AN ENERGY-EFFICIENT TUNABLE-PRECISION FLOATING-POINT FUSED MULTIPLY-ADD UNIT BASED ON NEURAL NETWORKS

Xiyang Sun, Yue Zhao and Sheng Zhang

Key Laboratory of Advanced Sensor and Integrated System, Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, 518055, China

ABSTRACT

Convolutional neural networks have been continuously updated in the last decade, requiring more diverse floating-point formats for the supported domain specific architectures. We have presented VARFMA, a tunable-precision fused multiply-add architecture based on the Least Bit Reuse structure. VARFMA optimizes the core operation of convolutional neural networks and supports a range of precision that includes the common floating-point formats used widely in enterprises and research communities today. Compared to the latest standard baseline fused multiply-add unit, VARFMA is generally more energy-efficient in supporting multiple formats, achieving up to 28.93% improvement for LeNet with only an 8.04% increase in area. Our design meets the needs of the IoT for high energy efficiency, acceptable area, and data privacy protection for distributed networks.

KEYWORDS

Fused Multiply-add, Tunable-precision, Distributed Network, Energy Efficiency, IoT

1. INTRODUCTION

In the last decade, deep learning (DL) has emerged as a popular field where the training and inference of parameters is achieved through the construction of convolutional neural networks (NN). DL achieved excellent results in domains such as image recognition [1], natural language processing [2], video and audio processing [3]. The concept of deep neural network (DNN) was already proposed by Rosenblat in 1962 [4] but became concrete and feasible when Rumelhart, Hinton, and Williams implemented the backpropagation algorithm (BP) by designing gradient descent in 1986 [5]. Moore's Law and Dennard's Law brought about a rapid development in electronics, which greatly increased the computing power of computers. However, with Moore's Law limiting the performance of general-purpose computers nowadays and Amdahl's Law limiting the performance of computer clusters, general-purpose computers have been unable to meet the growing demand for NN computing power. To solve this problem, many different types of research have emerged. On the software side, many lightweight NNs with sparsity have been proposed to reduce the demand for hardware resources [6]; on the hardware side, domain-specific architectures (DSA) have been popular and hardware modules dedicated to NN acceleration have become mainstream, such as Google’s Tensor Processing Unit (TPU) and Samsung’s Neural-Network Processing Unit (NPU) [7].

David C. Wyld et al. (Eds): ICDIPV, CBIoT, ICAIT, WIMO, NC, CRYPIS, ITCSE, NLCA, CAIML -2023
pp. 159-169, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131312

Before 2019, NN training was mostly deployed on cloud servers. However, to meet the requirement for data privacy in Internet of Things (IoT) domain, NN training is emerging at the distributed edge. Federated Learning techniques are proposed to allow decentralized participants to collaborate on model training for machine learning while satisfying the requirement of not disclosing private data to other participants. This puts a higher demand on the power consumption and area of the computation. Compared to fixed-point numbers, floating-point numbers (FP) have higher precision and a larger representation range. While some strategies can produce more reliable results for fixed-point numbers using deep analysis, these strategies also introduce more code volume to normalize the data. Therefore, FP is needed to provide sufficient precision and range. As shown in (1), FP fused multiply-add (FMA) is the core operation of the local training of NN, denoted as $O = A \times B + C$. Since the IBM RISC System 6000 in 1990 [8], FMA has been incorporated into IMB CELL, PowerPC, ARMv7-A, AMD's Piledriver, Intel's Haswell, and other architectures [9-11].

$$z = \sum_{i=1}^{n} X_i \times w_i + b \tag{1}$$

The standard FP format is no longer able to achieve the optimal requirements of NN, and more of them are being proposed to support arbitrary formats. In this paper, we improve the existing LBR and design the arbitrary tunable precision FP module VARFMA. The main contributions are:

(1) Expand the precision range supported by FMA, add data conversion design, and implement the calculation unit that supports exponent width and mantissa width as variables. The supported formats cover the mainstream FP32, TF32, FP16, BF16, PULP8 and 4M3E8.

(2) The analysis of VARFMA based on different NNs shows an energy efficiency improvement of up to 28.93% compared to the design of Muller et al. [20]. And the energy efficiency ratio of the lower-precision FP to the highest precision FP supported reaches up to 1.81× with an area increase of only 8.04%.

2. RELATED WORK

IEEE754-2008 specifies the format and hardware implementation of the basic functions of the standard FP, which is now widely adopted. The standard FP includes 1-bit sign, E bits exponent and M bits mantissa, as shown in Fig. 1. In the general case, it represents the number of $(-1)^{sign} \times 2^{(exponent-2^{E-1}+1)} \times 1.mantissa$. The expressed number of abnormal cases is explained in detail in the following section.

