Addition is Most You Need: Efficient Floating-Point SRAM Compute-in-Memory by Harnessing Mantissa Addition

Weidong Cao[1,*], Jian Gao[2,*], Xin Xin[3], Xuan Zhang[2]
[1]The George Washington University; [2]Northeastern University; [3]University of Central Florida

ABSTRACT

The compute-in-memory (CIM) paradigm holds great promise to efficiently accelerate machine learning workloads. Among memory devices, static random-access memory (SRAM) stands out as a practical choice for its exceptional reliability in the digital domain and excellent scalability. Recently, there has been a growing interest in accelerating floating-point (FP) deep neural networks (DNNs) with SRAM CIM due to their critical importance in DNN training and high-accurate inference. This paper proposes an energy-efficient SRAM CIM macro for FP DNNs. To achieve the design, we identify a lightweight approach that decomposes conventional FP mantissa multiplication into two parts: mantissa sub-addition (sub-ADD) and mantissa sub-multiplication (sub-MUL). Our study shows that while mantissa sub-MUL is compute-intensive, it only contributes to the minority of FP products, whereas mantissa sub-ADD, although compute-light, accounts for the majority of FP products. Recognizing "Addition is Most You Need", we develop a novel hybrid-domain SRAM CIM macro to accurately handle mantissa sub-ADD in the digital domain while improving the energy efficiency of mantissa sub-MUL using analog computing. Experiments with the MLPerf benchmark show its remarkable improvement in energy efficiency on average by 3×~ 3.6× (2.5×~3.1×) in inference (training) compared to a fully digital baseline without any accuracy loss, showcasing its great potential for FP DNN acceleration.

1 INTRODUCTION

The compute-in-memory (CIM) has emerged as a highly promising computing paradigm by colocating computation and storage in close proximity [1, 2, 4, 10, 16, 21]. In particular, it has shown remarkable energy efficiency in accelerating a variety of machine learning tasks, ranging from image classification and object recognition to natural language processing [14, 16, 19, 21]. Among the various memory devices explored [6, 8, 14, 16, 17, 19, 21, 22], static random-access memory (SRAM) stands out in building up CIM systems [14–16, 19, 21] for practical applications due to its exceptional reliability in the digital domain while maintaining superior performance, power, and area. Although both academia [14, 16] and industry [3, 5] have been advancing the performance of SRAM CIM, most endeavors [3, 5, 14] have been limited to accelerate quantized deep neural network (DNN) models for inference [20] due to their relatively simple arithmetic operations, lightweight computing unit, and, accordingly, high energy efficiency compared to their floating-point (FP) counterparts [12].

Recently, there has been a growing interest in studying FP SRAM CIM macros [16, 19, 21]. This is because DNN models have been predominantly based on FP arithmetic operations to achieve the highest training quality [11, 12]. In addition, for mission-critical applications such as autonomous driving, security robots, and defense drones, quantized DNN models do not always ensure the stringent accuracy requirement [6, 12, 13]. As a result, efficient CIM systems to accelerate FP DNN models are highly coveted and crucial for unlocking the full power of machine learning (ML). Unfortunately, only a few SRAM CIM macros have been proposed to speed up FP DNN models for edge or cloud applications [16, 19, 21]. Despite the notable performance improvements compared to GPUs and systolic neural processing units, there is still substantial untapped potential to further optimize the energy efficiency of FP CIM macros. In particular, current SRAM CIM macros [16, 19, 21] that focus on boosting vector-matrix multiplication (VMM) often treat FP multiplication and FP addition as the basic units of computation in the digital domain. They implement these FP operations by overminimizing computation error on every single bit operation. Our studies show that fully accurate computations incur energy inefficiency and are not necessary for FP DNNs due to their inherent resilience to small perturbations. Such insights provide us with promising opportunities to enhance the performance of FP SRAM CIM macros.

This paper proposes an efficient SRAM CIM macro to accelerate FP DNNs by harnessing the inherent advantage of FP arithmetic. By dividing the mantissa multiplication into two parts, i.e., mantissa sub-addition (sub-ADD) and sub-multiplication (sub-MUL), we find that sub-ADD contributes to ~75% of FP products while consuming <10% of the total energy (Section. 3.1). This insight of "Addition is Most You Need" motivates our design approach for the proposed FP SRAM CIM macro: dedicating digital resources to guarantee the accuracy of lightweight mantissa sub-ADD, while exploiting energy-efficient analog computing to reduce the complexity of mantissa sub-MUL. This hybrid-domain acceleration achieves state-of-the-art energy efficiency without compromising the accuracy of DNN models, outperforming the traditional fully-accurate digital-only baseline [21]. Key contributions of this work are listed below.

- We identify a lightweight mechanism to accelerate FP DNNs by decomposing FP mantissa multiplication into two parts: accuracy-oriented mantissa sub-ADD and efficiency-oriented mantissa sub-MUL.
- We tailor a hybrid-domain SRAM CIM macro to implement the proposed mechanism by placing mantissa sub-ADD in the digital domain and performing mantissa sub-MUL in the analog domain.
- Detailed circuit- and microarchitecture-level features of the proposed FP SRAM CIM macro, such as local computing cells and computation flow, are elaborated.
- Experimental evaluations reveal that the proposed hybrid-domain SRAM CIM macro can improve energy efficiency by 3.0×~ 3.6× (2.5×~3.1×) in inference (training) without degrading accuracy compared to the FP digital baseline.

