# Towards Floating Point-Based Attention-Free LLM: Hybrid PIM with Non-Uniform Data Format and Reduced Multiplications

Lidong Guo*
Tsinghua University
Beijing, China

Zhenhua Zhu*†
Tsinghua University
Beijing, China

Tengxuan Liu
Tsinghua University
& Infinigence-AI
Beijing, China

Xuefei Ning
Tsinghua University
Beijing, China

Shiyao Li
Tsinghua University
& Infinigence-AI
Beijing, China

Guohao Dai
Shanghai Jiao Tong University
& Infinigence-AI
Shanghai, China

Huazhong Yang
Tsinghua University
Beijing, China

Wangyang Fu†
Tsinghua University
Beijing, China

Yu Wang†
Tsinghua University
Beijing, China

## Abstract

Attention-free large language models (LLMs), such as Mamba and RWKV, have emerged as promising architectures to address the quadratic attention complexity of Transformer models. The inference bottleneck of these models lies in memory-bound matrix-vector multiplications (MVMs) and element-wise multiplications (EWMs). The emerging RRAM/SRAM-based Processing-In-Memory (PIM) architectures have shown great potential to overcome the memory wall problem. However, constrained by the supported data format and operator type, directly adopting PIM architectures for attention-free models faces three challenges: (1) RRAM-based analog PIM architectures perform integer (INT) MVMs using voltage, current, and conductance in the analog domain, limiting their application to the more accurate floating point (FP) data format; (2) SRAM-based digital PIM architectures require additional decoder circuits to support FP format, and the SRAM capacity cannot satisfy the storage requirement of LLMs; (3) When performing EWMs using PIM architectures, only one row/column or the diagonal memory cells are activated, resulting in severe device under-utilization.

To tackle the above challenges, this paper proposes an RRAM and 3D-SRAM-based hybrid PIM architecture with non-uniform data format, achieving FP-based algorithm accuracy, high device utilization, and high energy efficiency. **At the software level**, we first analyze the impact of quantization errors on the accuracy of attention-free LLMs. For the quantization error-insensitive MVM operations, we propose the *PIM-oriented exponent-free non-uniform (PN) data format.* The proposed PN format can be flexibly adjusted to fit the data distribution and approach the accuracy of the FP format using bit-slicing-based full INT operations. For the quantization error-sensitive EWM operations, we introduce the multiplication-free approximated FP multiplications to reduce the additional hardware overhead for PIM. **At the hardware level**, we propose a *hybrid PIM architecture*, including an RRAM analog PIM using shift-and-add for PN-based MVMs, and a 3D-SRAM digital PIM with high utilization for multiplication-free FP-based EWMs. Extensive experiments show that the proposed PIM architecture achieves up to 89× and 16× speedup with 2537× and 12 × energy efficiency improvement compared with GPU and PIM-baseline, respectively, while achieving FP-based algorithm accuracy.

## CCS Concepts

• **Hardware → Emerging architectures**.

## Keywords

Attention-free large language models, Processing-In-Memory

**ACM Reference Format:**
Lidong Guo, Zhenhua Zhu, Tengxuan Liu, Xuefei Ning, Shiyao Li, Guohao Dai, Huazhong Yang, Wangyang Fu, and Yu Wang. 2024. Towards Floating Point-Based Attention-Free LLM: Hybrid PIM with Non-Uniform Data Format and Reduced Multiplications. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '24), October 27–31, 2024, NEW JERSEY, NJ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3676536.3676776

## 1 Introduction

Transformer has emerged as a foundation model architecture for various tasks due to its excellent performance [3], while the self-attention module [23] brings quadratic computational complexity and makes it computationally and memory intensive. Recently, attention-free large language models (LLMs) are proposed to circumvent the costly attention computation [8, 21, 17, 16, 9, 27]. Among them, Mamba [9] and RWKV [16] show remarkable capabilities and receive widespread attention, which adopts the State Space Module (SSM) [10] and linear attention as substitutes for

*Both authors contributed equally to this research.
†Corresponding authors. Email: {yu-wang, fwy2018, zhuzhenhua}@tsinghua.edu.cn


*ICCAD '24, October 27–31, 2024, NEW JERSEY, NJ, USA*

ACM ISBN 979-8-4007-1077-3/24/10
https://doi.org/10.1145/3676536.3676776

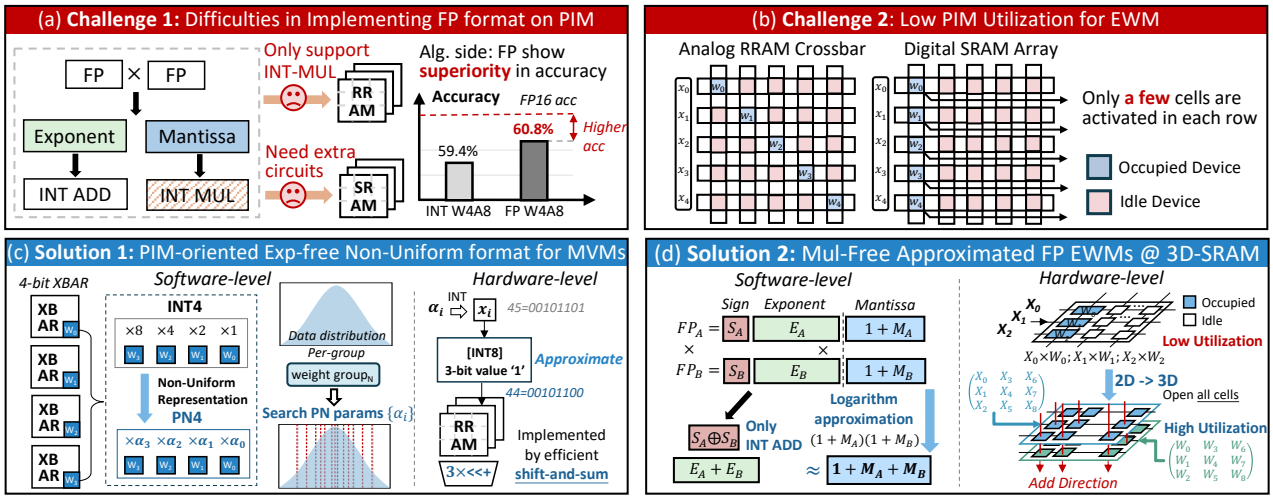



**Figure 1: (a)(b) Challenges of adopting PIM architectures for attention-free models. (c)(d) Our solutions: an RRAM and 3D-SRAM-based hybrid PIM architecture with non-uniform data format and reduced multiplications.**

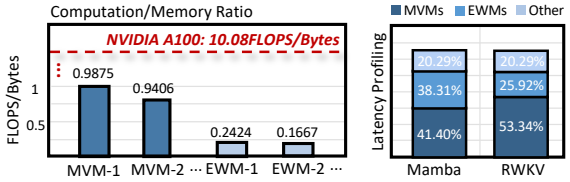



**Figure 2: Memory-bounded MVM and EWM operations become the inference bottlenecks.**

the self-attention module, respectively. Compared to Transformer, Mamba and RWKV realize an efficient Recurrent Neural Network (RNN)-like decoding pattern, maintaining constant computational and memory complexity during inference.

