Block-Wise Mixed-Precision Quantization: Enabling High Efficiency for Practical ReRAM-based DNN Accelerators

Xueying Wu, Edward Hanson, Nansu Wang, Qilin Zheng, *Student Member, IEEE*, Xiaoxuan Yang, *Member, IEEE*, Huanrui Yang, *Member, IEEE*, Shiyu Li, *Student Member, IEEE*, Feng Cheng, Partha Pratim Pande, *Fellow, IEEE*, Janardhan Rao Doppa, *Senior Member, IEEE*, Krishnendu Chakrabarty, *Fellow, IEEE*, and Hai (Helen) Li, *Fellow, IEEE.*

***Abstract*—Resistive random access memory (ReRAM)-based processing-in-memory (PIM) architectures have demonstrated great potential to accelerate Deep Neural Network (DNN) training/inference. However, the computational accuracy of analog PIM is compromised due to the non-idealities, such as the conductance variation of ReRAM cells. The impact of these non-idealities worsens as the number of concurrently activated wordlines and bitlines increases. To guarantee computational accuracy, only a limited number of wordlines and bitlines of the crossbar array can be turned on concurrently, significantly reducing the achievable parallelism of the architecture.**

While the constraints on parallelism limit the efficiency of the accelerators, they also provide a new opportunity for fine-grained mixed-precision quantization. To enable efficient DNN inference on practical ReRAM-based accelerators, we propose an algorithm-architecture co-design framework called Block-Wise mixed-precision Quantization (BWQ). At the algorithm level, BWQ-A introduces a mixed-precision quantization scheme at the block level, which achieves a high weight and activation compression ratio with negligible accuracy degradation. We also present the hardware architecture design BWQ-H, which leverages the low-bit-width models achieved by BWQ-A to perform high-efficiency DNN inference on ReRAM devices. BWQ-H also adopts a novel precision-aware weight mapping method to increase the ReRAM crossbar's throughput. Our evaluation demonstrates the effectiveness of BWQ, which achieves a $6.08\times$ speedup and a $17.47\times$ energy saving on average compared to existing ReRAM-based architectures.

***Index Terms*—DNN Acceleration, Processing-in-Memory (PIM), Resistive Random Access Memory (ReRAM), Model Compression, Mixed-Precision Quantization.**

I. Introduction

Resistive random access memory (ReRAM)-based Processing-in-memory (PIM) architectures can perform in-situ computation within the memory devices, and they have demonstrated great potential in accelerating DNN training/inference [1, 2]. However, the manufacturing technology of ReRAM devices is still in its early stage, and there exist many challenges to its practical adoption [3]. Most of the works about ReRAM-based DNN accelerators have overlooked practical considerations and rely on an idealized assumption regarding ReRAM devices and associated analog-to-digital converter (ADC) overhead. They assume that it is possible to activate all the rows and columns of a 128×128 or 256×256 array simultaneously within a single clock cycle without impacting computational accuracy[4, 5]. However, there are several challenges that render this assumption impractical. The major problem is the conductance variation of ReRAM devices. Since ReRAM crossbar arrays leverage Kirchhoff's Current Law to perform vector-matrix multiplication (VMM) operations, the conductance variation accumulated along the bitlines (BLs) is proportional to the number of concurrently activated wordlines (WLs) [6]. Activating too many WLs simultaneously also leads to high BL current, which would induce significant IR-drop and cause non-uniform voltage and current distribution along the crossbar [7]. Therefore, to achieve high-accuracy computation, the number of WLs that can be activated within a crossbar array simultaneously should be limited. Another challenge is that for practical ReRAM-based DNN accelerators, the number of ADCs for each crossbar array should be restricted as they consume a significant amount of power and area [5, 8]. As such, it is

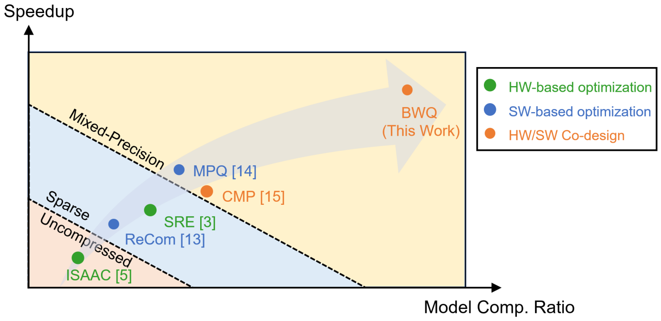


Fig. 1. Research about ReRAM-based DNN accelerators from different optimization perspectives.

This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CNS-1955196, Grant CNS-2233808, Grant EECS-2023752, Grant CNS-1822085, and Grant CSR-1955353. Xueying Wu, Edward Hanson, Nansu Wang, Qilin Zheng, Shiyu Li, Feng Cheng and Hai Li are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: xw221@duke.edu). Xiaoxuan Yang is with the Department of Electrical and Computer Engineering of the University of Virginia, Charlottesville, VA, 22904 USA. Huanrui Yang is with the EECS department of UC Berkeley, Berkeley, CA 94720 USA. Krishnendu Chakrabarty is with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA. Janardhan Rao Doppa and Partha Pratim Pande are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99163 USA

necessary to share one ADC among multiple BLs. Given that an ADC can only convert the signals of one BL in a single clock cycle, the number of BLs that can be activated simultaneously should match the number of ADCs in each crossbar [3]. For a practical ReRAM-based DNN accelerator, the VMM on the crossbar arrays should operate at a much finer granularity, termed as an Operation Unit (OU), rather than at the subarray granularity. [3, 9, 10]. It is demonstrated by several recent studies that for a practical ReRAM-based DNN accelerator to attain an acceptable level of inference accuracy, only nine WLs and eight BLs can be turned on concurrently [3, 11, 12].