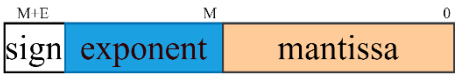

Figure 1. IEEE754 floating point format

FP formats used a lot in the past are the standard single-precision (FP32), double-precision (FP64) and half-precision (FP16). With the rise of DSA field, many unconventional FP formats have been proposed to maximize the performance of the architecture and achieve higher energy efficiency. Companies and research institutes have invented their FP formats, such as Intel's Nervana Flexpoint [5], Microsoft's Brainwave 9-bit FP [12], Google's TPU bfloat (BF16) [13] or NVIDIA's 19-bit Tensor Float (TF32) implementation in Tensor Cores [14]. 8-bit FP formats are

more often found in the academic domain, with ETH Zurich proposing PULP binary8 (PULP8) [15] and Schulte [16] supporting two kinds of 8-bit FPs. Their formats are shown in Fig. 2.

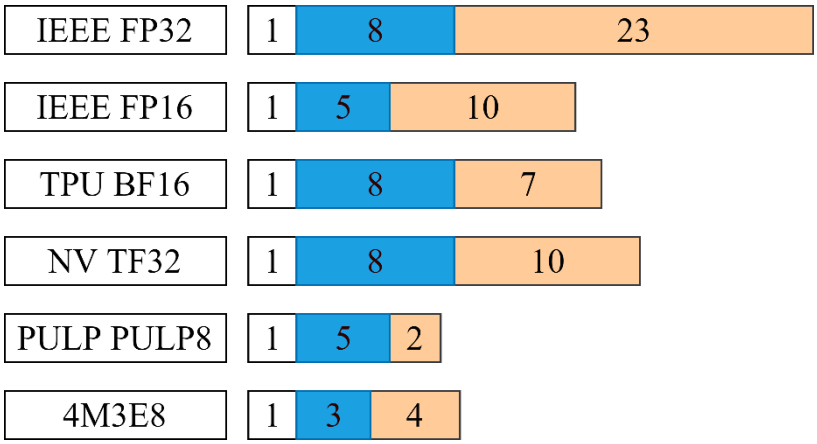

Figure 2. FP formats used in DSA

In order to support more FP formats, the main multi-precision FMAs can be divided into the following three approaches: Spatial Reuse (SR), Spatial Separation (SS), and Least Bit Reuse (LBR). SR combines multiple low-precision data in space by cutting the data path into segments and combining them into high-precision data computations by shifting and control logic. Himanshu Kaul et al. [4] designed a FMA unit that can compute one 24-bit, two 12-bit, or four 6-bit significants. SS applies separate slides to each FP precision and does not use multiplexing strategies in the time or space domain. Stefan Mach et al. [15] adopted a parallel slicing approach for the supported FP formats and achieved an energy efficiency of 74-1245 GFLOPS/W; Ankur Agrawal et al. [17] implemented FP16 and Half Floating Point 8 (HFP8) formats respectively, which include the 4M3E8 format which has 3-bit exponent and 4-bit mantissa, achieving an energy efficiency of 1.95-3.85 TFLOPS/W. LBR spatially multiplexes the lower bits portion of the datapath. Unlike SR, LBR does not support multiple lower-precision formats simultaneously, but treats the exponent and mantissa widths as control signals, and achieves arbitrary precision support by setting the exponent and mantissa widths in real-time. Alberto Nannarelli et al. [18] set the exponent and mantissa widths in the single-precision FP format based on multiplexing multipliers and adders. By adding decoders and quad-multiplexers, it has a smaller area and power consumption compared to separate multiplier and adder. In addition, there is also a time-domain reuse method, which combines low-precision data generated in multiple cycles.

SR is the most commonly used method for fixed-point computation units, but this is not fully applicable to the FP format. The fixed-point operation can be easily implemented by adding up two identical low-precision fixed-point, but Fig. 2 shows that as the overall width of the FP format decreases equally, the exponent part does not decrease in proportion to the mantissa part, which results in a large redundancy in the exponent calculation and makes the control logic of mantissa hard to design. Zhang H et al. [19] implemented the standard binary128 (QP), DP, SP, and HP. In order to align the formats, the mantissa was grouped by 15 bits. The processing of the mantissa required a total of 120 bits and the exponent part required a total of 40 bits. Himanshu Kaul's design completely reused the datapath but did not effectively support the existing FP formats. Moreover, SS cannot support arbitrary FP formats but only compute a few existing formats. In contrast, LBR has moderate area and latency, supports more data formats with built-in exponent width and mantissa width selectors, and is better able to support new DNN formats as they emerge.