[*]Equal contribution.
Correspondence: weidong.cao@gwu.edu & xuan.zhang@northeastern.edu.
This work is partially supported by NSF CCF #1942900.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655930

2 BACKGROUND

2.1 Floating-Point Arithmetic Operations

We first briefly introduce the essential FP arithmetic operations underpinning FP DNN models.

FP format: A general FP number in scientific notation is expressed as $f = (-1)^S \cdot 2^E \cdot 1.M$. Here, S ($S = 0$ or $S = 1$), E, and M ($M \in (0, 1)$) represent the sign, exponent, and mantissa (fraction) of the number, respectively. Note that '1' before 'M' is a hidden bit, which is not explicitly shown in the binary format of an FP number and E is the actual exponent with an offset to the exponent encoded in the standard format. Taking the IEEE half-precision FP format (FP16) as an example, S is 1-bit, E is 5-bit, M is 10-bit (refer to Figure 2(d)).

FP multiplication: The multiplication of FP numbers is straightforward, i.e., exponent summation (①) and mantissa multiplication (②), as shown below using two positive numbers for simplicity.

$$(2^{E_0} \cdot 1.M_0) \cdot (2^{E_1} \cdot 1.M_1) = \underbrace{(2^{E_0+E_1})}_{①} \cdot \underbrace{(1.M_0 \cdot 1.M_1)}_{②}. \quad (1)$$

FP addition: However, the addition of FP numbers is non-trivial, which is expressed as

$$2^{E_0} \cdot 1.M_0 + 2^{E_1} \cdot 1.M_1 = \underbrace{2^{E_{\max}}}_{①} \cdot \underbrace{\left\{ \begin{array}{l} \underbrace{2^{E_0-E_{\max}}}_{②} \cdot 1.M_0 \\ + \\ \underbrace{2^{E_1-E_{\max}}}_{②} \cdot 1.M_1 \end{array} \right.}_{③}. \quad (2)$$

It involves alignment first, i.e., finding the maximum exponent, $E_{\max} = \max\{E_0, E_1\}$, among all exponents (①) and obtaining the exponent difference, $E_{0(1)} - E_{\max}$, between each exponent and the maximum one (②). Subsequently, the mantissa part would be shifted towards the right according to the exponent difference and then summed together (③). The sum will then undergo extra processes like truncation to comply with the standard FP format.

FP vector-matrix multiplication: By generalizing Eq. (2) to the accumulation of n FP numbers, where each number, $2^{E_i} \cdot 1.M_i$ is assumed to be a product of a weight-activation pair, i.e., $(-1)^{W_{S,i}} \cdot 2^{W_{E,i}} \cdot 1.W_{M,i}$ and $(-1)^{X_{S,i}} \cdot 2^{X_{E,i}} \cdot 1.X_{M,i}$ based on Eq. (1) (i.e., $E_i = W_{E,i} + X_{E,i}$ and $1.M_i = 1.W_{M,i} \cdot 1.X_{M,i}$), the vector-matrix multiplication (VMM) of FP numbers is achieved, which is the backbone operation of FP DNN models. FP VMM can support the computation of DNN models with the highest accuracy and the best training quality [12]. Therefore, efficient FP VMM acceleration on hardware is highly desirable.

2.2 Compute-in-Memory for FP DNNs

Compute-in-Memory (CIM) to accelerate FP VMM is conceptually illustrated in Figure 1. First, the alignment is carried out by summing the exponent parts of weight-activation pairs ($W_{E,i}$ and $X_{E,i}$, Step ①) and obtaining the exponent difference between each exponent sum, E_i, and the maximum one, $E_{\max}$ (Step ②). Subsequently, exponent differences ($E_i - E_{\max}$) are used to shift the mantissa parts of activations, which are then used for mantissa multiplication and accumulation (Step ③). Note that these steps in Figure 1 exactly match those shown in Eq. (2).

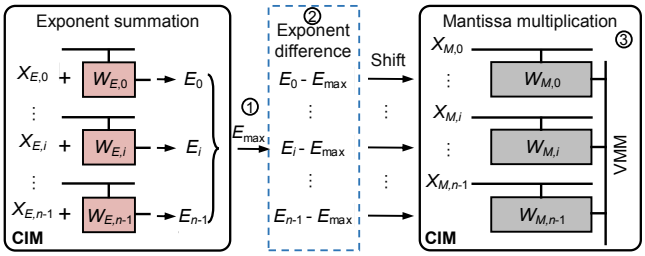


Figure 1: Illustration of CIM for FP VMM acceleration. $X_{E,i}$ and $W_{E,i}$ are the exponent parts of activation and weight. $X_{M,i}$ and $W_{M,i}$ are the mantissa parts of activation and weight. ①, ②, and ③ are circuit-level representations of the steps in Eq. (2).

Previous work has explored CIM acceleration for FP DNN models with emerging non-volatile memory (NVM) devices [6, 9]. Despite their great promise compared to GPUs and systolic neural processing units, such NVM devices are still in the early stages of development and are not reliable enough for practical applications. Recently, static random-access memory (SRAM) has emerged as the cornerstone in the design of CIM systems for practical applications because of its exceptional accuracy in digital computing, coupled with superior performance, power efficiency, and area [14, 16, 19, 21]. However, exploration of FP SRAM CIM macros has been limited to a few edge/cloud applications [16, 19, 21]. As an example, a prior work [16] proposes a reconfigurable FP/INT CIM processor to enable flexible support of BFloat16 (BF16)/FP32 and INT8/16 in the same digital CIM macro. Another work [21] divides FP operations into high-efficiency intensive-CIM and flexible sparse-digital parts, as it observes that most exponents of FP data are clustered in a small range. Circuit-level techniques, such as the time-domain exponent summation mechanism, are also used to improve the energy efficiency of CIM macros [19]. Despite these different implementations, they primarily focus on accurately accelerating almost every exponent summation and mantissa multiplication in the digital domain, resulting in remarkable, yet unnecessary energy waste.