Despite the superiority of attention-free LLMs, these models suffer from the severe memory wall problem in von Neumann architectures. As shown in Figure 2, matrix-vector multiplications (MVMs) and element-wise multiplications (EWMs) are the most time-consuming parts in both Mamba and RWKV models, totally accounting for about 80% of the inference decoding latency. Furthermore, the ratio between computation amount and memory movement of both two operations is extremely low, significantly below the upper bound of GPU's compute intensity (i.e., the amount of computation that can be done per byte of memory movement). This indicates that these operations are bound by massive memory movements and can not fully utilize the computing power of GPUs.

In order to tackle the memory wall problem, researchers have proposed emerging RRAM/SRAM-based Processing-In-Memory (PIM) architectures that perform *in-situ* MVM computations to eliminate the weight matrix data movements, improving performance and energy efficiency [4, 19, 28, 11, 22, 24].

However, limitations exist in PIM architectures in terms of the supported *data format* and *operator type*. Directly adopting existing PIM architectures for attention-free models leads to algorithm accuracy loss and severe under-utilization of computing resources. Firstly, RRAM-based analog PIM architectures are limited to performing inaccurate integer (INT) and fixed-point-based MVMs. Recently, the superiority of non-uniform data format has been validated on LLMs [12, 7], especially in the case of low bit-width. However, RRAM-based PIM architectures perform MVMs in the analog domain, limiting the supported operation to only INT/fixed-point multiplications [22], thereby resulting in significant accuracy loss, as shown in Figure 1(a). Secondly, although SRAM-based PIM architectures can perform MVMs in the more flexible digital domain, additional circuits to support the floating-point (FP) format are required. Besides, the SRAM capacity is limited, making it difficult to meet the requirements of large language models. Thirdly, performing EWMs using MVM-oriented PIM architectures causes extremely low resource utilization. Unlike MVM operations, EWM does not require any accumulation steps. One possible way to perform EWMs using PIM architecture is to store the vector on the diagonal memory cell of RRAM crossbar or one row/column of SRAM crossbar. Then, the calculation is executed in the form of MVM. However, in this case, only a few memory cells are activated in each row/column, causing severe device under-utilization, as shown in Figure 1(b).

To address the above challenges of inaccurate INT data format, hardware costly FP multiplications, and inefficient PIM implementation for EWMs, we propose an RRAM and 3D-SRAM-based hybrid PIM architecture with non-uniform data format and reduced multiplications. Our key contributions include:

- At the software level, we adopt different data formats for different operations based on their quantization error sensitivity. For the quantization error-insensitive MVMs, we propose a novel *PIM-oriented exp-free non-uniform data format (PN)*, enabling near-FP accuracy using bit-slicing-based full INT operations. For the quantization error-sensitive EWM operations, we introduce the multiplication-free FP multiplication approximation method, performing accurate FP multiplications using only INT-addition operations and reducing the additional hardware overhead for PIM.
- At the hardware level, we propose a hybrid PIM architecture for attention-free LLM. An RRAM-based analog PIM using efficient shift-and-add operations is proposed for PN-based MVMs and a 3D-SRAM-based digital PIM with high utilization is adopted for multiplication-free FP-based EWMs.

- We conduct extensive experiments on Mamba/RWKV model families. The results show that the proposed PN format achieves FP-based accuracy with minimal overhead. Besides, the proposed architecture achieves 89× and 16× speedup with 2537× and 12 × energy efficiency improvement compared with GPU and PIM baseline, respectively.

## 2 Background

### 2.1 Attention-Free Large Language Model

Various attention-free models have been proposed to mitigate the quadratic attention complexity of Transformer models. Among them, Mamba [9] and RWKV [16] exhibit performance comparable to the Transformer. As shown in Figure 3, in addition to MVMs of linear layers, a large number of EWM operations are introduced. Specifically, Mamba block adopts the S4 module, which can be defined with four parameters ($\mathbf{\Delta}, \mathbf{A}, \mathbf{B}, \mathbf{C}$) as follows:

$$\begin{aligned} \mathbf{B}, \mathbf{C}, \mathbf{\Delta} &= Linear_{\mathbf{B},\mathbf{C},\mathbf{\Delta}}(\mathbf{x_t}), \\ \mathbf{h_t} &= exp(\Delta \odot \boldsymbol{A}) \odot \mathbf{h_{t-1}} + \Delta \odot \boldsymbol{B} \odot \mathbf{x_t}, \\ \mathbf{y_t} &= \boldsymbol{C}\mathbf{h_t}, \end{aligned} \tag{1}$$

where $Linear(\cdot)$ is the projection of linear layer and the hidden state $\mathbf{h_t}$ representing historical text information is updated each step in the form of element-wise multiplication.

Different from the Mamba model, RWKV leverages past information in the input dimension and adopts a linear attention module. The input feature is generated by a linear interpolation between the current and previous time-step input, followed by linear layers:

$$\mathbf{y} = Linear((\mu \odot \mathbf{x_t} + (1-\mu) \odot \mathbf{x_{t-1}})). \tag{2}$$

Besides, the proposed linear attention module is a variant of AFT module [27], including EWMs and exponential calculations.

### 2.2 Data Format

In this paper, the data format we focus on primarily refers to the data type, i.e., how to use binary encoding to represent a real number. We can use the following equation to describe the relation between the real number's value $V$ and the n-bit encoded binary number $x$:

$$V = f(x) = f(\{x[n-1], ..., x[0]\}). \tag{3}$$

Based on Equation 3, we can categorize data formats into two types, i.e., uniform data formats and non-uniform data formats. The uniform data formats exhibit the characteristic that the value differences between two adjacent encoded binary numbers are the same, as shown in Equation 4. INT and fixed-point data formats are two widely used uniform data formats.

$$\begin{aligned} &\forall\ adjcent\ (x, y),\ f(x) - f(y) = constant, \\ &INT:\ f_{INT}(x) = -x[n-1] \times 2^{n-1} + \sum_{i=0}^{n-2} x[i]2^i. \end{aligned} \tag{4}$$