The above constraints impose significant limitations on the achievable parallelism of ReRAM-based accelerators, as multiple cycles are required to finish the computation with the entire crossbar array. In the OU-based operation scheme, ADC latency dominates the runtime, and ADC contributes mostly to the overall energy consumption due to the increased number of cycles for computation. Therefore, model compression methods are desired to reduce the number of computation cycles and improve the efficiency of the accelerators. Fig. 1 provides a qualitative illustration of the performance of various ReRAM-based DNN accelerators. The speedup of previous studies is scaled considering the effects of the OU-based operation scheme. Basically, these studies can be categorized into three classes according to their optimization perspectives: hardware (HW)-based optimization, software (SW)-based optimization, and HW/SW co-design. HW-based optimization solutions only improve the accelerators' performance from an HW perspective, featuring intra-layer pipeline (ISAAC [5]) or leveraging the natural sparsity of the neural networks with index reordering (SRE [3]). SW-based optimization solution ReCom [13] and MPQ [14] proactively compress the models with algorithms. However, these works fail to take into account the practical constraints of the underlying ReRAM devices and thus are not able to achieve the optimal model compression ratio. HW/SW co-design-based solutions perform optimizations from both HW and SW perspectives and are supposed to achieve the highest performance. However, existing solutions only compress the model with coarse-grained schemes. For example, CMP [15] conducts layer-wise mixed-precision quantization, and its model compression performance is limited. In contrast, our work proposes a block-wise mixed-precision quantization algorithm (BWQ-A) that considers the parallelism constraints of ReRAM-based accelerators. BWQ-A performs mixed-precision quantization at the weight block (WB) level, matching the size of the OU. The WBs are assigned varying bit precisions based on their individual significance, and the precisions are learned throughout the training process. Leveraging a much finer quantization granularity, BWQ-A can achieve a 58.27× weight compression and a 9.47× activation compression on average over the floating-point baseline models with less than 1% of accuracy degradation. At the architecture level, we present a ReRAM-based hardware accelerator BWQ-H, which enables the efficient inference of the models quantized using BWQ-A. Experimental results show that BWQ-H can boost the energy efficiency by 17.47× and reduce the latency by 6.08× on average compared to existing ReRAM-based architectures.

The main contributions of our work include:

- We introduce BWQ-A, a novel mixed-precision quantization algorithm at the weight block level. Leveraging the small quantization granularity, we are able to achieve a higher weight and activation compression ratio under similar accuracy compared to the existing quantization schemes.
- We propose a practical ReRAM-based accelerator with the OU-based operation scheme. BWQ-H enables efficient inference of the models quantized using BWQ-A. It employs a novel precision-aware weight mapping scheme that increases memory utilization within an OU for the mixed-precision quantized models. A memory controller is also designed to facilitate mixed-precision computation within the same crossbar.
- We analyze the scalability of BWQ-A and BWQ-H with larger OU sizes. This serves as a future road map for practical ReRAM accelerators depending on the evolution of the manufacturing technology of ReRAM devices.

The remainder of this paper is organized as follows: In Section II, we discuss the background of ReRAM-based DNN accelerators and review the mixed-precision quantization schemes for model compression. Section III introduces BWQ-A, the block-wise mixed-precision quantization algorithm. Next, Section IV presents the architecture of the hardware accelerator BWQ-H that enables efficient inference of the BWQ-A algorithm. Section V shows the evaluation methodology, and Section VI shows the performance of the algorithm and the architecture. Lastly, Section VII concludes the paper and outlines key points for future works.

II. Background

A. ReRAM-based DNN Accelerators

The potential of ReRAM-based platforms to achieve high performance and energy efficiency computation has made them promising for accelerating DNN inference. ReRAM is a type of non-volatile memory that stores information by changing the resistance of the metal oxide material [4]. In ReRAM-based accelerators, the conductance of ReRAM devices is used to represent the weights of neural networks, and the analog VMM computations are performed inside the crossbar arrays. This allows for highly parallel computation and eliminates the need for data movement between memory and computation units.

Most of the studies on ReRAM-based DNN accelerators have utilized crossbars with sizes of 128×128 or 256×256 [4, 5]. These design choices strike a balance between the throughput and utilization of the ReRAM crossbars. These studies usually assume that it is possible to activate all the rows and columns of the array simultaneously, and the VMM computation with the whole weight matrix on the subarray could finish within a single clock cycle. However, this assumption is ideal. According to the experimental results in [3], activating the entire ReRAM crossbar within one cycle leads to significant accuracy loss and introduces high peripheral circuitry overhead. To guarantee high-accuracy computation, it is necessary to reduce the accumulated conductance variation of the ReRAM devices and IR-drop along the BLs. Thus, the number of concurrently activated WLs should be limited. Additionally, the number of BLs to be turned on concurrently should also be limited to constrain the overhead of the ADCs. According to [8], the ADCs account for 50% $\sim$ 70% of the overall power consumption in a ReRAM-based accelerator. For energy and area efficient implementation, one ADC should be shared among multiple BLs and the number of concurrently activated BLs should also be restricted by the number of ADCs. A practical ReRAM accelerator should operate at the granularity of an OU, which is much smaller than the entire crossbar. Several studies have demonstrated that, in practice, an OU can accommodate a block with only nine WLs and eight BLs [3, 11].