Our analysis reveals that by leveraging the inherent advantage of FP arithmetic operations and exploiting the inherent resilience of DNN models to small computation errors, there is significant untapped potential to leverage approximating computing (e.g., analog computing) to accelerate FP DNNs without degrading their accuracy (Section 3.1). This revelation opens valuable opportunities to further enhance the performance of FP SRAM CIM macros.

3 APPROACH

3.1 Insight: Addition is Most You Need

By carefully examining FP arithmetics, we find a lightweight manner to allow hardware-friendly implementation of SRAM CIM macros with enhanced energy efficiency while maintaining the accuracy of FP DNNs. To show the key idea, we decomposed the conventional FP mantissa multiplication in Eq. (1) into two parts as

$$(1.M_0 \cdot 1.M_1) = (1 + M_0) \cdot (1 + M_1) = (\underbrace{1 + M_0 + M_1}_{\text{sub-ADD}} + \underbrace{M_0 \cdot M_1}_{\text{sub-MUL}}), \quad (3)$$

where $(1 + M_0 + M_1)$ and $(M_0 \cdot M_1)$ are defined as mantissa sub-addition (sub-ADD) and mantissa sub-multiplication (sub-MUL). Following the decomposition and given $M_{0(1)} \in (0, 1)$, we study

the significance of sub-MUL in mantissa multiplication, i.e.,

$$\frac{M_0 \cdot M_1}{(1+M_0)\cdot(1+M_1)} = \frac{1}{(1+1/M_0)\cdot(1+1/M_1)} \le \frac{1}{4}. \quad (4)$$

Here, the '=' holds true if and only if $M_0 = M_1 \to 1$. Eq. (4) shows that for a single weight-activation pair, if the computation accuracy of sub-ADD can be ensured, the *total computation error of mantissa multiplication (also for the FP product) does not exceed* 1/4 *compared to its ground truth even by aggressively removing the sub-MUL*. Figure 5 shows our experimental results (the red dots) by throwing away sub-MUL operations in FP multiplication of a single weight-activation pair with exemplary FP DNN models from our benchmark. The experiment verifies that the total computation error resulting from removing all sub-MUL operations is bounded by 1/4, and *the error is smaller than 1/4 in most cases.*

The analysis here shows that although mantissa sub-MUL operations are computationally intensive, they often impact a minority of FP products. In contrast, mantissa sub-ADD, although computationally lighter, often constitutes the majority of FP products. And addition is much more energy efficient compared to multiplication. For example, INT8 addition consumes about 10% energy of INT8 multiplication according to previous work [7]. This pivotal insight of "Addition is Most You Need" inspires our FP SRAM CIM macro-design strategy: **allocating digital resources to ensure the precision of compute-light mantissa sub-ADD and employing energy-efficient analog computing for compute-intensive mantissa sub-MUL**.

3.2 Hybrid-Domain FP SRAM CIM Macro

Microarchitecture overview: Figure 2(a) illustrates the microarchitecture of the proposed SRAM CIM macro (targeting FP16 DNN models as an example). It consists of a time-domain CIM exponent summation array (ESA, to add the exponent parts of weight-activation pairs), time-domain MAX identifier (to find the maximum exponent sum), exponent difference extractor (EDE, to extract the exponent difference between each exponent sum and the maximum one), time-to-digital processing unit (to convert the exponent difference in the time domain into digits), EDE-based input alignment unit (to shift mantissa parts of activations), hybrid-domain CIM mantissa VMM array (HD-MVA, for FP VMM), analog-to-digital converters (ADCs), shift-and-add (S&A) module, local digital adder tree, partial-product management (PM) unit, and other auxiliary modules such as the IO module and pipeline and timing control unit. For exponent summation and alignment, we implement them in the time domain, since previous work [18, 19] has shown that the time-domain implementation is more energy efficient as compared to other digital implementation manners. For mantissa multiplication, we develop a hybrid-domain SRAM CIM macro to precisely manage mantissa sub-ADD in the digital realm while enhancing the energy efficiency of mantissa sub-MUL through analog computing.
Computation flow: The computation process of the proposed SRAM CIM Macro is shown in Figure 2(b), which contains seven key steps. In Step ①, ESA computes the sum of the exponent parts of each weight-activation pair in the time domain, i.e., $E_i = W_{E,i} + X_{E,i}$. In Step ②, the maximum exponent sum, $E_{\max}$, among all sums, is found by the time-domain MAX identifier. In Step ③, all exponent sums (E_i, $i = 0, ..., n-1$) and $E_{\max}$ are sent to the EDE unit, where