For the non-uniform data format, the differences between two adjacent encoded binary numbers vary a lot. FP and NormalFloat (NF) [7] are two representative non-uniform data formats:

$$\begin{aligned} &FP:\ f(x) = f(\{S, E, M\}) = (-1)^S \times 2^{E-\text{ bias}} \times (1+M); \\ &NF:\ f(x) = \frac{1}{2}\left(Q\left(\frac{f_{INT}(x)}{2^n+1}\right) + Q\left(\frac{f_{INT}(x+1)}{2^n+1}\right)\right), \end{aligned} \tag{5}$$

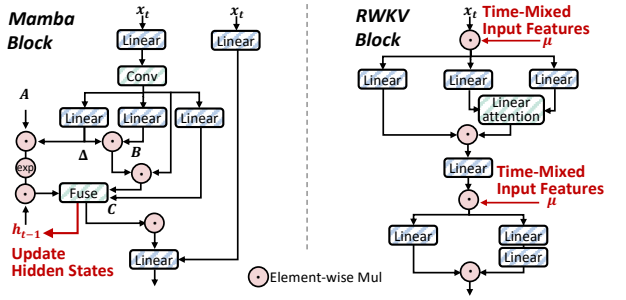
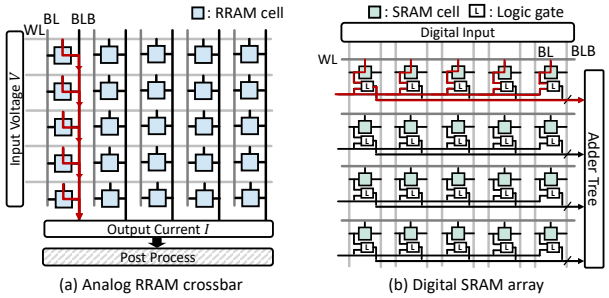



**Figure 3: Overview of Mamba and RWKV block. (LayerNorm module is not shown for clarification)**




(a) Analog RRAM crossbar (b) Digital SRAM array

**Figure 4: Analog RRAM and digital SRAM-based PIM.**

where $S$, $E$, and $M$ are values of the sign, exponent, and mantissa, respectively. $n$ is the bit-width of NF (usually 4-bit), and $Q(\cdot)$ is the quantile function of the standard normal distribution $N(0, 1)$.

### 2.3 Processing-In-Memory Architecture

PIM architectures perform MVM or logic computations inside memory to alleviate the memory wall problem. Existing PIM architectures can be categorized into two types, i.e., analog PIM and digital PIM, as shown in Figure 4.

The analog PIM architectures use voltage, current, capacitance, and conductance to represent the matrix and vector data, and the MVMs are performed in the analog domain according to Kirchhoff's law and Ohm's law [19, 4]. For example, in the RRAM-based PIM, the matrix and vector are represented by the RRAM conductance $\mathbf{G}$ and the word-line voltage vector $\mathbf{V}$, respectively. The MVM results are derived by the output vector $\mathbf{I}$ from bit-lines. Based on the above working principle, analog PIM is mainly suitable for directly representing values using voltage and conductance, and it cannot support FP computation because the FP format requires exponent alignment before mantissa computation.

The digital PIM architectures have been proposed in recent years [5, 25, 22]. In digital PIM architectures, the weight matrix is stored in SRAM array, and the input vector is loaded to the memory array bit-by-bit. The memory cell or the memory row in digital PIM is attached to digital computing units such as logic gates, which perform the *in-situ* bit-wise multiplication between weight bits and input bits. After that, these bit-wise multiplication results are sent to a near-memory adder tree to generate MVM results. Due to the bit-wise operation characteristic, digital PIM architectures can support in-memory FP computation by adding additional alignment and encoder/decoder circuits [22].

**Table 1: Comparison of different data formats for MVMs. W$x$A$y$ means $x$-bit weight and $y$-bit activation. ↓ and ↑ mean lower is better and higher is better, respectively.**

| Mamba-130M | Perplexity ↓ | Accuracy ↑ | Accuracy loss ↓ |
|---|---|---|---|
| FP16 | 16.04 | 64.6 | / |
| INTW8A8 | **17.90** | **63.6** | **1.00%** |
| INTW4A8 | 617.2 | 59.4 | 5.20% |
| FPW4A8 | 65.38 | 60.8 | 3.80% |
| NFW4A8 | 38.05 | 60.8 | 3.80% |

Perplexity: uncertainty of the generated text on LAMBADA dataset [15].
Accuracy: answer accuracy on PIQA question answering dataset [2].

# 3 Software Optimization: PIM-Friendly Data Format and Approximated FP Multiplication

## 3.1 Motivations

In order to deploy attention-free LLMs on PIM architectures, the first step is to determine which data format should be used for computation on PIM. On the one hand, as introduced in Section 2.3, different PIM approaches support different data formats. RRAM-based analog PIM mainly performs INT-based computations, while SRAM-based digital PIM supports FP and other data formats by introducing additional data format processing modules. On the other hand, different operators have varying sensitivity to quantization errors. Therefore, to achieve a better tradeoff between accuracy and computational performance, it is necessary to analyze the suitable data format for each operator.

For MVMs in attention-free LLMs, Table 1 shows the comparison results of Mamba-130M's performance across different data formats. Compared to the FP16-based baseline, INT W8A8 can achieve comparable accuracy, showing that MVMs are quantization error-insensitive. However, if a lower bit-width INT data format, e.g., W4A8, is adopted to reduce the hardware overhead of PIM, more accuracy loss occurs compared to other data formats with the same bit-width. The superiority of FP and NF formats comes from the characteristics of non-uniform data representation. As shown in Figure 5, the INT format has uniform numerical representation intervals, while FP and NF formats exhibit non-uniform representation. Given that the weights in pre-trained models usually have a normal distribution with variable deviation, the non-uniform data formats fit the weights better, bringing smaller quantization errors.

For EWMs in attention-free LLMs, we observe that they are highly sensitive to quantization error. Specifically, adopting the INT format for EWM operation results in a notable drop in model performance, even with a relatively high bit-width of 8-bit, as shown in Table 2. The performance will be worse with a lower bit width, which is unacceptable for natural language tasks demanding high accuracy. The reason for the performance drop is that the inputs of EWMs are dynamically variable and have a wide dynamic range, resulting in larger quantization errors.