B. Mixed-Precision Quantization

Quantization is a model size reduction technique that converts the floating-point weights into low-precision floating-point or integer formats. Low precision quantization is particularly beneficial for ReRAM-based accelerators as it efficiently reduces the number of columns required to represent each weight, thus reducing the computation cycles and ADC's power consumption. Compared with uniform quantization, mixed-precision quantization can achieve lower average bit precision under similar accuracy levels [16–18]. However, determining the optimal precision for each layer or each channel is a challenging task. Most previous works have either performed manual bit-with selection, which relies on expert knowledge [16], or neural architecture search with reinforcement learning, which requires massive computation [17, 18].

To address the above challenges, Bit-level sparsity quantization (BSQ) [19] proposes a layer-wise mixed-precision quantization scheme that learns the precision of the weights in each layer through a single-pass training process. To achieve an optimal trade-off between model size and accuracy, BSQ proposes exploiting bit-level sparsity by training bit representation of the weights instead of floating-point weights. During training, a bit-level group Lasso regularizer is also incorporated to induce higher sparsity. As evaluated on a diverse range of models and datasets, BSQ is able to achieve higher compression ratios under similar accuracy compared to previous methods [19].

C. Motivation

A practical ReRAM-based accelerator's achievable throughput and energy efficiency are limited by the OU size due to the computation accuracy and the peripheral circuitry overhead concerns. The VMM with the entire weight matrix on a single crossbar array should require multiple ADC cycles as each cycle can only activate a 9×8 OU, which is the maximum size that a state-of-the-art ReRAM accelerator can achieve [11]. BSQ's potential to achieve ultra-low weight precision provides a possible solution for efficient DNN inference on ReRAM-based accelerators, as the low-bit-width models greatly reduce the required ADC cycles and ADC energy consumption. However, BSQ only considers quantization at the layer level. According to Dash et. al. [20], the significance, or the contribution towards the training objective of each weight element within the same layer can vary considerably. BSQ assigns a uniform bit precision to all the weight elements in a layer, which may not lead to the optimal trade-off between accuracy and average bit-width per weight. On the other hand, the constraints for a practical ReRAM accelerator provide a new opportunity for finer-grained mixed-precision quantization. This motivates us to explore a higher weight compression ratio with a new quantization scheme, block-wise mixed-precision quantization algorithm, or BWQ-A. We also propose a corresponding hardware design BWQ-H to enable efficient implementation of the BWQ-A algorithm on ReRAM-based architecture.

III. BWQ-A: Block-Wise Mixed-Precision Quantization Algorithm

BSQ's potential to achieve ultra-low weight precision provides a possible solution for efficient DNN inference on ReRAM-based accelerators under the OU-based operation scheme, as the low-bit-width models effectively reduce the required clock cycles and ADC energy consumption. However, BSQ only performs layer-wise mixed-precision quantization, overlooking the fact that the significance, or the contribution towards the training objective of each weight element within the same layer can vary considerably [20]. On the other hand, the OU-based operation scheme provides a new opportunity for finer-grained mixed-precision quantization. This motivates us to explore a higher weight compression ratio with a novel quantization scheme BWQ-A.

A. Weight Compression

To implement mixed-precision quantization at the block level, we first divide the weights of each layer into multiple 2D WBs with the same size as an OU. For fully-connected layers, we can directly partition the 2D weight tensor with the shape of (C_{out}, C_{in}), as illustrated in Fig. 2 (a). However, for convolutional layers, the weights form a 4D tensor and must be flattened into 2D. We apply the reshaping method discussed in CSP [21] to reshape the convolutional layers. This method transforms a 4D tensor with a shape of (C_{out}, C_{in}, k, k) into a 2D tensor with a shape of $((C_{in} \times k \times k), C_{out})$. Next, we divide the weights in the flattened tensor into WBs (see Fig. 2 (b)).

Inspired by BSQ, we train the weights in their bit-level representations to exploit the bit-level sparsity. The bit-level representation of the weight applied in BWQ-A is defined as:

$$W = sign(W) \odot \frac{s}{2^n - 1} \sum_{b=0}^{n-1} W_s^{(b)} 2^b m^{(b)}, \qquad (1)$$

where n denotes the number of bits, s is the scaling factor, and $W_s^{(b)}$ is the b^{th} bit in the binary representation, which is non-negative. A binary mask $m^{(b)}$ is also introduced to indicate whether a certain bit b in the WB should be removed ($m^{(b)} = 0$) or retained ($m^{(b)} = 1$). To induce higher sparsity

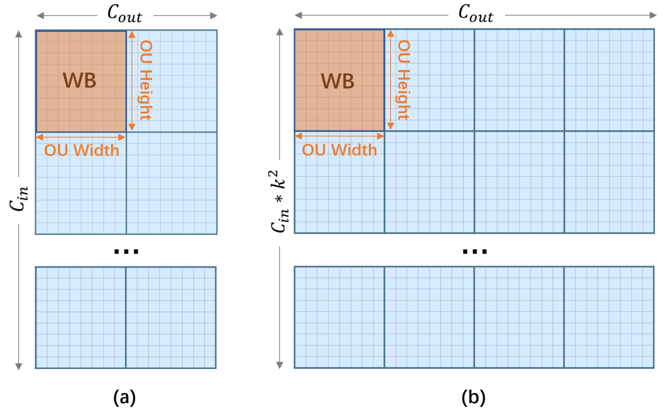


Fig. 2. Partitioning weight layers into weight blocks (WBs). (a) Fully-connected layer. (b) Convolutional layer.