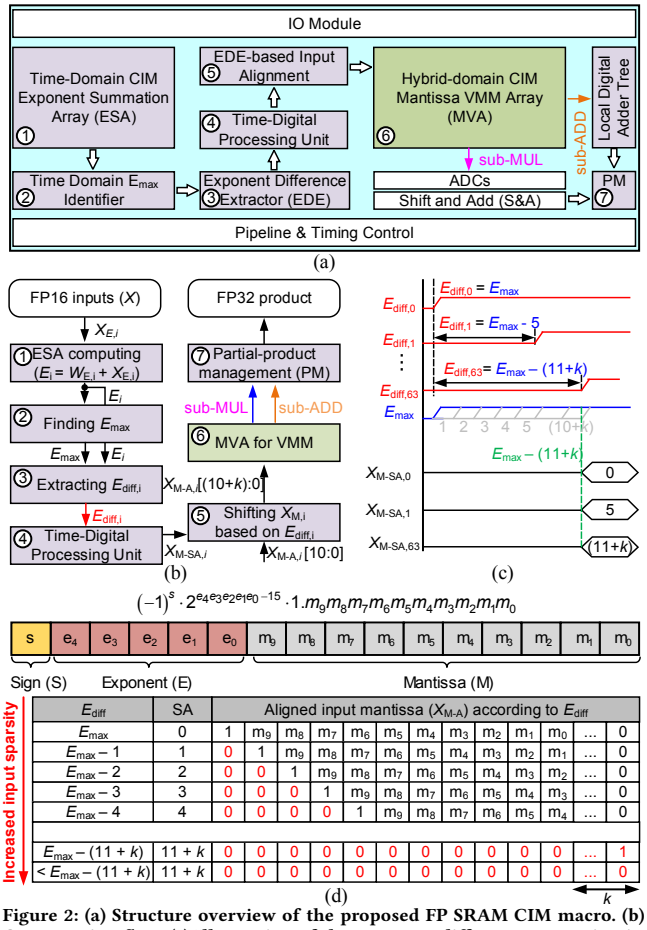


Figure 2: (a) Structure overview of the proposed FP SRAM CIM macro. (b) Computation flow. (c) Illustration of the exponent difference extraction in the time domain. (d) Illustration of mantissa shift based on the exponent difference (using FP16 as an example).

the exponent difference $E_{\text{diff},i}$ between E_i and $E_{\max}$ is obtained, i.e., $E_{\text{diff},i} = E_{\max} - E_i$. In particular, $E_{\text{diff},i}$ is calculated by measuring the time interval between the rising edges of E_i and $E_{\max}$ as shown in Figure 2(c). In Step ④, the time-to-digital processing unit converts $E_{\text{diff},i}$ from the time domain into a digital shift-amount (SA) value, $X_{\text{M-SA},i}$, for each mantissa, $X_{M,i}$. The conversion relation between $E_{\text{diff},i}$ and $X_{\text{M-SA},i}$ is straightforward as illustrated in Figure 2(c). With this $X_{\text{M-SA},i}$, in step ⑤, the original mantissa $X_{M,i}$ is shifted to generate the aligned mantissa, $X_{\text{M-A},i}$ with the same bit-width, i.e., $11 + k$. If $E_{\text{diff},i} > (11 + k)$, $X_{\text{M-A},i}$ would become 0, allowing computation to be skipped. "11" is the bit width of the mantissa part in FP16 format (1 hidden bit plus 10 explicit mantissa bits). Note that k is a tuning value, allowing for a further trade-off between accuracy and energy efficiency [19], which is discussed in the next paragraph. Otherwise, if $E_{\text{diff},i} \le (11 + k)$, $X_{\text{M-A},i}$, as shown in Figure 2(d), the insert-0 behavior coupled with the growth of the exponent difference increases the bit-level sparsity, which reduces the energy consumption of the proposed HD-MVA with the input-sparsity-aware circuit. In step ⑥, MVA uses the aligned mantissa parts to achieve VMM with analog computing, sub-MUL, and digital computing, sub-ADD. Finally (Step ⑦), $E_{\max}$, the partial products from sub-MUL and sub-ADD are post-processed by the PM unit to obtain the final product in the standard FP32 format.

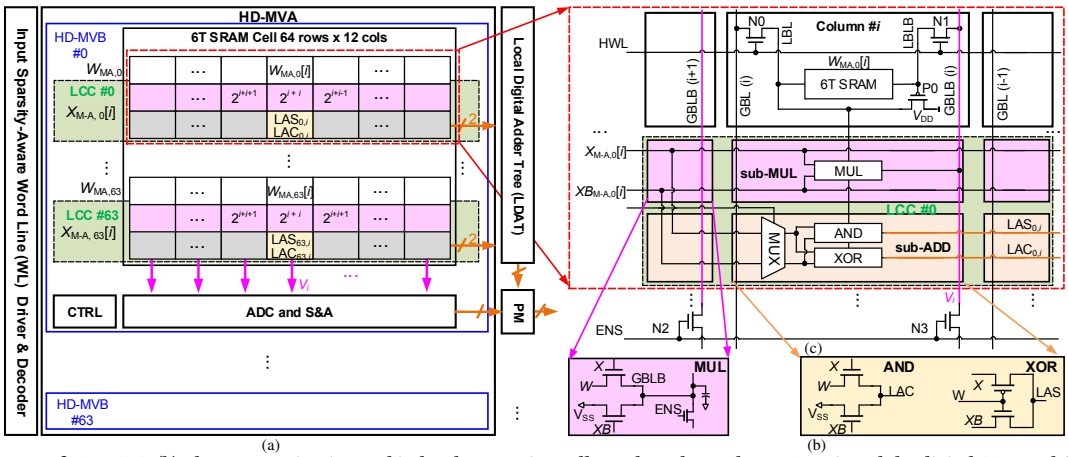


Figure 3: (a) Structure of HD-MVA. (b) Elementary circuits used in local computing cells, such as the analog MUL unit and the digital AND and OR gate. (c) Local computing cell to implement the proposed hybrid computing mechanism.