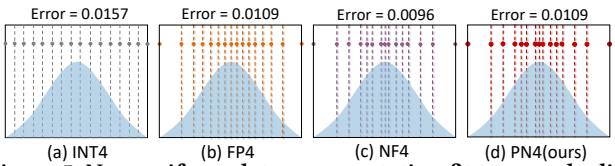



**Figure 5: Non-uniform data representation fits unevenly distributed weight better.**

**Table 2: Comparison of different data formats for EWMs**

| Mamba-130M | HellaSwag acc↑ | PIQA acc↑ | Arc-E acc↑ |
|---|---|---|---|
| Base | 35.292 | 64.635 | 48.022 |
| INT-W8A8 | 30.567 | 56.392 | 35.183 |
| Approx-FP16 | 35.142 | 65.016 | 47.559 |

Inspired by the above observations, our key target is to reduce PIM hardware overhead while maintaining high accuracy as FP-based MVM & EWM computations.

## 3.2 PIM-oriented Exp-free Non-uniform data format towards MVM Optimization

In RRAM-based analog PIM architectures, the multiplication of MVM is performed by the RRAM device via Ohm's law and the accumulation step is achieved through the bit-line current accumulation in the RRAM crossbar. Assume $w$ and $a$ are the conductance matrix and voltage vector, respectively, then the current accumulation results of $N$-row crossbar can be represented by:

$$y_j = \sum_{i=0}^{N-1} w_{ij} a_i. \tag{6}$$

Considering the precision of RRAM device is limited, one RRAM device can only store partial bits of one matrix value. Then, multiple RRAM devices in the same row form the encoded binary number of one weight according to Equation 3, e.g., $W_{ij} = f(\{w_{ij}[n-1], ..., w_{ij}[0]\})$ and $w_{ij}[l]$ is stored in the $l$-th RRAM device. When selecting a proper data format for the weight matrix in PIM, we should guarantee that the MVM results can be obtained by combining the accumulated currents from different crossbar columns. For example, when computing MVM between weight $W$ and activation $a$, the INT format can be used to derive the correct results:

$$\begin{aligned} Y_j &= \sum_i W_{ij} a_i = \sum_i (\sum_l w_{ij}[l] 2^l) a_i \\ &= \sum_l (2^l \sum_i (w_{ij}[l] a_i)) = \sum_l 2^l y_j[l]. \end{aligned} \tag{7}$$

While for the FP format, the encoded binary numbers should be aligned at the exponent bits before mantissa multiplication and addition. Therefore, the direct current accumulation of $w_{ij}[l]a_i$ from different crossbar rows can not yield correct MVM results, causing the analog PIM to be difficult to support non-uniform data formats like FP and NF.

**PN Format Definition (Figure 6(a)).** In order to reduce quantization error using non-uniform data format and maintain the bit-slicing computing paradigm of PIM (i.e., Equation 7), we propose a PIM-oriented non-uniform data format (PN) $f_{PN}$:

$$f_{PN}(x) = \sum_l w[l] \alpha_l, \tag{8}$$

where $\{\alpha_l\}$ are the scaling factors. The key idea of the proposed PN format is to replace the bit factors of $\{2^l\}$ in the INT format with a set of non-uniform and adjustable bit factors, i.e., $\{\alpha_l\}$ in Equation 8. By selecting proper values of $\{\alpha_l\}$, the PN format can be adjusted to fit the data distribution and reduce quantization error similar to FP and NF formats. As examples shown in Figure 5 and Figure 6, the PN format can adapt to normally distributed weight data better, reducing quantization errors by more than 30%.

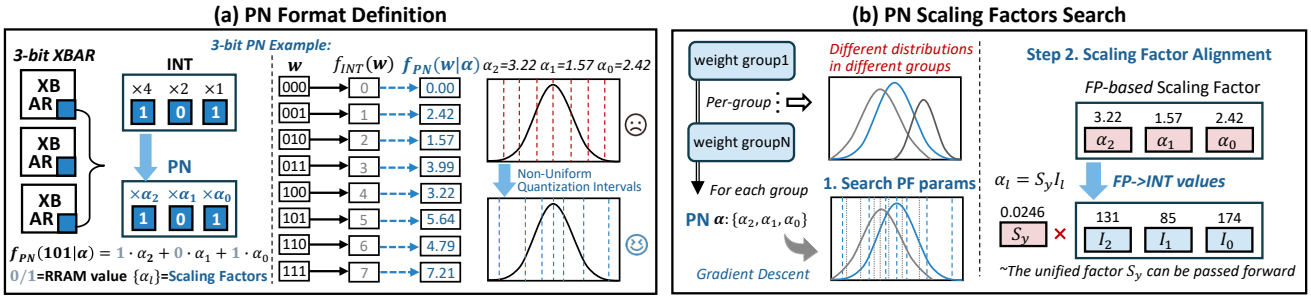



**Figure 6: Overview of PIM-oriented Exp-free Non-uniform data format: (a) PN Format Definition; (b) PN Scaling Factors Search.**

More importantly, with the proposed PN format, the equivalence and correctness of the bit-slicing-based multiply-accumulation flow in analog domain remain as follows:

$$\begin{aligned} Y_j &= \sum_i W_{ij} a_i = \sum_i (\sum_l w_{ij}[l]\alpha_l) a_i \\ &= \sum_l (\alpha_l \sum_i (w_{ij}[l] a_i)) = \sum_l \alpha_l y_j[l]. \end{aligned} \tag{9}$$

The only difference from original architecture is that the accumulated output of the $l$-th crossbar needs to be multiplied by the scaling factor $\alpha_l$ rather than the fixed value $2^l$.

**PN Scaling Factors Search (Figure 6(b)).** With the proposed PN data format, we can obtain an accurate data representation if the scaling factors are set properly. Thus, the most important thing for the PN format is to select the appropriate scaling factors, which is defined as follows:

$$\min_{\boldsymbol{\alpha}} \mathcal{L}\left(f_{PN}(\boldsymbol{W} \mid \boldsymbol{\alpha}), \boldsymbol{W}\right), \tag{10}$$

where $\boldsymbol{\alpha}$ denotes the bit-level scaling factors of PN format and $\mathbf{W}$ denotes the model weights. $\mathcal{L}$ is the loss function representing the difference between two sets of data. The details of the entire search process for scaling factors are shown in Algorithm 1.

Due to the large range of weight values in the whole model, adopting the same PN scaling factors for all the weights incurs accuracy loss. In contrast, within a subset of model weights (a weight group), both the number of weights and the range of values are smaller, enabling lower representation error. Inspired by such observation, we adopt the group-based search scheme to further exploit the advantage of the proposed PN format. We partition the weight tensor into several groups with the same number of values (Algorithm 1 **line 1**) and search different PN scaling factors for different groups using gradient descent (Algorithm 1 **line 5~10**). It is worth mentioning that we only adopt the PN format for weights, while the activation values are still represented by the INT format. So the search process can be completed offline, i.e., post-training data format conversion, eliminating runtime overhead.