in the model, a WB-level group Lasso is integrated during the training process. The WB-level group Lasso for the r^{th} layer is formulated as:

$$B_{GL}(W^r) = \sum_{g=0}^{G_r-1} \sum_{b=0}^{n-1} ||W_s^{(g,b)} m^{(b)}||_2, \tag{2}$$

where G_r is the total number of WBs in the r^{th} layer, and $W_s^{(g,b)}$ denotes the binary representation of the weights of the b^{th} bit within the g^{th} WB. Applying the WB-level group lasso is able to make a certain bit of all weight elements in the same WB zero simultaneously. This method regularizes the weights in each WB individually, rather than penalizing the weights in the same layer as a whole. The overall training objective is formulated as:

$$L = L_{CE} + \alpha \sum_{r=1}^{R} \frac{\#Param(W^r) \times \#Bit(W^r)}{\#Param(W^{(1:R)})} B_{GL}(W^r), \tag{3}$$

where L_{CE} is the cross-entropy loss, R is the number of layers of the model, and α is the regularization strength. To minimize the total number of bits in the model, the loss function includes coefficients for each WB-level group Lasso. These coefficients impose greater penalties on layers with a higher number of bits.

To enhance the model's resilience against quantization noises, we conduct regular re-quantization followed by adjusting the precision in a block-wise manner at specific intervals throughout the training process. The overall weight compression scheme of BWQ-A is illustrated in Fig. 3 (a). In the re-quantization process, we convert each bit of the weights into exact binary values. The precision adjustment scheme in BWQ-A, which is depicted in Fig. 3 (b), conducts the removal of the zero-valued bits in a block-wise manner. The initial precision for all WBs is set to 8-bit (in the simplified example of Fig. 3 (b), the WBs are initially set to 4-bit), and the initial value of the binary bit mask $m^{(b)}$ for every bit of the weights is 1. During the precision adjustment process, we check each bit of W_s within a WB from the most significant bit (MSB) down to the least significant bit (LSB). If a certain bit for all the weight elements within a WB is always zero, then that bit could be removed in this block by setting the corresponding bit mask to 0. The bit value checking stops until we encounter the first non-zero bit. In this way, the WBs are assigned different precisions according to their individual contribution towards the training objective. As illustrated in Fig. 3 (a), in the forward pass, the weights are updated with the product of the weight tensor and the mask tensor. Therefore, the pruned bits of the weights will remain zero in the subsequent training epochs. This ensures that the sparsity of the model is non-decreasing throughout the training process.

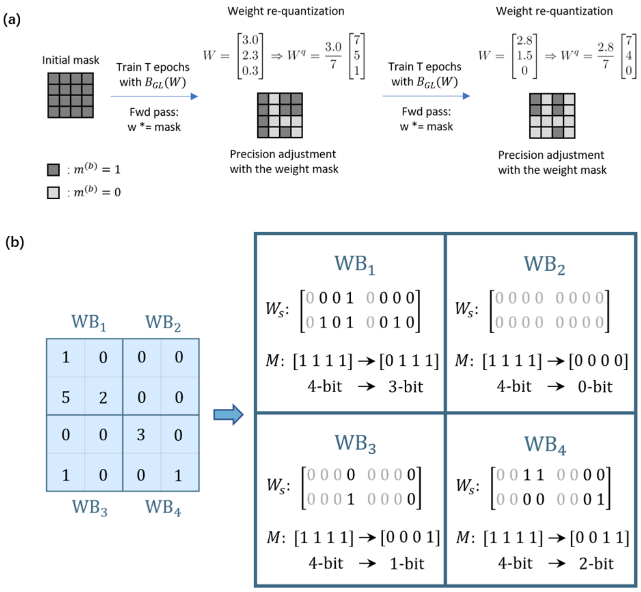

(a)
Initial mask
Train T epochs with $B_{GL}(W)$
Fwd pass: w *= mask
Weight re-quantization
$W = \begin{bmatrix}3.0\\2.3\\0.3\end{bmatrix} \Rightarrow W^q = \frac{3.0}{7}\begin{bmatrix}7\\5\\1\end{bmatrix}$
Precision adjustment with the weight mask
Train T epochs with $B_{GL}(W)$
Fwd pass: w *= mask
Weight re-quantization
$W = \begin{bmatrix}2.8\\1.5\\0\end{bmatrix} \Rightarrow W^q = \frac{2.8}{7}\begin{bmatrix}7\\4\\0\end{bmatrix}$
Precision adjustment with the weight mask
■ : $m^{(b)} = 1$
□ : $m^{(b)} = 0$
(b)
WB_1
WB_2
1 0 0 0
5 2 0 0
0 0 3 0
1 0 0 1
WB_3
WB_4
WB_1
W_s: $\begin{bmatrix}0\,0\,0\,1 & 0\,0\,0\,0\\0\,1\,0\,1 & 0\,0\,1\,0\end{bmatrix}$
M: [1 1 1 1] → [0 1 1 1]
4-bit → 3-bit
WB_2
W_s: $\begin{bmatrix}0\,0\,0\,0 & 0\,0\,0\,0\\0\,0\,0\,0 & 0\,0\,0\,0\end{bmatrix}$
M: [1 1 1 1] → [0 0 0 0]
4-bit → 0-bit
WB_3
W_s: $\begin{bmatrix}0\,0\,0\,0 & 0\,0\,0\,0\\0\,0\,0\,1 & 0\,0\,0\,0\end{bmatrix}$
M: [1 1 1 1] → [0 0 0 1]
4-bit → 1-bit
WB_4
W_s: $\begin{bmatrix}0\,0\,1\,1 & 0\,0\,0\,0\\0\,0\,0\,0 & 0\,0\,0\,1\end{bmatrix}$
M: [1 1 1 1] → [0 0 1 1]
4-bit → 2-bit

Fig. 3. (a) The quantization-aware-training scheme of BWQ-A. (b) Block-wise precision adjustment operation.