3. ARCHITECTURE

This section shows details of VARFMA architecture. In the beginning, the format requirements for the input data and the data conversion method are introduced. Then it shows the modules that makeup VARFMA and the circuit designed to support tunable precision. Finally, the hardware implementation scheme and test set are presented.

3.1. Tunable FP Format Requirement

To achieve arbitrary precision FP support, the design specifies a lower-precision exponent and mantissa data layout when the required precision width is less than the maximum precision width. As shown in Fig. 3, the capital E and M represent the maximum precision supported respectively, which are fixed values; ew and mw represent the actual exponent width and mantissa width of the input, which are variable values. The input data needs to be formatted and brought into the FMA calculation.

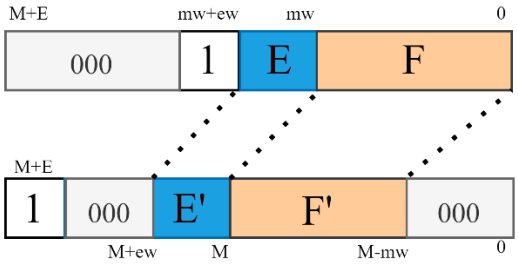

Figure 3. Format conversion of tunable precision FP

From Fig. 2, it can be seen that all of the innovative FP formats are shorter than FP32 in length, so the input FP formats need to be processed. The processing includes: shift the sign bit originally ranked in mw+ew position to M+E bit; shift the exponent in [mw+ew-1,mw] position to [M+E-1,M] and supplement zeros the most bits; shift the mantissa in [mw-1,0] position to [M-1, M-mw] and supplement zeros the least bits.

3.2. Architecture Design

This section describes the design architecture of this paper in detail, focusing on the variable parameter (var_parameter) module and the variable rounding (var_prerounding) module added to support arbitrary precision FP.

Starting from the overall architecture, as shown in Fig. 4, VARFMA receives the system signals, operands a, b, c, and control signals. The datapath mainly consists of the following logic modules: var_parameter module (module 1), preprocessing module (module 2), pre-alignment module (module 3), multiplier (module 4), adder (module 5), normalization module (module 6), Leading Zero Count (LZC) (module 7), var_prerounding module (module 8), rounding and exception check module (module 9) and pipeline logic. The dark modules are added or modified compared to the standard FMA architecture to implement variable precision.