**Algorithm 1** PN Parameter Search with Scaling Factor Alignment

**Input**: Pretrained model weights $\mathcal{W}$, group size $g$, bit-width $n$.
**Output**: PN parameter list $\mathcal{P}_{FN}$={$S_y, \mathbf{I}$}$_{i\to N}$.

1: Partition $\mathcal{W}$ into $N$ groups ($\mathcal{W}_{1\to N}$) of group size $g$
2: **for** $i = 1 \to N$ **do**
3: $\mathcal{D}_{target}$ = GetOptimalDistribution($\mathcal{W}_i$)
4: $\alpha_i$ = InitPNParams($n$)
5: **while** True **do**
6: $\mathcal{D}_i$ = GetFormatDistribution($\alpha_i$, $f_{PN}$)
7: $Loss_i = \mathcal{L}(\mathcal{D}_{target}, \mathcal{D}_i)$
8: $\alpha_i$ = GradientDescentUpdate($\alpha_i$, $Loss$)
9: **if** $Loss_i < Thres$ **then**
10: Break
11: $S_{yi}, \mathbf{I}_i$ = INTmodeAlign($\alpha_i$)
12: Add $S_{yi}, \mathbf{I}_i$ to $\mathcal{P}_{FN}$

With the PN format defined by the searched parameters, the direct outputs of $l$-th RRAM crossbar need to be multiplied by the scaling factor $\alpha_l$. To achieve high computing accuracy, the scaling factor $\alpha_l$ should be FP-based data, which brings excessive FP multiplication and addition overhead. To solve this problem, we propose the INT-mode alignment method to convert the FP-based scaling factors to INT values (Algorithm 1 **line 11**):

$$\alpha_l = S_y I_l, \tag{11}$$

where $I_l$ is an 8-bit INT value and $S_y$ is a unified FP-based factor to align the scaling factors with FP/INT format. Since $S_y$ is shared among all weight groups in one layer, it is not involved in the actual forward process of the PIM architecture. Instead, it can be passed forward and processed at the output of the entire model, thereby achieving full INT MVM operations during the runtime.

## 3.3 Mul-free FP Operation Approximation towards EWM Optimization

Due to the significant accuracy loss when adopting 8-bit INT format for EWMs, we conduct EWMs in 16-bit FP format and explore a more efficient implementation. As shown in Figure 1(d), an FP multiplication can be divided into three parts: XOR between the sign bits, INT-addition between the exponent bits, and INT-multiplication between the mantissa bits. Among them, the INT-multiplication operation incurs the highest overhead.

In order to reduce the overhead of INT-multiplications, we introduce the logarithmic approximation method based on $log_2(1+k) \cong k$ to transform the INT-multiplication of mantissa bits to efficient INT-addition operations [14]. Given two FP numbers $A$ and $B$, the multiplication of the mantissa bits can be expressed as:

$$1 + M_{AB} = (1 + M_A) \times (1 + M_B). \tag{12}$$

The logarithm of $M_{AB}$ is computed as:

$$\log_2(1 + M_{AB}) = \log_2(1 + M_A) + \log_2(1 + M_B) \cong M_A + M_B. \tag{13}$$

Then we apply the anti-logarithm approximation, $2^k \cong k + 1$, to approximate the calculation of $M_A + M_B$:

$$1 + M_{AB} \cong 2^{M_A + M_B} \cong 1 + M_A + M_B, \tag{14}$$

It is necessary to consider the overflow case for the mantissa calculation. If $M_A + M_B \geq 1$, the mantissa bits need to shift right

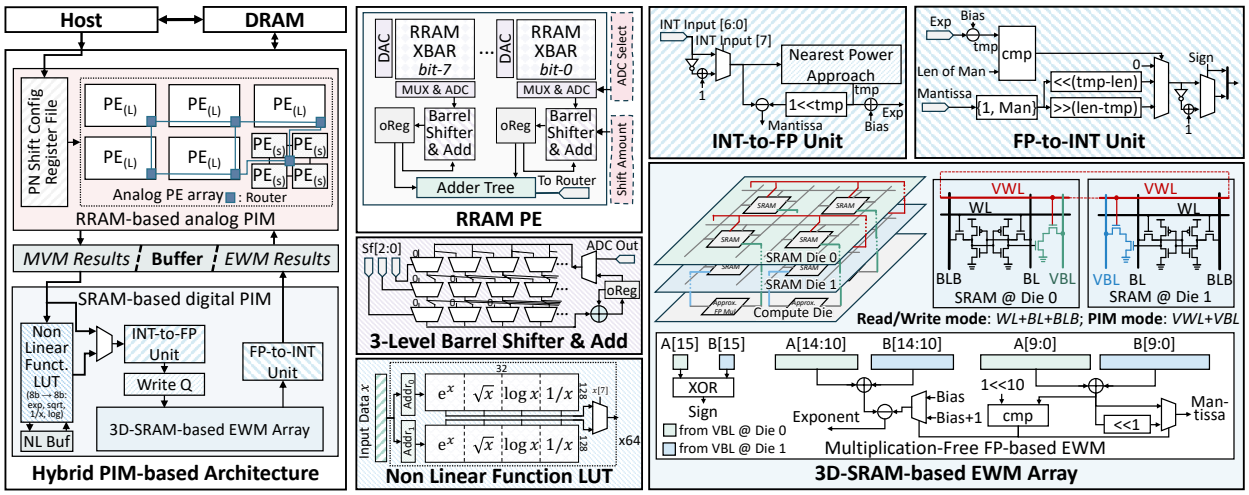



**Figure 7: The proposed hybrid PIM architecture includes: RRAM-based analog PIM using shift-and-add for PN-based MVMs; 3D-SRAM-based digital PIM for multiplication-free FP-based EWMs; LUT-based non-linear function unit and buffer.**

for one bit and the exponent value is added by one. Finally, FP-multiplication can be performed using only efficient INT-addition operations, which can be easily deployed on digital PIM.

To verify the feasibility of the FP-multiplication approximation method, we further analyze the representation error and evaluate the model's accuracy. The error between the approximated multiplication and the actual result is as follows:

$$\Delta_{AB} = AB \cdot \frac{M_A M_B}{1 + M_A + M_B + M_A M_B}. \tag{15}$$

The accuracy results shown in Table 2 demonstrate that the FP multiplication approximation for EWM operations incurs almost no accuracy loss (i.e., <0.5%) while adopting 8-bit INT format results in a larger performance drop (i.e., 10%).