Our compression objective is to achieve the highest possible weight compression ratio while ensuring that the quantized model's accuracy does not drop by more than 1%. Therefore, we gradually increase the regularization strength by a fixed interval Δ_α until the accuracy loss is larger than 1%.

B. Activation Compression

$$y = PACT(x) = 0.5(|x| - |x-\beta| + \beta) = \begin{cases} 0, & x \in (-\infty, 0) \\ x, & x \in [0, \beta) \\ \beta, & x \in [\beta, +\infty) \end{cases} \tag{4}$$

To minimize the end-to-end latency of the accelerator, BWQ-A performs activation quantization as well, as reducing the activation bit-width effectively decreases the buffer's read latency. Weight compression and activation compression are two orthogonal processes in BWQ-A, and we predetermine the activation bit-width before each training process. To achieve low-bit-width activations, we apply the Parameterized Clipping Activation Function (PACT) proposed in [22] prior to activation quantization. As shown in Equation 4, PACT removes outliers of the activation values and constrains the activations to a relatively small range, thereby decreasing the quantization noise. We first determine the weight quantization scheme with 8 bit activations and then gradually decrease the

Algorithm 1 Quantization scheme of BWQ-A.

```
Require: $\Delta_\alpha$, Init_Act_Precision, Init_Weight_Precision
Act_Precision ← Init_Act_Precision
$\alpha \leftarrow 0$
M ← 1 /*Initialize the binary mask*/
$W_{fp}$ ← Random_Weight_Init()
$W_b$ ← Bit_Representation($W_{fp}$, Init_Weight_Precision)
/*Determine the weight regularization strength*/
while acc_loss $\leq$ 1% do
    $\alpha \leftarrow \alpha + \Delta_\alpha$
    for epoch=1,..., Total_Training_Epoch do
        train($W_b$, M, $\alpha$, Act_Precision)
        if epoch in Quant_Epochs then
            $W_b$ ← Quant($W_b$)
            M ← Prec_Adjust($W_b$)
        end if
    end for
end while
/*Determine the activation precision*/
while acc_loss $\leq$ 1% do
    Act_Precision ← Act_Precision - 1
    for epoch=1,..., Total_Training_Epoch do
        train($W_b$, M, $\alpha$, Act_Precision)
        if epoch in Quant_Epochs then
            $W_b$ ← Quant($W_b$)
            M ← Prec_Adjust($W_b$)
        end if
    end for
end while
return $W_b$, M, Act_Precision
```

activation precision until the accuracy degradation exceeds 1%. The overall quantization scheme of BWQ-A is illustrated in Algorithm 1.

IV. BWQ-H: Hardware Acceleration Of BWQ-A With ReRAM-PIM

This section describes the implementation of BWQ-H, which is designed for the efficient inference of BWQ-A models. BWQ-H is a practical ReRAM-based PIM accelerator that adopts the OU-based operational scheme. While the OU-based scheme limits the parallelism of the ReRAM crossbars, the confined number of simultaneously activated WLs in an OU allows for a significant reduction in the required ADC precision. Since ADC power consumption increases exponentially with its precision, accelerators with the OU-based operation scheme can achieve more energy-efficient performance by utilizing low-precision ADCs.

The overall architecture of BWQ-H is illustrated in Fig. 4. The accelerator comprises multiple tiles connected through a Network-on-Chip (NoC) and external memory. Within each tile, there are several PIM banks, accumulation units, tile-level input/output registers, functional units, and the local bus. Each PIM bank includes a ReRAM crossbar, input/output register, WL decoder, DACs, MUX, ADCs, S&A units, and the memory controller.

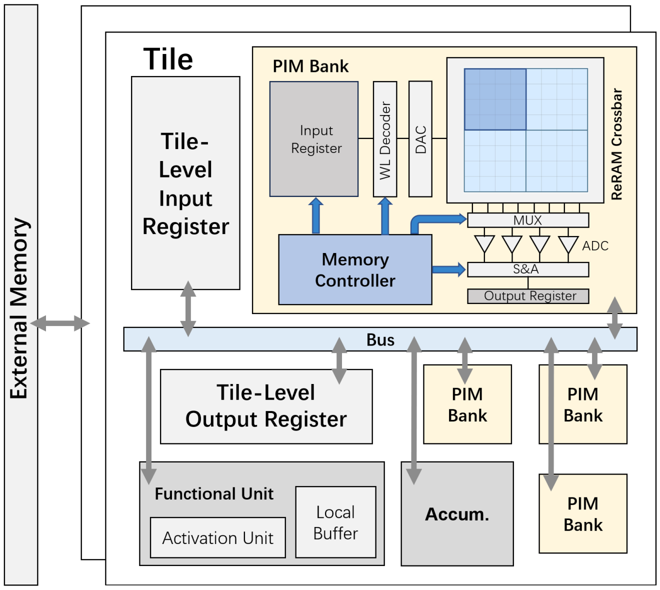


Fig. 4. Architecture of BWQ-H. The grey arrows represent the dataflow, while the blue arrows denote the control flow of the memory controller.