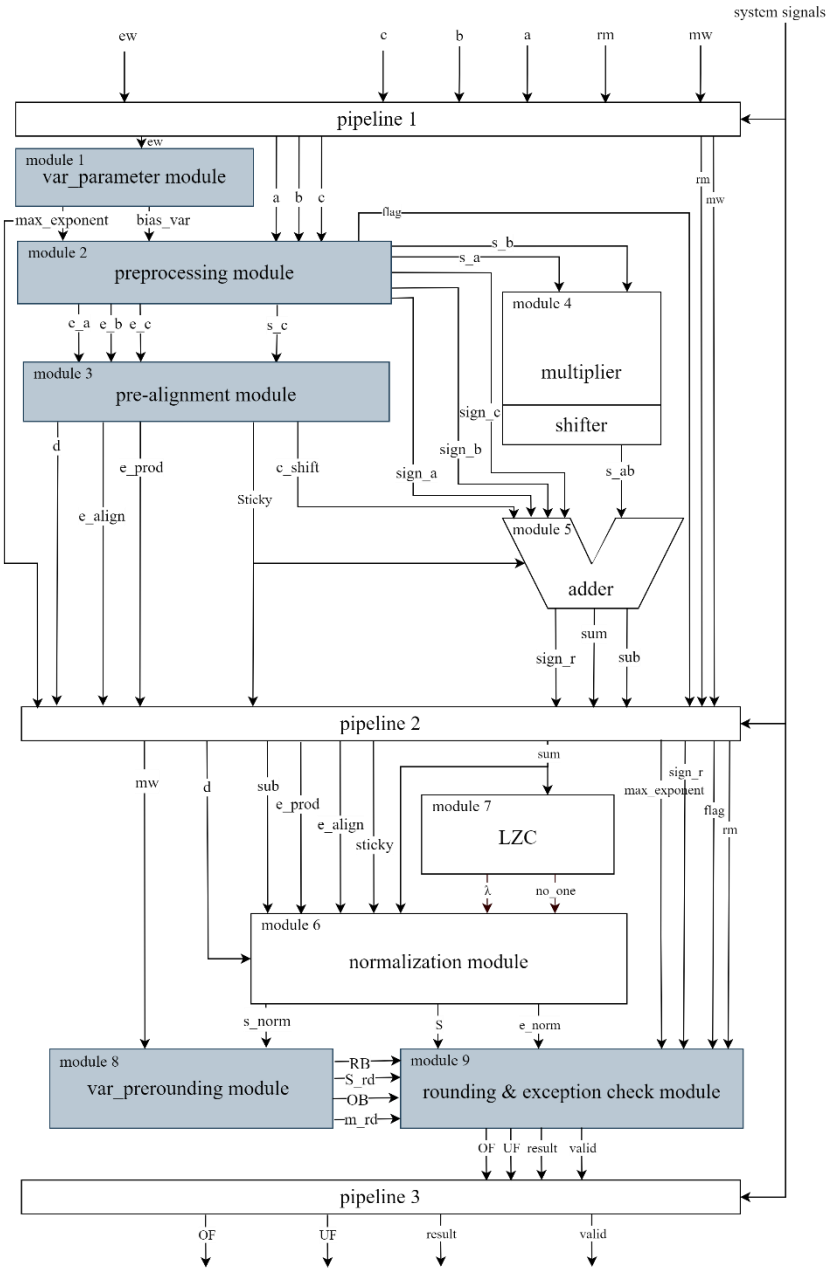

Figure 4. The overall architecture of VARFMA

Since VARFMA supports floating-point formats of arbitrary precision, this means that the top-level input control signal needs to contain two additional input variables: ew and mw. The var_parameter module needs to determine the ew to get the exponent bias (bias_var) and max biased exponent for the FMA operation (max_exponent). As shown in Fig. 5, bias_var can be achieved with parallel comparators and selectors as the maximum bias width (BW). Ew is issued into the module, compared with the index in each BW comparator, and then the result is connected to the corresponding selection side of the selector. The output of a selector is 1 if successfully selected, otherwise is 0. Max_exponent can be compared and selected in a similar way, where the number of selectors and comparators is the maximum exponent width that the FMA module can support.

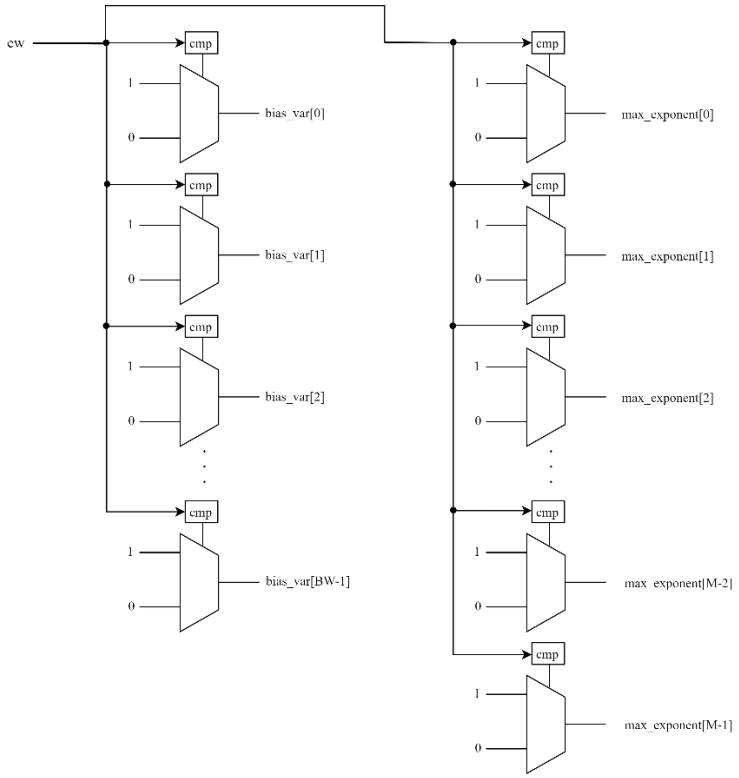

Figure 5. Design of var_parameter module

The preprocessing module predetermines the input operands a, b and c in FP format. It also determines whether the data is in a special case according to IEEE-754 rules, including Infinity, Not a Number (NAN), Zero and Subnormal. The module also turns the mantissa into the significant by attaching a hidden number to the most bit of mantissa. The hidden number is 0 when the exponent is 0, or it will be 1 otherwise. It is important to note that in VARFMA, the comparison object is max_exponent when judging Infinity and NAN.