# 4 Hardware Design

## 4.1 Overall Architecture

At the hardware level, we propose a hybrid PIM architecture for attention-free LLM, supporting both PN-based MVM operations and multiplication-free FP-based EWM operations, as shown in Figure 7. The overall architecture mainly includes three parts: the RRAM-based analog PIM part, the SRAM-based digital PIM part, and a buffer for intermediate data. **The RRAM-based analog PIM part** uses mixed-size RRAM processing elements (PEs) for MVM operations with different dimensions to achieve high device utilization rates for different workloads. Specifically, the large RRAM PEs use 256×256 RRAM crossbars for linear layers and the small RRAM PEs use 16×16 crossbars for one-dimensional convolution layers. **The SRAM-based digital PIM part** mainly contains four components, i.e., an SRAM-based Look-Up-Table (LUT) for non-linear functions, a two-layer 3D-SRAM-based EWM array, and two data format conversion units for transferring data between MVM and EWM. **The buffer** stores fixed parameters for EWMs, the hidden states that need to be updated, and the intermediate results from MVM array and EWM array. The MVM and EWM arrays communicate with each other via the buffer.

## 4.2 RRAM-based Analog PIM for PN Format

As shown in Figure 7, the RRAM PEs are interconnected through a 2D-mesh-based Notwork-on-Chip (NoC). Each PE is adjacent to a router with data merging functionality, which receives and merges the data from other PEs, and outputs the result to other PEs or the buffer. A PE contains multiple RRAM crossbars corresponding to different bits of a weight matrix. In the existing INT-oriented RRAM-based PIM, the outputs of each crossbar need to be right-shifted based on the position of stored weight bits (Equation 7), and then summed to obtain final results.

As introduced in Equation 9 and 11, to implement the PN format in RRAM PE, the ADC outputs of each crossbar should be multiplied with INT scaling factors $\{I_l\}$ before summation. The additional INT multiplier incurs extra hardware overhead compared to the shifter in the existing PIM. In order to reduce the multiplication overhead, we propose the shift-and-add-based approximation method. We approximate $I_l$ as an 8-bit data consisting of three-bit "1"s and five-bit "0"s. Then, the multiplication is converted into three consecutive shift-and-add operations:

$$y \cdot I = \sum_{k=0}^{2} y << m, \ where \ I[m] = 1. \tag{16}$$

The circuit implementation is shown in Figure 7. Since the original RRAM-based PIM architecture also includes shifters and adders, the main additional overhead is lightweight 3-stage barrel shifters. The shift amount is read from the PN shift configuration register file, which is configured offline after the PN scaling factors search.

## 4.3 3D-SRAM-based EWM Array

Existing SRAM-based digital PIM architectures are mainly designed for MVMs. As a result, the weight data stored in one SRAM row is simultaneously multiplied with the same input vector value, enabling high column-wise computing parallelism. However, when performing EWM using these digital PIM architectures, each matrix data should be element-wise multiplied with the corresponding input data. In this case, only part of SRAM columns can be activated, resulting in lower computational utilization (e.g., ~30%).

**Table 3: The comparison of algorithm accuracy on Mamba and RWKV model. W$x$A$y$ means $x$-bit weight and $y$-bit activation.**

| Mamba | Method | Hella./PIQA/Arc-E/Wino. acc | Avg↑ | RWKV | Method | Hella./PIQA/Arc-E/Wino. acc | Avg↑ |
|---|---|---|---|---|---|---|---|
| 130M | FP16 | 35.29/64.64/48.02/52.02 | 49.99 | 169M | FP16 | 32.24/64.20/47.60/50.83 | 48.72 |
| | INT-W8A8 | 35.03/63.44/45.88/52.49 | 49.21 | | INT-W8A8 | 32.13/65.29/47.35/52.33 | 49.27 |
| | PN-W8A8 | 35.07/63.49/47.09/52.41 | **49.52** | | PN-W8A8 | 32.17/64.42/47.01/53.59 | **49.30** |
| 370M | FP16 | 46.45/69.48/55.01/55.25 | 56.55 | 430M | FP16 | 40.77/68.06/52.31/53.28 | 53.61 |
| | INT-W8A8 | 46.17/69.21/54.08/55.88 | **56.33** | | INT-W8A8 | 40.86/67.46/52.40/51.62 | 53.08 |
| | PN-W8A8 | 46.14/68.28/53.66/54.46 | 55.64 | | PN-W8A8 | 40.77/68.17/52.74/52.09 | **53.44** |
| 1.4B | FP16 | 59.08/74.16/65.49/61.17 | 64.97 | 1.5B | FP16 | 52.90/72.03/60.90/55.01 | 60.23 |
| | INT-W8A8 | 59.20/73.39/64.14/58.56 | 63.82 | | INT-W8A8 | 51.15/70.57/56.61/53.91 | 58.06 |
| | PN-W8A8 | 58.94/73.67/65.07/59.04 | **64.18** | | PN-W8A8 | 52.84/71.71/60.35/53.99 | **59.72** |
| 2.8B | FP16 | 66.09/75.08/69.69/63.06 | 68.48 | 3B | FP16 | 59.93/74.37/64.73/58.80 | 64.46 |
| | INT-W8A8 | 65.84/73.56/69.07/63.06 | 67.88 | | INT-W8A8 | 59.78/73.50/64.14/56.35 | 63.44 |
| | PN-W8A8 | 65.94/74.86/69.19/63.38 | **68.35** | | PN-W8A8 | 59.64/72.80/63.93/58.25 | **63.66** |

Inspired by the emerging high-density 3D-SRAM technology [1, 20], we propose the 3D-SRAM-based EWM array, consisting of two SRAM dies and one base logic compute die. By vertically transferring data and performing multiplication-free FP computations, the SRAM utilization rate is improved to nearly 100%. As shown in Figure 7, we use an 8T SRAM cell structure to construct the proposed 3D-SRAM. Besides the existing word-line (WL), a vertical WL (VWL) is added, and VWLs of the two stacked SRAM dies are connected vertically for each row. Additionally, each SRAM cell is augmented with a vertical bit-line (VBL) for digital PIM computation. The VBLs are connected vertically to the bottom compute die. It is worth noting that since only one VBL is added per SRAM cell, SRAM die 0 adds the VBL on the right side of 6T-cell, while SRAM die 1 on the left side, reducing layout and routing complexity.

The 3D-SRAM exhibits two operating modes. In the traditional memory read/write mode, the WL and BL(BLB) of the two SRAM dies are independently activated, allowing simultaneous writing of two operands for EWM operations. In the digital PIM mode, the VWL and VBL are activated. The 32-bit operands data (16 bits for each operand) from the two dies are sent via VBL to the multiplication-free FP computation unit in the compute die for calculation, as shown in Figure 7.

## 4.4 Other Components

Besides the core units for MVM and EWM, the proposed hybrid PIM also contains a LUT-based non-linear function unit, as illustrated in Figure 7. We implement four kinds of basic non-linear functions with 8-bit inputs and 8-bit outputs to cover the computation requirements in attention-free LLMs. To improve computation parallelism, we duplicate the non-linear function LUT into 64 copies.

