in LFP format is acceptable.

#### B. Logarithmic And Antilog Converter

The challenge of performing multiplication and division in the logarithmic domain is the hardware implementation of the Log and Antilog converters in Fig. 2. To solve this problem, this brief calculates the truth table of the 3- and 4-bit Log and Antilog converters in Tables I and II that provide an efficient method to implement the Log and Antilog converters.

In the case of the 3-bit Log converter, it can be calculated by  $x+1$  if (3.1) is false, where  $x_k$  is the binary digit of  $x[k]$  and  $v$  is used to decide whether the Log converter is realized by  $x$  or  $x+1$ . Similarly, the 3-bit Antilog converter can be calculated by  $x-1$  if (3.2) is false. Similarly, the 4-bit Log and Antilog converters can be calculated by  $x+1$  and  $x-1$  if (3.3) is false. In summary, the Log and Antilog converters can be implemented exactly with small hardware cost.

$$\begin{cases} v = \bar{x}_2 \bar{x}_1 + x_2 x_1, & (3.1) \\ v = \bar{x}_2 \bar{x}_1 + x_2 x_1 x_0 + \bar{x}_2 x_1 \bar{x}_0, & (3.2) \\ v = \bar{x}_3 \bar{x}_2 \bar{x}_1 + x_3 x_2 x_1 x_0, & (3.3). \end{cases} \quad (3)$$

### IV. RESULTS

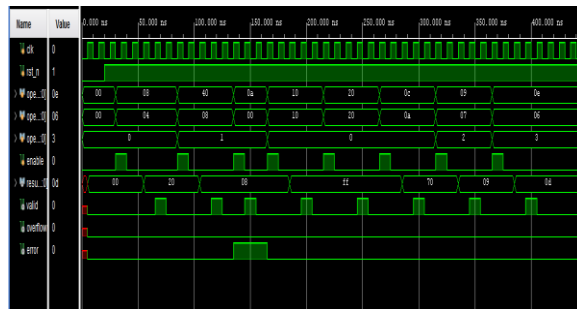


Fig.3. Output waveform

The waveform demonstrates that the proposed 8-bit logarithmic floating-point arithmetic circuit performs accurate arithmetic operations with proper timing control. The design efficiently generates valid outputs with minimal delay, handles errors correctly, and avoids overflow, proving it is suitable for low-cost and high-accuracy applications.

## V. CONCLUSION

This thesis successfully demonstrates the design of a low-cost, high-accuracy 8-bit logarithmic floating-point arithmetic unit using the Vivado design suite. The proposed approach simplifies computation by leveraging logarithmic properties, reduces area and power, and achieves faster performance. Future work includes extending this design for adaptive precision arithmetic and integration into RISC-V soft processors for edge computing applications.

## REFERENCES

- [1] Z. Wang, J. Zhang, and N. Verma, "Reducing quantization errors for inner-product operations in embedded digital signal processing systems [tips&tricks]," *IEEE Signal Process. Mag.*, vol. 33, no. 6, pp. 141–147, Nov. 2016.
- [2] A. Kuzmin, M. Van Baalen, Y. Ren, M. Nagel, J. Peters, and T. Blankevoort, "FP8 quantization: The power of the exponent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 14651–14662.
- [3] Y. Zhang et al., "Integer or floating point? New outlooks for low-bit quantization on large language models," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Niagara Falls, ON, Canada, Jul. 2024, pp. 1–6.
- [4] J. Choquette, "NVIDIA hopper H100 GPU: Scaling performance," *IEEE Micro*, vol. 43, no. 3, pp. 9–17, May/Jun. 2023.

[5] Intel. (2022). Habana Gaudi2 White Paper. [Online]. Available: <https://habana.ai/wp-content/uploads/pdf/2022/gaudi2-whitepaper.pdf>

[6] Higher Performance Neural Networks with Small Floating Point, document WP530(v1.0.1), Xilinx, San Jose, CA, USA, 2021.

[7] C. Wu, M. Wang, X. Chu, K. Wang, and L. He, "Low-precision floatingpoint arithmetic for high-performance FPGA-based CNN acceleration," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 1, pp. 1–21, Mar. 2022.

[8] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji, "High-performance FPGA-based CNN accelerator with block-floating-point arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1874–1885, Aug. 2019.

[9] T. Tambe et al., "Algorithm-hardware co-design of adaptive floatingpoint encodings for resilient deep learning inference," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6.

[10] C. Guo et al., "ANT: Exploiting adaptive numerical data type for low-bit deep neural network quantization," in *Proc. 55th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2022, pp. 1414–1433.

[11] J.-S. Wu, T.-W. Hsu, and R.-S. Liu, "SG-float: Achieving memory access and computing power reduction using self-gating float in CNNs," *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 6, Nov. 2023, Art. no. 101.

[12] J. Zhou et al., "DyBit: Dynamic bit-precision numbers for efficient quantized neural network inference," *IEEE Trans. Comput.- Aided Design Integr. Circuits Syst.*, vol. 43, no. 5, pp. 1613–1617, May 2024.

[13] H. Shen, N. Mellempudi, X. He, Q. Gao, C. Wang, and M. Wang, "Efficient post-training quantization with fp8 formats," *Proc. Mach. Learn. Syst.*, vol. 6, pp. 483–498, May 2024.

[14] M. Gautschi, M. Schaffner, F. K. Gurkaynak, and L. Benini, "A 65 nm CMOS 6.4-to-29.2pJ/FLOP0.8V shared logarithmic floating point unit for acceleration of nonlinear function kernels in a tightly coupled processor cluster," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, Benin, Jan. 2016, pp. 82–83.

- [15] B. Xiong, Y. Li, S. Li, S. Fan, and Y. Chang, “Half-precision logarithmic arithmetic unit based on the fused logarithmic and antilogarithmic converter,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 2, pp. 243–247, Feb. 2022.
- [16] T. Zhang, Z. Niu, and J. Han, “A design framework for hardware efficient logarithmic floating-point multipliers,” *IEEE Trans. Emerg. Topics Comput.*, vol. 12, no. 4, pp. 991–1001, Oct. 2024.
- [17] M. van Baalen et al., “FP8 versus INT8 for efficient deep learning inference,” 2023, arXiv:2303.17951.
- [18] P. Micikevicius et al., “FP8 formats for deep learning,” 2022, arXiv:2209.05433.
- [19] M. N. Sumaiya and R. S. S. Kumari, “Logarithmic mean-based thresholding for SAR image change detection,” *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 11, pp. 1726–1728, Nov. 2016.
- [20] Z. Liu et al., “Post-training quantization for vision transformer,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 28092–28103.