The pre-alignment module handles the significants of the operands, and the actual length of the significants of operators is p-width, which is obtained from the input mw of the top-level module: . To support FP with arbitrary precision, the input data width of the multiplier and adder is the maximum significant width supported, denoted as P. To reuse the multiplier and adder, it is necessary to make up the full operator's significant into P bits. When shifting the significand, the exponent difference of product and addend is used using bias_var.

The multiplier, adder, normalization module and LZC are consistent with the standard FMA, the multiplier needs to multiply the two P-bit significants to get the 2P-bit product. The adder calculates the sum of pre-aligned product and addend to get the temporary sum and sign bit sign_r. LZC, as the name suggests, it is responsible for counting and getting the number of leading zeros, denoted as λ. The normalization module includes the shift of sum according to λ, the new sticky bit calculation, and the exponent update. It can be seen from (2) that the significant to be processed by the multiplier has a lower data toggle rate when supporting lower precision, thus reducing the computational power consumption.

$$\begin{aligned} MM_{product} &= MM_A \times MM_B = \{MM_a, MM_0\} \times \{MM_b, MM_0\} \\ &= MM_a \times MM_b \ll shift_{2(M-mw)} + MM_a \times MM_0 \ll shift_{M-mw} + MM_0 \times MM_b \ll shift_{M-mw} \end{aligned} \tag{2}$$

The P-bit significant s_norm from the normalization module is not fully needed. Instead, the needed p-bit significant can be obtained according to mw. In the var_prerounding module, as shown in Fig. 6, the s_norm within the p bits is reserved to produce m_rd, the data used later to get an accurate result. The round bit (RB) is selected as the bit after the p-th bit of s_norm by a selector; the odd bit (OB) is selected as the p-th bit of s_norm. The remaining bits are orthogonal to each other to obtain the new sticky bit S_rd. The final sticky bit used in the rounding module is obtained by putting S_rd and the sticky bit S obtained in the pre-alignment module in the OR gate.

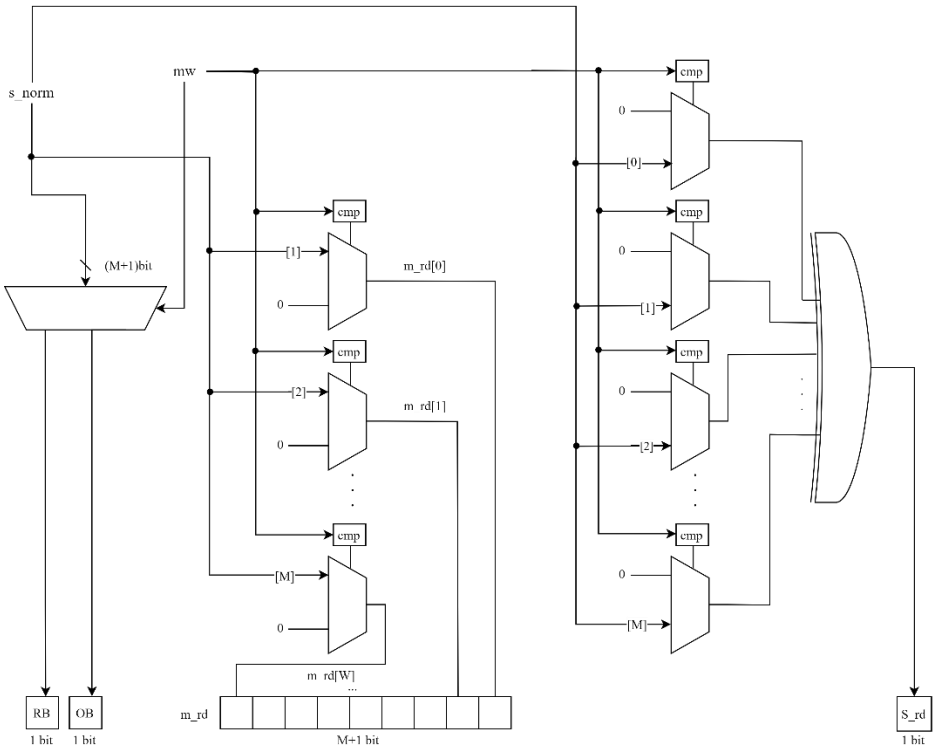

Figure 6. Design of var_prerounding module

Rounding and exception checking use m_rd, e_norm (obtained by the normalization module), final sticky bit, RB, OB and the max_exponent (obtained by the var_paremeter module). Depending on the final sticky bit, RB, OB and round mode, it can be determined whether the result should be rounded or not. The exception checking module checks for overflow and underflow. An overflow exception is generated when the calculated exponent is greater than or equal to max_exponent. Underflow means the calculated exponent is less than or equal to the actual minimum exponent value, which corresponds to Zero and Subnormal.