Ideally, the bit-width of an aligned activation mantissa is $(11 + E_{\max} - E_{\min})$, where $E_{\min}$ is the minimum exponent sum among all E_i. However, if $(E_{\max} - E_{\min})$ is too large, the product of a weigh-activation pair whose exponent sum is in the neighbors of $E_{\min}$ could have a negligible contribution to the final product of the VMM. Therefore, we use the tunable k to find a proper shift-amount value for alignment, which can ensure the accuracy of the computation while further improving the energy efficiency. Additionally, we use a coarse pipeline to increase the throughput of the proposed CIM macro, where Steps ① to ④ are classified into pipeline stage one and the remaining steps are in pipeline stage two.

Sub-circuit design: The key circuit blocks to implement steps ① to ⑤ are similar to the previous work [18, 19], whose details are thus omitted here. Instead, we focus on introducing the proposed HD-MVA as shown in Figure 3(a). The HD-MVA contains 64 hybrid-domain CIM mantissa VMM banks (HD-MVB). Each HD-MVB is used for computing the product of aligned activation mantissa $X_{\text{M-A}}$ and weight mantissa W_{MA}, i.e., $\sum(X_{\text{M-A},i} \cdot W_{\text{MA},i})$. Here, $X_{\text{M-A},i}$ is $(11 + k)$-b wide; $W_{\text{MA},i}$ is 12-b wide and represented in 2's complement. To show the working mechanism of HD-MVB, we assume $k = 0$ for simplicity and take the circuit in Figure 3(b) as an example.

Each HD-MVB comprises a 6T-SRAM cell array (64 rows and 12 columns) and a local computing cell (LCC) associated with each row. In particular, each bit of $W_{\text{MA},i}$ is stored in the same row but across different columns of the 6T-SRAM cell array in the order from the most significant bit (MSB) to the least significant bit (LSB). Each LCC includes a one-bit multiplication unit (MUL) for sub-MUL in the analog domain and a half-adder for sub-ADD in the digital domain. In each input cycle j, the row i sends a 2-b sum of $(X_{\text{M-A},i}[j] + W_{\text{MA},i}[j])$, i.e., $\text{LAS}_{i,j}$ and $\text{LAC}_{i,j}$, to the local digital adder tree (LDAT). Here, $\text{LAS}_{i,j}$ is the local-add sum bit, and $\text{LAC}_{i,j}$ is the local-add carry bit. In the same cycle, each column (global bitline bar, GBLB) also generates a partial product of $X_{\text{M-A}}[j]$ and W_{MA}. Across cycles, the LDAT accumulates partial sums of mantissa summation and the S&A logic accumulates the partial products of mantissa multiplication. Finally, PM combines these results to generate the standard FP32 product for subsequent processing.

Similarly to previous LCC structures [18, 19], the two pass-transistors ($N0/N1$) connect GLB/GBLB to the local bitline (LBL/LBLB) for read and write operations in SRAM mode. In standby mode, the horizontal wordline (HWL) is activated, i.e., HWL = 1, with GBLs pre-charging LBLs. In CIM mode, $N0/N1$ are switched off to decoupled GBLs from LBLs. The selected WL is activated to read 1b of weight mantissa (e.g., $W_{\text{MA}}[j]$), which is stored in the SRAM cell accessed with a large voltage swing. The multiplexer (MUX) is used to select a specific bit of $X_{\text{M-A}}$ to be added to the corresponding bit of the weight mantissa in a cycle, whose control signal is from the decoder in the input module of the HD-MVA. The schematics of the MUL unit for sub-MUL operations and the AND/XOR gate for sub-ADD operations are illustrated in Figure 3(c). All of these circuits use a two-transistor structure to minimize area overhead.

Table 1: Benchmark from the MLPerf Inference: Edge Benchmark Suite v3.1.

Task	Model	Dataset	Quality (99% FP32)
Image classification	ResNet50-v1.5	ImageNet	76.014% (top1 Acc)
Object detection	Retinanet	OpenImages	0.3755 mAP
Language processing	BERT-large	SQuAD v1.1	90.874% (f1_score)
Speech-to-text	RNNT	Librispeech	92.548% (1 - WER)

4 EXPERIMENTS

4.1 Experimental Methodology

Benchmarks: We evaluate both the inference and training performance of the proposed FP SRAM CIM macro. Current FP SRAM CIM macros mainly target edge applications with FP16 DNN models [16, 19, 21]. We follow the convention and use the MLPerf Inference: Edge benchmark suite v3.1 [13] and MLPerf Training Benchmark Suite v3.1 [11] as our benchmarks. The inference benchmark includes image classification, object detection/segmentation, speech-to-text, language processing, and recommendation tasks, while the training benchmark extends them further to large language models and image generation. We select four inference tasks and one training task to showcase our proposed CIM macro's performance. Their associated datasets, models, and quality metrics for inference are listed in Table 1. The metrics for these models are top-1 accuracy for the image classification (ResNet50-v1.5), mean average precision (mAP) for the objection detection (Retinanet), f1_score which represents the harmonic mean of the precision and recall for the language processing (BERT-large), and 1-word error rate (WER) for the speech-to-text (RNNT).