Considering the difference in data formats between MVM and EWM, we add INT and FP conversion units in the SRAM-based digital PIM, placed at the input and output of the 3D-SRAM, respectively. The detailed circuit designs of the data format conversion units are shown in Figure 7. We implement these units using combinational logic circuits to reduce the time and hardware overhead.

# 5 Evaluation

## 5.1 Experimental Setup

**Benchmarks.** We evaluate the Mamba and RWKV model families with different available sizes on a range of popular downstream zero-shot evaluation datasets, including HellaSwag [26], PIQA [2], Arc-Easy [6], and Winogrande [18]. The experiments are conducted with different bit-widths and data format configurations.

**Table 4: Hardware configurations.**

| GPU Platform: NVIDIA A100 | | | |
|---|---|---|---|
| Frequency | 1.41GHz | Bandwidth | 2039GB/s |
| Peak performance | 19.5TFLOPS | Off-chip memory | 80GB |
| **RRAM-based Analog PE Array** | | **3D-SRAM-based EWM Array** | |
| RRAM xbar size | 256×256; 16×16 | 3D-SRAM size | 2×64×64 |
| Read latency | 3.16ns | Read latency | 1.25ns |
| ADC | 8-bit; 1.3GSps | # INT-to-FP unit | 64 |
| # ADC | 1 ADC @ 8 cols | # FP-to-INT unit | 64 |
| Digital frequency: 1GHz; Buffer size: 192KB; LUT capacity: 64KB | | | |

**Baselines.** We compare the proposed hybrid-PIM architecture with NVIDIA A100 GPU and an RRAM-based PIM baseline [28]. For the latter, both MVM and EWM operations are performed in RRAM crossbars with INT format. Since some EWM's inputs are intermediate results from the previous layer, we take the overhead of RRAM device writing into account.

**Methodology.** The hardware configurations are summarized in Table 4. The numbers of RRAM PEs and 3D-SRAM-based EWM arrays depend on the specific model size. The RRAM PEs are simulated using MNSIM 2.0 simulator [29]. We implement and synthesize the digital circuits by Synopsys Design Compiler® at TSMC 28nm technology node. We use CACTI [13] for buffer and LUT simulation.

## 5.2 Algorithm Accuracy Evaluation

We report the accuracy results on both Mamba and RWKV model families in Table 3. For the PN-W8A8 method, weights and activations are represented by 8-bit PN and INT format, respectively. As Table 3 shows, the proposed PN format consistently outperforms the INT format on different types and sizes of attention-free models, while achieving near-FP accuracy.

**Table 5: Comparison results in W4A8 configuration.**

| Model | Method | Hella./PIQA/Arc-E/Wino. acc | Avg↑ |
|---|---|---|---|
| Mamba | FP-W4A8 | 49.55/67.91/55.26/56.80 | 57.38 |
| | INT-W4A8 | 49.25/67.80/53.60/55.62 | 56.57 |
| | PN-W4A8 | 49.85/68.66/55.85/56.41 | **57.69** |
| RWKV | FP-W4A8 | 45.46/68.38/54.12/53.00 | 55.24 |
| | INT-W4A8 | 44.69/67.56/52.35/52.41 | 54.25 |
| | PN-W4A8 | 45.33/68.51/53.80/53.77 | **55.35** |

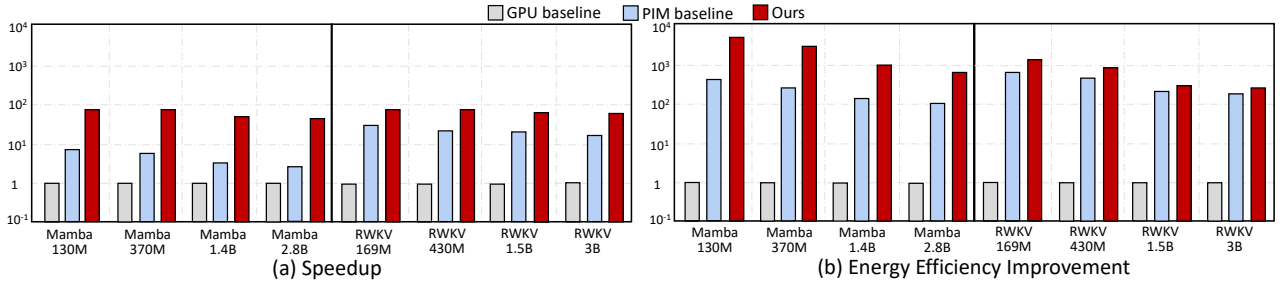



**Figure 8: Comparison results for speedup and energy efficiency improvement on Mamba and RWKV model families. The overall 40.8~65.9× speedup and 235.7~5023.1× energy efficiency improvement compared to the GPU baseline.**

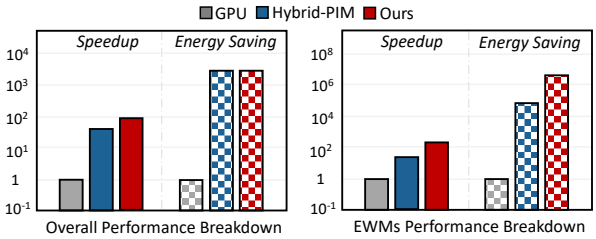



**Figure 9: Comparison with the hybrid-PIM baseline.**

We further evaluate the average accuracy of the proposed PN format with W4A8 configuration in Table 5. Compared to the INT format, the proposed PN format achieves 1.12% and 1.10% higher average accuracy on Mamba and RWKV models, respectively. Additionally, the average accuracy of PN format is even slightly better than that of the FP format. The accuracy gains come from the group-based PN scaling factors search, which enables a more accurate representation based on the weight distribution within each group.

## 5.3 Hardware Performance Evaluation

We compare the proposed architecture with GPU and PIM baseline in terms of speedup and energy efficiency. As shown in Figure 8, compared to the performance of GPU baseline, the PIM baseline achieves 3.1~26.1× speedup and 65.7~559.1× energy efficiency improvement. Nonetheless, it exhibits inefficiency in INT format EWM implementations and incurs ~3% more accuracy loss. By applying the multiplication-free FP multiplication method and the hybrid PIM design, the proposed architecture further achieves 49.9~89.6× speedup compared to GPU implementation, as well as 167.3~2537.1× energy efficiency improvement.