Because of the complexity of the FP operation datapath, three pipeline logic stages are inserted into the process to shorten the critical path delay.

3.3. Implementation

The VARFMA has been implemented in TSMC 28nm technology of standard cells, using Synopsys tools and targeting 1.3 GHz. To compare the test results, the standard FP32 [20] has been fully implemented under the same conditions.

To evaluate the performance of VARFMA on NNs, two CNNs are chosen for the experimental test set: the middle layer of ResNet50 and LeNet, which typically represent the heavyweight and lightweight CNNs respectively. LeNet is one of the earliest CNN and the starting point of a number of recent NN architectures. The weight-sharing feature of the convolutional layer makes it save considerable computation and memory space compared to the fully connected layer. ResNet50 is one of the most widely used CNN for image classification, and its smaller convolutional kernel and deeper channels are important representatives of DL networks. Convolution calculation is shown in (3), where the activation A and the weight W are summarized in three dimensions: convolution kernel size (ksize), number of input channels (ch), and number of convolution kernels (knum).

$$\sum_{k=1}^{ksize}\sum_{c=1}^{ch}\sum_{i=1}^{knum} A_{k,c,i} \times W_{k,c,i} \tag{3}$$

We have chosen six FP formats: FP32, TF32, FP16, BF16, PULP8, and 4M3E8, as shown in Fig. 2. These are the data formats that have been widely integrated into DSA or evaluated in papers in recent years.

4. IMPLEMENTATION RESULTS

This section explores the area and energy efficiency of VARFMA in different formats, performance improvement for ResNet and LeNet datasets and performance analysis against the standard FP32 FMA described in [20] as a baseline.

Tab. 1 shows the energy efficiency and area of VARFMA compared with FP32 FMA when running a layer of ResNet and LeNet. The energy efficiency ratio of different FP formats to FP32 is shown in the last column of table using the same FMA in the same network. The data were measured at 1.3 GHz.

Table 1. Comparison of this work with the baseline architecture

Design	Area (cells)	Precision	Energy Efficiency (GFLOPS/W)		Ratio	
			LeNet	ResNet	LeNet	ResNet
This work	5859	FP32	1770.635	1826.356	1.00×	1.00×
		TF32	2349.115	2362.777	1.33×	1.29×
		FP16	2354.646	2384.009	1.33×	1.31×
		TF16	2617.274	2614.642	1.48×	1.43×
		PULP8	**3201.182**	3143.894	**1.81**×	1.72×
		4M3E8	2880.567	2996.773	1.63×	1.64×
Muller et al. [20]	5423	FP32	1814.882	2009.895	1.00×	1.00×
		TF32	2131.497	2264.020	1.17×	1.13×
		FP16	2131.497	2264.020	1.17×	1.13×
		TF16	2261.263	2358.063	1.25×	1.17×
		PULP8	2482.811	2523.782	1.37×	1.26×
		4M3E8	2381.389	2445.908	1.31×	1.22×

As shown in tab. 1, the addition of var_parameter mode and var_prerounding mode results in a larger total design area, with an 8.04% increase in area compared to the baseline.

Next, we analyze the energy efficiency improvement in VARFMA in different formats. As the width of the supported precision decreases, the overall energy efficiency of VARFMA shows a linear increase, reaching the maximum improvement in the PULP8 format, 80.8% for LeNet and 72.1% for ResNet respectively. Since the FP FMA exponent and mantissa computations are mostly separated, the mantissa is in a longer and more complex datapath than the exponent, which is on the critical path of the computation. And since the FMA calculation involves the difference of exponent of operands, the mantissa is needed to be shifted, resulting in the width of this datapath being three times wider than the original mantissa width. For these reasons, the calculation of mantissa causes major power consumption. From the above analysis, it can be understood that the energy efficiency of formance than 4M3E8.