In the remainder of this section, we first introduce a precision-aware weight mapping method that increases the utilization of the ReRAM crossbar arrays for the mixed-precision quantized models. Next, we present the design of a memory controller, which generates control signals for the peripheral circuits to enable the computation with mixed-precision weights on the same crossbar.

A. Precision-Aware Weight Mapping

Fig. 5 provides an illustration of different weight mapping schemes. In this example, a 4×4 WB is mapped to the crossbar consisting of 4×4 OUs. Within the WB, there are 4 3-bit weight vectors A, B, C, and D. Fig. 5 (a) illustrates the traditional weight mapping method. Here, different bits from the same weight vector are mapped in consecutive columns of the crossbar array. For our mixed-precision quantized models, however, this mapping method will lead to an induced peripheral circuit overhead or low OU utilization if the number of columns in an OU is not divisible by the weight precision in a certain block, as shown in Fig. 5 (a) and (b). In Fig. 5 (a), since the second weight vector B spans two OUs, the computation results with $B[0]$ on OU1 and the results with $B[1]$ and $B[2]$ on OU2 should be accumulated. Thus, complex indexing control logic for shift-and-add (S&A) units is required to sum up the computation results with bits of the same weight vector from different OUs activated in different clock cycles. This can cause additional overhead for the peripheral circuitry and result in increased computation latency. To avoid the overhead associated with the complex control logic, one alternative is to only allow the bits of the same weight to be mapped within the same OU, as depicted in Figure 5 (b). However, this can lead to low memory utilization of the OUs, thus leading to throughput reduction. In the given example, each OU would contain a spare column, resulting in a 25% reduction in OU throughput.

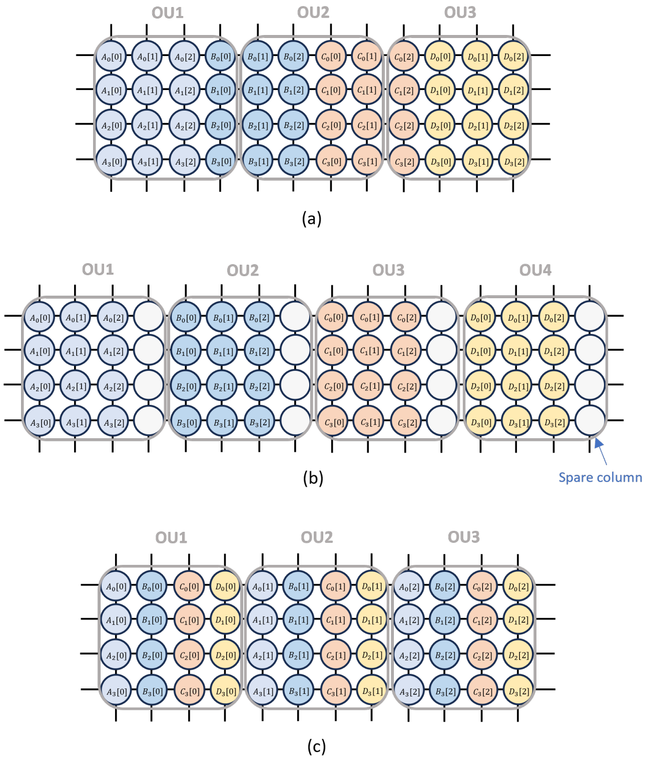


Fig. 5. Example of different weight mapping schemes. Cells with the same color represent the same weight vector, $V_i[n]$ denotes the n^{th} bit of the i^{th} element in vector V. (a) Conventional mapping scheme, in which bits of the same weight element may span different OUs. (b) A mapping scheme that constrains the bits of the same weight element to be mapped on the same OU. The blank cells represent spare columns. (c) Precision-aware mapping scheme.

To enable efficient computation of the mixed-precision quantized models, we propose a precision-aware weight mapping scheme, which is illustrated in Fig. 5 (c). Instead of mapping different bits of the same weight vector in consecutive columns, we assign them to different OUs. Within an OU, we map the same bit of different weights in the 4×4 WB. The computation results with different bits of the same weight vector are accumulated by the S&A units from different OUs. This precision-aware mapping scheme eliminates the need for additional peripheral circuits and is able to achieve 100% memory utilization within an OU.

Fig. 6 (a) shows an example of mapping multiple WBs with different precisions onto the crossbar with the aforementioned scheme. This is a simple example of mapping a weight tensor that contains four 2×2 WBs. First, we convert the weights into their binary representation. Here, WB_1, WB_2, WB_3, and WB_4 are represented with 2, 0, 1, and 2 bits, respectively. Then, precision-aware mapping is conducted by grouping identical bits of the weights in a WB into the same OU. To achieve higher utilization of the crossbar, we skip the spare OUs, and the occupied OUs are densely mapped from left to right. Note that if the total bit-width for each row of WBs varies, there may exist spare OUs in the crossbar (as illustrated by four spare ReRAM cells on the top-right of Fig. 6 (b)). However, this will not impact the throughput of the OUs or result in increased power consumption. This is due to the fact that within one clock cycle, only one OU can be activated, and the computation with unused OUs will be skipped. Weight sparsity on the crossbars under the precision-aware mapping scheme is evaluated in Section VI-D. [5].