Baseline: The fully digital architecture of the state-of-the-art FP SRAM CIM macro [21] is used as our baseline. This work implements FP operations in two parts: high-efficiency intensive-CIM and flexible sparse-digital parts, as it is observed that most of the exponents of FP data are clustered in a small range. This macro accelerates FP VMM in the digital domain without any approximate computation. For a fair comparison, we build the two architectures based on the same pipeline with 28nm CMOS technology

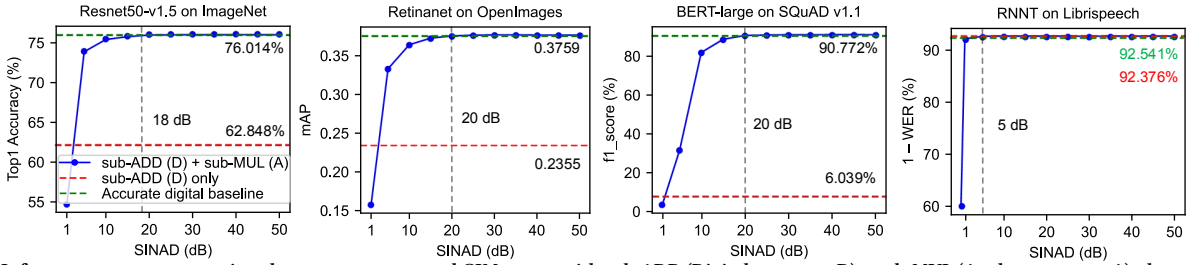


Figure 4: Inference accuracy comparison between our proposed CIM macro with sub-ADD (Digital compute, D) + sub-MUL (Analog compute, A), the macro with sub-ADD (Digital compute, D) only, and the accurate digital baseline with MLPerf Inference: Edge Benchmark Suite v3.1 Tasks.

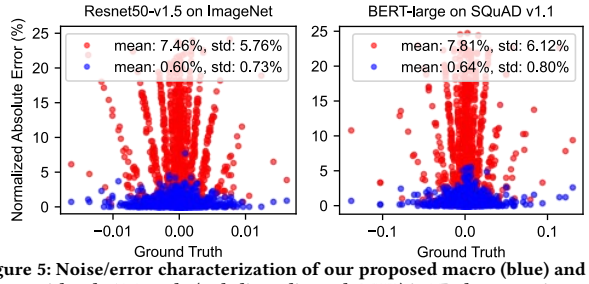


Figure 5: Noise/error characterization of our proposed macro (blue) and the macro with sub-ADD only (red, discarding sub-MUL) in FP element-wise multiplication with exemplary DNN layers from MLPerf. The ground truth is from the fully accurate digital baseline. Our macro achieves SINAD_{hw} = 30 dB with a much smaller computation error.

and use the same capacity of SRAM cells, i.e., 832kb. We use the same power supply $V_{\text{DD}} = 0.8V$ and the clock frequency of 200 MHz for them. We do not consider the tunable parameter k mentioned in Section 3.2 when comparing with two architectures, i.e., $k = (E_{\max} - E_{\min})$. We first evaluate the computation accuracy of the analog flow (for the sub-MUL operations) used in our proposed CIM macro. We then evaluate the accuracy/energy efficiency for the two designs. Finally, we present a summary.

4.2 Macro Accuracy Evaluation

Noise characterization of the analog dataflow: We first characterize the computation accuracy of the analog flow used for the sub-MUL operations. The proposed SRAM CIM macro employs an analog dataflow for sub-MUL operations and hence is subject to various hardware non-idealities, such as process, voltage, and temperature (PVT) variations, thermal noise, and quantization noise of ADCs. Together, these non-idealities act to influence the computation accuracy of the macro. We adopt the signal-to-noise and distortion ratio (SINAD) to represent the computation accuracy of the analog dataflow for sub-MUL operations [2]. To characterize the SINAD of our proposed CIM macro, we perform hundreds of such Monte Carlo (MC) simulations on hardware by using random weight-activation pairs, the errors statistics between the hardware computation results, D_{hw}, and the software ground truth, D_{sw}, can be obtained. The variation ϵ of the hardware noise is then expressed as $(\epsilon)^2 = P_{\text{noise}} = \text{mean}(D_{\text{hw}} - D_{\text{sw}})^2$. With the variations, the signal-to-noise and distortion ratio (SINAD) of the analog dataflow is characterized as $\text{SINAD}_{\text{hw}} = 10\log_{10}((P_{\text{sig}}(D_{\text{sw}}) + P_{\text{noise}})/P_{\text{noise}})$. Intuitively, the smaller variation ϵ is, the higher the SINAD is, and the more accurate the computation is. The evaluations in Figure 5 are from thousands of simulations with exemplary DNN layers from MLPerf image classification and language processing tasks. The ground truth is the element-wise product from the accurate digital baseline. The statistics of the normalized absolute error between D_{hw} and D_{sw} of our proposed macro (blue) shows an equivalent SINAD_{hw} of 30 dB with analog computing. The results indicate that our proposed macro yields a mean normalized absolute error that is around 12× smaller compared to the one (red) of the sub-ADD-only computation (i.e., discarding sub-MUL).

Inference accuracy: To examine whether the SINAD_{hw} derived from our analog dataflow can support adequate system-level inference accuracy, we analyze the inference accuracy of an FP DNN model on the proposed CIM macro by sweeping the level of SINAD. In this way, we can obtain $\text{SINAD}_{\text{min}}$, the minimum SINAD required to achieve the software-equivalent inference accuracy. In particular, we apply a reverse method to inject equivalent software-level noise into each layer's input activations of an FP DNN model, i.e., adding noise to D_{sw} with the corresponding SINAD.