Comparing running LeNet with ResNet, it is obvious that the ResNet energy efficiency is generally higher than LeNet. Except when running TF16 and PULP8 on VARFMA where ResNet energy efficiency is up to 1.79% lower, the performance is generally 4.03%-10.75% higher for ResNet in the remaining cases. This is since the middle layer of ResNet is a one-side sparse network, so the computation reduces the toggle rate of data due to partial operands being set to zero, which brings less internal dynamic power consumption. On the other hand, the improvement ratio of ResNet is generally lower than that of LeNet for the same reason: the energy improvement of lower precision is obtained by setting part of mantissa to zeros, so the reduced data toggle rate at the input side already eliminates this part of the data reversal and the improvement is not as obvious as that of LeNet.

Finally, the performance of VARFMA is compared with the baseline. As mentioned above, VARFMA trades an 8.04% area increase for an energy efficiency improvement. Fig. 7 compares the energy efficiency ratio of VARFMA to the baseline running LeNet and ResNet at different formats. It can be seen that VARFMA outperforms the baseline in terms of energy efficiency on both NNs as a whole. The energy efficiency improvement ranges from 5.3% to 10.47% in TF32, FP16 and BF16 formats, and exceeds 20% in both PULP8 and 4M3E8 formats, reaching a maximum of 28.93% running LeNet. VARFMA shows higher energy efficiency when running NN with the same unilateral bit-sparsity weight data. Due to the additional multi-precision design, it fails to show the designed advantage at the maximum format FP32, and the energy efficiency is 2.44% lower for LeNet and 9.13% lower for ResNet.

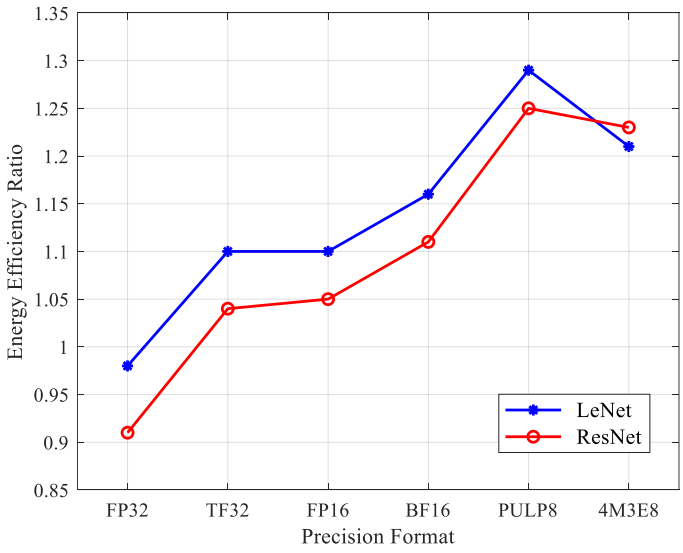

Figure. 7 Tendency of the energy efficiency ratio of VARFMA to the baseline

In addition, VARFMA generally has a higher improvement ratio on both NNs compared to the baseline. It reaches a maximum ratio of 1.81×, while the baseline only has a maximum of 1.37×, which is even less than the average of the VARFMA improvement. This is due to that NN is a method with dense data reuse and the internal data transfer rate is much higher than SAXPY and MATRIX MULTIPLICATION, so the processing of mantissa makes VARFMA has greater energy efficiency improvement.

5. CONCLUSIONS AND FUTURE WORK

We present VARFMA, an FMA design capable of inputting exponent width and mantissa width by variables, which can support FP in any format within maximum precision. It shows a generally higher energy efficiency compared to the baseline for both LeNet and ResNet, reaching up to 28.93% improvement at PULP8. In addition, VARFMA shows an up to 1.81× energy efficiency improvement ratio of lower-precision FP to FP32 than baseline, with only an 8.04% increase in area, which makes it more advantageous for edge-end distributed NNs. The design can be used in distributed machine learning frameworks such as Federated Learning to train NN models locally, ensuring data security and privacy.

The LBR design makes FMA more flexible but is not conducive to handling data in single instruction multiple data (SIMD) format. We would like to experiment more with how to use VARFMA to support SIMD formats in the future. We are working on a configurable FP format architecture design based on FP data rearrangement with SIMD support.

ACKNOWLEDGEMENT

This work was supported by Shenzhen Science and Technology Program (JCYJ20180508152046428) in China.

REFERENCES

[1] Shih, J. L., Lee, C. H., & Yang, C. S. (2007). "An adult image identification system employing image retrieval technique." Pattern recognition letters, vol. 28, no. 16, pp. 2367-2374.

[2] E. Cambria and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]," in IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 48-57.

[3] Grulich, P. M., & Nawab, F. (2018). "Collaborative edge and cloud neural networks for real-time video processing." Proceedings of the VLDB Endowment, vol. 11, no. 12, pp. 2046-2049.