We also compare with a hybrid PIM baseline, which adopts ReDCIM [22], a digital-SRAM-based architecture supporting FP format, to perform the EWMs. The MVM operations are still performed using the base RRAM-PIM design. As shown in Figure 9, the proposed architecture achieves up to 3.1× speedup with comparable overall energy efficiency. Since the two architectures only differ in the implementation of EWM operations, we evaluate the performance of EWMs specifically, and our architecture achieves 6.1× speedup and 54.8× energy savings, which validate the efficiency of approximated FP-multiplication and 3D-SRAM architecture.

## 5.4 Hardware Overhead

We evaluate the area and power consumption of one RRAM-based analog PE and the 3D-SRAM-based EWM array, respectively, as shown in Table 6 and Table 7. The overhead of the shifters and adders introduced by the PN format implementation is negligible, accounting for 1.28% of the total area and 0.83% of the total power. Besides, the introduced format conversion units account for 25.13% of the total area, with a power consumption less than 20%. For the buffer and LUT, the area is $0.44mm^2$ and $0.13mm^2$ with the power consumption of $129mW$ and $98mW$, respectively.

**Table 6: Area and power breakdown of analog PE.**

| Component | Area($mm^2$) | Breakdown | Power($mW$) | Breakdown |
|---|---|---|---|---|
| XBAR | 0.026 | 16.67% | 4.96 | 0.97% |
| ADC | 0.080 | 51.28% | 494 | 96.26% |
| shift&adder | 0.002 | 1.28% | 4.27 | 0.83% |
| Others | 0.048 | 30.77% | 9.97 | 1.94% |
| Total | 0.156 | 100% | 513.2 | 100% |

**Table 7: Area and power breakdown of EWM array.**

| Component | Area($mm^2$) | Breakdown | Power($mW$) | Breakdown |
|---|---|---|---|---|
| 3D-SRAM | 0.0289 | 74.87% | 9.19 | 83.47% |
| FP-to-INT | 0.0039 | 10.10% | 0.69 | 6.27% |
| INT-to-FP | 0.0058 | 15.03% | 1.13 | 10.06% |
| Total | 0.0386 | 100% | 11.01 | 100% |

## 6 Conclusions

In this work, we propose a hybrid PIM architecture for attention-free large language models. We design a PIM-oriented non-uniform data format, breaking the limitations of uniform data format on the RRAM-based PIM architecture. We also adopt the approximated FP multiplication method to perform element-wise multiplications on PIM efficiently. Besides, a hybrid PIM architecture, including RRAM and 3D-SRAM array, is proposed to implement the above techniques with high device utilization and energy efficiency. Overall, the proposed architecture achieves significant speedup and energy efficiency improvement compared to the GPU and PIM baseline, while achieving FP-based algorithm accuracy.

## Acknowledgments

This work was supported by the National Key R&D Program of China (2023YFB4502200), the National Natural Science Foundation of China (No. 62325405, 62104128, U21B2031, 62204164), Tsinghua EE Xilinx AI Research Fund, Tsinghua-Meituan Joint Institute for Digital Life, and Beijing National Research Center for Information Science and Technology (BNRist).

## References


[1] Dawit Burusie Abdi, Shairfe M Salahuddin, Juergen Boemmels, Edouard Giacomin, Pieter Weckx, Julien Ryckaert, Geert Hellings, and Francky Catthoor. 2023. 3d sram macro design in 3d nanofabric process technology. *IEEE Transactions on Circuits and Systems I: Regular Papers*.

[2] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence* number 05. Vol. 34, 7432–7439.

[3] Tom Brown et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877–1901.

[4] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. 2016. Prime: a novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News*, 44, 3, 27–39.

[5] Yu-Der Chih et al. 2021. 16.4 an 89tops/w and 16.3 tops/mm 2 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. IEEE, 252–254.

[6] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

[8] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. 2022. Hungry hungry hippos: towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*.

[9] Albert Gu and Tri Dao. 2023. Mamba: linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

[10] Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

[11] Chuan-Jia Jhang, Cheng-Xin Xue, Je-Min Hung, Fu-Chun Chang, and Meng-Fan Chang. 2021. Challenges and trends of sram-based computing-in-memory for ai edge devices. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 68, 5, 1773–1786.

[12] Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. 2023. Llm-fp4: 4-bit floating-point quantized transformers. *arXiv preprint arXiv:2310.16836*.

[13] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. 2007. Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0. In *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. IEEE, 3–14.

[14] Zijing Niu, Honglan Jiang, Mohammad Saeed Ansari, Bruce F. Cockburn, Leibo Liu, and Jie Han. 2021. A logarithmic floating-point multiplier for the efficient training of neural networks. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI* (GLSVLSI '21). Association for Computing Machinery, Virtual Event, USA, 65–70. ISBN: 9781450383936. DOI: 10.1145/3453688.3461509.

[15] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

[16] Bo Peng et al. 2023. Rwkv: reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

[17] Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: towards larger convolutional language models. In *International Conference on Machine Learning*. PMLR, 28043–28078.

[18] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Communications of the ACM*, 64, 9, 99–106.

[19] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. 2016. Isaac: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44, 3, 14–26.

[20] Kota Shiba, Mitsuji Okada, Atsutake Kosuge, Mototsugu Hamada, and Tadahiro Kuroda. 2023. A 12.8-gb/s 0.5-pj/b encoding-less inductive coupling interface achieving 111-gb/s/w 3d-stacked sram in 7-nm finfet. *IEEE Solid-State Circuits Letters*, 6, 65–68.

[21] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: a successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*.

[22] Fengbin Tu et al. 2022. Redcim: reconfigurable digital computing-in-memory processor with unified fp/int pipeline for cloud ai acceleration. *IEEE Journal of Solid-State Circuits*, 58, 1, 243–255.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

[24] Yitu Wang, Zhenhua Zhu, Fan Chen, Mingyuan Ma, Guohao Dai, Yu Wang, Hai Li, and Yiran Chen. 2021. Rerec: in-reram acceleration with access-aware mapping for personalized recommendation. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[25] Bonan Yan et al. 2022. A 1.041-mb/mm 2 27.38-tops/w signed-int8 dynamic-logic-based adc-less sram compute-in-memory macro in 28nm with reconfigurable bitwise operation for ai and embedded applications. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 65. IEEE, 188–190.

[26] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

[27] Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. 2021. An attention free transformer. *arXiv preprint arXiv:2105.14103*.

[28] Zhenhua Zhu, Hanbo Sun, Yujun Lin, Guohao Dai, Lixue Xia, Song Han, Yu Wang, and Huazhong Yang. 2019. A configurable multi-precision cnn computing framework based on single bit rram. In *Proceedings of the 56th Annual Design Automation Conference 2019*, 1–6.

[29] Zhenhua Zhu et al. 2023. Mnsim 2.0: a behavior-level modeling tool for processing-in-memory architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42, 11, 4112–4125.