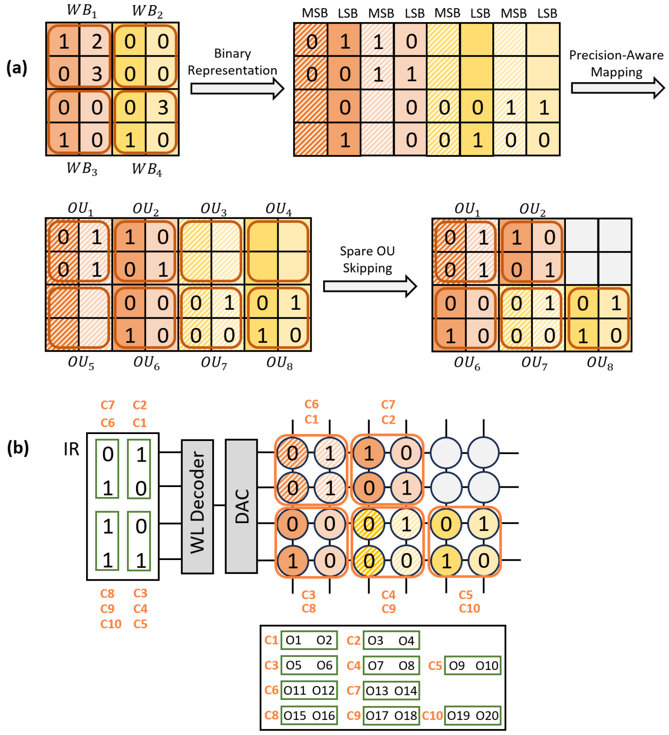


Fig. 6. (a) Mapping four WBs with different precisions to the crossbar array with the precision-aware mapping scheme. (b) An example of mixed-precision computation on BWQ-H. Ci and Oi denote the i^{th} cycle and the i^{th} output, respectively.

B. Memory Controller Design

To enable the computation with mixed-precision weights on the same crossbar, a memory controller is required to generate control signals for the peripheral circuits. The memory controller contains a look-up-table (LUT) to store the bit-width of all the WBs in one PIM bank, and generates control signals to the WL decoder, multiplexer (MUX), the S&A units, and the input register (IR) at the PIM bank level.

The control logic of the memory controller is illustrated in Algorithm 2. To perform OU-based computation with the mixed-precision WBs, the controller (1) generates the row and column addresses for the current OU, and sends the addresses to the WL decoder and the MUX so that the corresponding rows and columns are activated; (2) produces a skip signal for the S&A upon completing the computation across multiple OUs related to one WB, so that the computation results of different WBs are not accumulated; (3) generates an enable signal for the IR once the computation for a row of WBs is finished, so that the corresponding activation data for the next WB row can be fetched from the register and sent to the activated WLs. The control signal flow is denoted by the blue arrows in Fig. 4.

Fig. 6 (b) shows an example of how BWQ-H conducts mixed-precision computations coordinated by the memory controller. In the first cycle C1, the activation data stored in the upper-right section of the IR is sent to the 1st and 2nd WLs, and performs the multiply-accumulate (MAC) operation with OU_1. This results in the outputs of O1 and O2. To reuse the activation data, in C2, we perform the MAC operation with OU_2, and generate O3 and O4. Since OU1 and OU2 store different bits of the same WB, the outputs in C1 and C2 are accumulated by the S&A units. As the four ReRAM cells on the top-right of the crossbar are spare, we skip the computation on these cells and proceed to OU_6. Next, in C3, we fetch the activation data in the bottom-right section of the IR and send it to the 3rd and 4th WLs. In C3, the S&A units receive a skip signal from the memory controller, preventing O5 and O6 from being accumulated with previous outputs. To complete the computation presented in this example, a total of 10 cycles are required. The subsequent computation sequence is depicted in Fig. 6 (b).

Algorithm 2 Control logic of the memory controller.

```
Require: $C_{out}, C_{in}$, k, OU_Width,
    OU_Height, Bitwidth_Table
Num_Hblock ← Ceil($C_{out}$ / OU_Width)
Num_Vblock ← Ceil($C_{in}$ * k * k / OU_Height)
Col_Start_Idx ← 0
Fetch_Next ← 0
for i = 0, 1, ..., Num_Hblock - 1 do
    Fetch_Next ← 0
    for j = 0, 1, ..., Num_Vblock - 1 do
        Weight_Precision ← BW_Table[i-1][j-1]
        if Weight_Precision $\neq$ 0 then
            Psum ← 0
            Activated_Rows ← [(i-1) * OU_Height :
            i * OU_Height]
            for k = 0, 1, ..., Weight_Precision - 1 do
                Activated_Cols ← [Col_Start_Idx :
                Col_Start_Idx + OU_Width]
                Psum ← Shift_Left(Psum) +
                Current_ADC_Output
                Col_Start_Idx ← Col_Start_Idx + OU_Width
            end for
        end if
        if j == Num_Vblock - 1 then
            Fetch_Next ← 1
        end if
    end for
end for
```

V. Evaluation Methodology

To assess BWQ's efficacy, we examine the impact of its two components, BWQ-A and BWQ-H, both separately and in combination. First, we verify the influence of BWQ-A on the model's performance and compression ratio. Then, we evaluate BWQ-H's inference latency and energy overhead with the quantized models obtained by BWQ-A to examine the effectiveness of the proposed co-design scheme.