[4] H. Kaul et al., "A 1.45GHz 52-to-162GFLOPS/W variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm CMOS," 2012 IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, 2012, pp. 182-184.

[5] Köster, U., Webb, T., Wang, X., Nassar, M., Bansal, A. K., Constable, W., ... & Rao, N. (2017). "Flexpoint: An adaptive numerical format for efficient training of deep neural networks." Advances in neural information processing systems, pp. 30.

[6] Kim, K. H., Hong, S., Roh, B., Cheon, Y., & Park, M. (2016). Pvanet: Deep but lightweight neural networks for real-time object detection. arXiv preprint arXiv:1608.08021.

[7] T. Tan and G. Cao, "FastVA: Deep Learning Video Analytics Through Edge Processing and NPU in Mobile," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Toronto, ON, Canada, 2020, pp. 1947-1956.

[8] J. Reinders. (Jun. 2017). Intel AVX-512 Instructions, Intel Software Developer Zone. [Online]. Available: https://software.intel.com/en-us/blogs/2013/avx-512-instructions.

[9] N. Kurd et al., "Haswell: A Family of IA 22 nm Processors," in IEEE Journal of Solid-State Circuits, vol. 50, no. 1, pp. 49-58.

[10] N. Brunie, F. de Dinechin and B. de Dinechin, "A mixed-precision fused multiply and add," 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Pacific Grove, CA, USA, 2011, pp. 165-169.
[11] K. Ueyoshi et al., "QUEST: A 7.49TOPS multi-purpose log-quantized DNN inference engine stacked on 96MB 3D SRAM using inductive-coupling technology in 40nm CMOS," 2018 IEEE International Solid - State Circuits Conference - (ISSCC), San Francisco, CA, USA, 2018, pp. 216-218.
[12] E. Chung et al., "Serving DNNs in Real Time at Datacenter Scale with Project Brainwave," in IEEE Micro, vol. 38, no. 2, pp. 8-20.
[13] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & Yoon, D. H. (2017) "In-datacenter performance analysis of a tensor processing unit." In Proceedings of the 44th annual international symposium on computer architecture, pp. 1-12.
[14] Xie, S., Davidson, S., Magaki, I., Khazraee, M., Vega, L., Zhang, L., & Taylor, M. B. (2018). "Extreme datacenter specialization for planet-scale computing: Asic clouds." ACM SIGOPS Operating Systems Review, vol. 52, no.1, pp. 96-108.
[15] S. Mach, F. Schuiki, F. Zaruba and L. Benini, "FPnew: An Open-Source Multiformat Floating-Point Unit Architecture for Energy-Proportional Transprecision Computing," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 4, pp. 774-787.
[16] M. J. Schulte and E. E. Swartzlander, "A family of variable-precision interval arithmetic processors," in IEEE Transactions on Computers, vol. 49, no. 5, pp. 387-397.
[17] A. Agrawal et al., "9.1 A 7nm 4-Core AI Chip with 25.6TFLOPS Hybrid FP8 Training, 102.4TOPS INT4 Inference and Workload-Aware Throttling," 2021 IEEE International Solid- State Circuits Conference (ISSCC), San Francisco, CA, USA, 2021, pp. 144-146.
[18] A. Nannarelli, "Fused Multiply-Add for Variable Precision Floating-Point," 2019 32nd IEEE International System-on-Chip Conference (SOCC), Singapore, 2019, pp. 342-347.
[19] H. Zhang, D. Chen and S. -B. Ko, "Efficient Multiple-Precision Floating-Point Fused Multiply-Add with Mixed-Precision Support," in IEEE Transactions on Computers, vol. 68, no. 7, pp. 1035-1048.
[20] Muller, J. M., Brisebarre, N., De Dinechin, F., Jeannerod, C. P., Lefevre, V., Melquiond, G., ... & Torres, S. (2018). Handbook of floating-point arithmetic.

AUTHORS

Xiyang Sun received the B.E. degree in Integrated Circuit Design and Integrated System from Tianjin University in 2020. She is currently pursuing the M.E. degree with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China.

Yue Zhao received the B.E. degree in Telecommunications Engineering from Tianjin University, Tianjin, China, in 2020. He is currently pursuing the M.E. degree with the Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China. His research interests include High Performance Computer Architecture and Integrated Circuit Systems Design.

Sheng Zhang is with the Tsinghua Shenzhen International Graduate School at Shenzhen, Tsinghua University, Key Laboratory of Advanced Sensor and Integrated System, Shenzhen