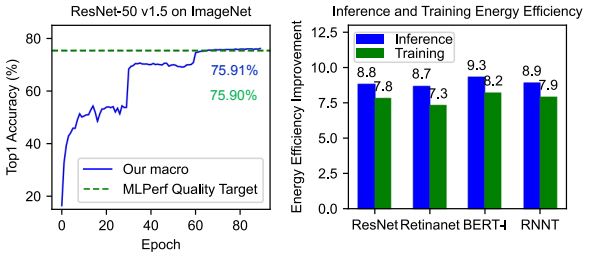
Figure 4 demonstrates the effects of varying the SINAD on the inference accuracy of different DNNs with software-level sweeping, where the blue line represents the sweeping curve of our proposed CIM macro (sub-ADD (Digital compute, D) + sub-MUL (Analog compute, A)). Typically, a 20 dB SINAD is sufficient for all DNN models to reach the ideal inference accuracy. Especially, RNNT on Librispeech only needs 5 dB to achieve the accuracy of the digital baseline. We believe that the robustness of RNNT against the noise from the sub-MUL analog computing is attributed to its network architecture by using Long short-term memory (LSTM) cells. LSTMs have a unique architecture with input, output, and forget gates. These gates effectively regulate the flow of information, allowing the network to retain important information over long sequences and forget irrelevant data. This gating mechanism allows the errors to propagate through the network in a more controlled manner. Since our macro provides a SINAD as high as 30 dB, all inference accuracy can be guaranteed without degeneration. However, if the sub-MUL operations are excluded from the FP VMM, the accuracy of models significantly drops. As an example, the Top-1 accuracy of ResNet50-v1.5 on ImageNet decreases to 62.848% without the sub-MUL operations. The BERT-large on SQuAD v1.1 can even fail to work by throwing away the sub-MUL operations.

Training accuracy: We adopt the same manner as above to evaluate the training accuracy of an FP DNN model on the proposed CIM macro. We inject the noise based on the obtained SINAD_{hw} into the FP DNN models during the training to emulate the non-ideal hardware. Figure 6(a) shows a training example by using ResNet-50 v1.5 on ImageNet. The top-1 training accuracy with the proposed CIM macro is 75.91% with no accuracy loss to the digital baseline.

4.3 Macro System-Level Evaluation

Energy efficiency: We further evaluate the energy efficiency of our proposed CIM macro on FP DNN acceleration by comparing it to the fully digital baseline [21]. On average, our proposed CIM macro can improve energy efficiency by 3×~3.6× (2.5×~3.1×) for inference

If $\text{SINAD}_{\text{hw}} \geq \text{SINAD}_{\text{min}}$, we consider that the analog dataflow has sufficient computation accuracy to guarantee the system-level inference accuracy.



(a) (b)

Figure 6: (a) The training trajectory on MLPerf Training Benchmark Suite v3.1 Image classification task with our proposed macro. (b) Normalized inference (training) energy efficiency to the digital baseline using model layers with a more dense distribution of exponents of FP weights/activations.

Table 2: Summary of performance of the proposed CIM macro for layers with dense exponent distribution or on average.

	Inference		Training	
	Accuracy	Energy Efficiency	ACC	EE
Baseline [21]	No loss	1×	No loss	1×
Our work (Dense)	No loss	8.7×~ 9.3×	No loss	7.3×~8.2×
Our work (AVG)	No loss	3×~ 3.6×	No loss	2.5×~3.1×

(training) compared to the baseline across various benchmark models. The improvement is mainly due to the efficient acceleration of sub-MUL operations in the analog domain. In particular, as shown in Figure 6, if the exponent distribution of FP weight/activations in a particular model layer is more dense, our macro is able to improve the energy efficiency by 8.7×~ 9.3× (7.3×~8.2×) during inference (training) compared to the baseline. This further improvement is primarily due to the lower cost of the mantissa alignment.

Performance and area: Since our proposed macro aligns the mantissa in the time domain, it can reduce the time of each pipeline stage, leading to a 1.23× improvement in performance compared to the baseline. The local digital adder tree (LDAT) in our proposed macro is only used for the accumulation of partial sums of sub-ADD, which minimizes the area overhead. In contrast, the baseline requires area-intensive digital adder trees for the accumulation of all partial sums, and RISC-V CPU for the computation of weights and activations that are sparse in their distributions. Compared to this baseline, our proposed CIM macro achieves 1.43× area efficiency.

Summary: Finally, we make a summary of the performance of the proposed CIM macro for FP DNN acceleration as shown in Table 2. It achieves better performance in terms of energy efficiency, latency, and area efficiency compared to baseline [21] with the same amount of computing resources. This comparison is preliminary and based on circuit simulations without detailed optimizations. Future efforts will focus on optimizing the hardware configuration of the proposed macro and evaluating it with silicon prototypes.

5 CONCLUSION

This paper has introduced an efficient SRAM CIM macro for FP DNNs. It splits FP mantissa multiplication into mantissa sub-addition (sub-ADD) and sub-multiplication (sub-MUL). It shows that sub-MUL is compute-intensive but contributes less to FP vector-matrix-multiplication (VMM), sub-ADD contributes more to FP VMM yet less compute-heavy. Leveraging this insight, we develop a hybrid-domain SRAM CIM macro that efficiently processes sub-ADD digitally and enhances sub-MUL's energy efficiency through analog computing. Experiments show significant improvements in energy efficiency over digital baselines without hurting accuracy.

REFERENCES

[1] Weidong Cao and Xuan Zhang. 2023. A/D Alleviator: Reducing Analog-to-Digital Conversions in Compute-In-Memory with Augmented Analog Accumulation. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5.

[2] Weidong Cao, Yilong Zhao, and et al. 2022. Neural-PIM: Efficient Processing-In-Memory With Neural Approximation of Peripherals. *IEEE Trans. Comput.* 71, 9 (2022), 2142–2155.

[3] Yu-Der Chih, Po-Hao Lee, Hidehiro Fujiwara, Yi-Chun Shih, Chia-Fu Lee, Rawan Naous, Yu-Lin Chen, and et al. 2021. 16.4 An 89TOPS/W and 16.3TOPS/mm2 All-Digital SRAM-Based Full-Precision Compute-In Memory Macro in 22nm for Machine-Learning Edge Applications. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 64. 252–254.

[4] William J. Dally, Yatish Turakhia, and Song Han. 2020. Domain-Specific Hardware Accelerators. *Commun. ACM* 63, 7 (jun 2020), 48–57.

[5] Hidehiro Fujiwara, Haruki Mori, Wei-Chang Zhao, Mei-Chen Chuang, Rawan Naous, Chao-Kai Chuang, and et al. 2022. A 5-nm 254-TOPS/W 221-TOPS/mm2 Fully-Digital Computing-in-Memory Macro Supporting Wide-Range Dynamic-Voltage-Frequency Scaling and Simultaneous MAC and Write Operations. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. 1–3.

[6] Mohsen Imani, Saransh Gupta, Yeseong Kim, and Tajana Rosing. 2019. FloatPIM: In-Memory Acceleration of Deep Neural Network Training with High Precision. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. 802–815.

[7] Norman P. Jouppi, Doe Hyun Yoon, Matthew Ashcraft, Mark Gottscho, and et al. 2021. Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 1–14.

[8] Heesu Kim, Hanmin Park, Taehyun Kim, Kwanheum Cho, Eojin Lee, Soojung Ryu, Hyuk-Jae Lee, Kiyoung Choi, and Jinho Lee. 2021. GradPIM: A Practical Processing-in-DRAM Architecture for Gradient Descent. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 249–262.

[9] Yandong Luo and Shimeng Yu. 2020. Accelerating deep neural network in-situ training with non-volatile and volatile memory based hybrid precision synapses. *IEEE Trans. Comput.* 69, 8 (2020), 1113–1127.

[10] Tianrui Ma, Weidong Cao, Fei Qiao, Ayan Chakrabarti, and Xuan Zhang. 2022. HOGEye: Neural Approximation of HOG Feature Extraction in RRAM-Based 3D-Stacked Image Sensors. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*.

[11] Peter Mattson, Christine Cheng, and et al. 2020. MLPerf Training Benchmark. arXiv:cs.LG/1910.01500

[12] Paulius Micikevicius, Dusan Stosic, and et al. 2022. FP8 Formats for Deep Learning. arXiv:cs.LG/2209.05433

[13] Vijay Janapa Reddi, Christine Cheng, and et al. 2020. MLPerf Inference Benchmark. arXiv:cs.LG/1911.02549

[14] Jian-Wei Su, Yen-Chi Chou, and et al. 2021. 16.3 A 28nm 384kb 6T-SRAM Computation-in-Memory Macro with 8b Precision for AI Edge Chips. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*. 250–252.

[15] Xiaoyu Sun, Weidong Cao, and et al. 2024. Efficient Processing of MLPerf Mobile Workloads Using Digital Compute-In-Memory Macros. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 4 (2024), 1191–1205.

[16] Fengbin Tu and et al. 2022. A 28nm 29.2TFLOPS/W BF16 and 36.5TOPS/W INT8 Reconfigurable Digital CIM Processor with Unified FP/INT Pipeline and Bitwise In-Memory Booth Multiplication for Cloud Deep Learning Acceleration. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*. 1–3.

[17] H.-S. Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P. Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E. Goodson. 2010. Phase Change Memory. *Proc. IEEE* 98, 12 (2010), 2201–2227.

[18] Ping-Chun Wu, Jian-Wei Su, Yen-Lin Chung, and et al. 2022. A 28nm 1Mb Time-Domain Computing-in-Memory 6T-SRAM Macro with a 6.6ns Latency, 1241GOPS and 37.01TOPS/W for 8b-MAC Operations for Edge-AI Devices. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. 1–3.

[19] Ping-Chun Wu, Jian-Wei Su, and et al. 2023. A 22nm 832Kb Hybrid-Domain Floating-Point SRAM In-Memory-Compute Macro with 16.2-70.2TFLOPS/W for High-Accuracy AI-Edge Devices. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. 126–128.

[20] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. 2019. Quantization Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Jinshan Yue, Chaojie He, and et al. 2023. A 28nm 16.9-300TOPS/W Computing-in-Memory Processor Supporting Floating-Point NN Inference/Training with Intensive-CIM Sparse-Digital Architecture. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. 1–3.

[22] Feichi Zhou, Zheng Zhou, Jiewei Chen, Tsz Hin Choy, Jingli Wang, Ning Zhang, Ziyuan Lin, Shimeng Yu, Jinfeng Kang, H-S Philip Wong, et al. 2019. Optoelectronic resistive random access memory for neuromorphic vision sensors. *Nature nanotechnology* 14, 8 (2019), 776–782.