A. Algorithm Performance Validation

BWQ-A is compared against the floating-point baseline models and BSQ models in terms of accuracy and model compression ratio. We consider a spectrum of representative models on CIFAR-10, CIFAR-100 [23] and ImageNet [24] datasets. For the CIFAR experiments, ResNet-20, ResNet-18, ResNet-34 [25], VGG16-BN, VGG19-BN [26], MobileNetV2 [27] and DenseNet-121 [28] are examined. For the evaluations on ImageNet, we use ResNet-34 and DenseNet-121.

In the CIFAR experiments, the floating-point baseline models are trained for 200 epochs with SGD optimizer with 0.9 momentum and 1e-4 weight decay. The initial learning rate is 0.1, and a cosine annealing learning rate scheduler is utilized. The training for BWQ-A uses the same hyperparameters as the baseline models, except that BWQ-A is trained for 650 epochs because it incorporates a quantization-aware-training approach. This extension of training epochs proves advantageous as it facilitates the models' adaptation to the quantization noises. The number of training epochs is comparable with the total training epochs (training and fine-tuning) used in previous studies such as BSQ [19] and CSQ [29]. The VGG models are re-quantized every 200 epochs, while the ResNet, DenseNet, and MobileNet models are re-quantized every 300 epochs. A final re-quantization and precision adjustment operation is conducted after the training process, finalizing the precisions of the WBs.

For the ImageNet experiments, we use the pretrained models from torchvision [30] as the floating-point baseline models. The BWQ-A models are trained for 90 epochs with SGD optimizer with 0.9 momentum. We set the initial learning rate as 0.001, and use a cosine annealing learning rate scheduler. A weight decay parameter of 1e-4 is incorporated. The models are re-quantized at the 60th and the 90th epoch. All the training processes are conducted using two A5000 GPUs with distributed data parallel.

B. Architecture Modelling and Comparison

For hardware performance, we compare BWQ-H with several representative ReRAM-based accelerators, including a baseline architecture ISAAC [5] with no model compression, SRE [3], BSQ [19] and SME[31], which perform model compression with HW-based, SW-based and HW/SW co-design approaches, respectively. The detailed operational principles of SRE and SME are discussed in Section VI-B. BSQ's hardware performance on a ReRAM-based accelerator is simulated by modifying BWQ-H and following the same simulation method as BWQ-H. We modified the MNSIM [32] simulator to evaluate the performance of BWQ-H. For the analysis of the overhead for implementing the BWQ-A model with the OU-based operation scheme, we build an RTL model in Verilog for the memory controller and synthesize the model using Synopses Design Compiler with TSMC commercial 28nm standard library. Table I shows the hardware configuration to evaluate the BWQ-H architecture.

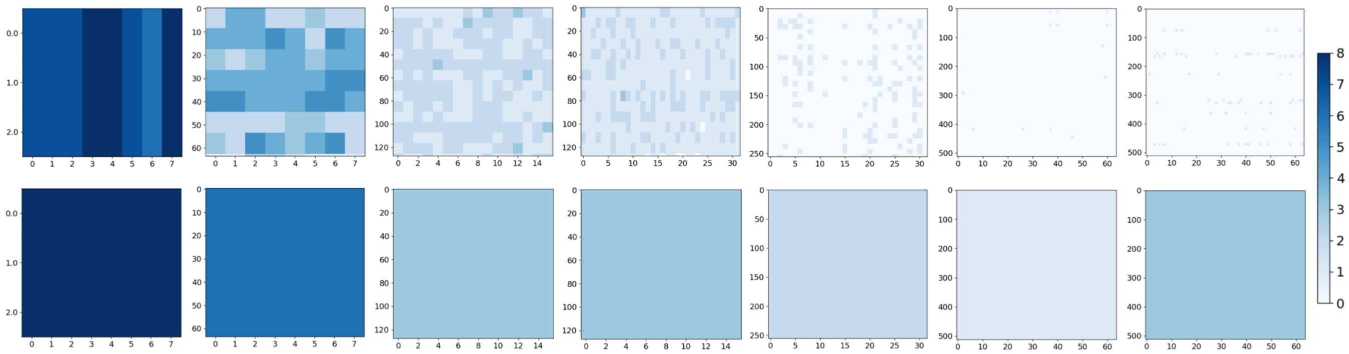


Fig. 7. Quantization results of 7 representative layers of VGG19-BN trained on CIFAR-10. The first and the second lines present the quantization results of BWQ-A and BSQ, respectively. Each column contains two bit-width maps corresponding to the same weight layer. The X-axis and Y-axis represent the index of the WBs in the C_{out} dimension and the C_{in} (or $C_{in} \times k \times k$) dimension, respectively. A colorbar is presented on the right. The specific layers are: "features.0", "features.4", "features.13", "features.18", "features.26", "features.43", "features.64".

TABLE I
HARDWARE CONFIGURATION.

BWQ-H configuration (1.2 GHz, 168 tiles per chip, 12 banks per tile)		
Component	**Spec**	**Power**
Memristor array	number: 1 per bank; size: 128×128; bit-per-cell: 1; OU size: 9 × 8	0.89W
DAC	number: 9 per bank; resolution: 1-bit	0.36W
ADC	number: 8 per bank; resolution: 4-bit	23.22W
Buffer	bitwidth: 64-bit	0.59W
Memory controller	number: 1 per bank; tech node: 28nm	92.8mW
Other digital components	S&A: 4 per bank; IR: 2KB; OR: 256B	92.6mW
Chip total		25.25W

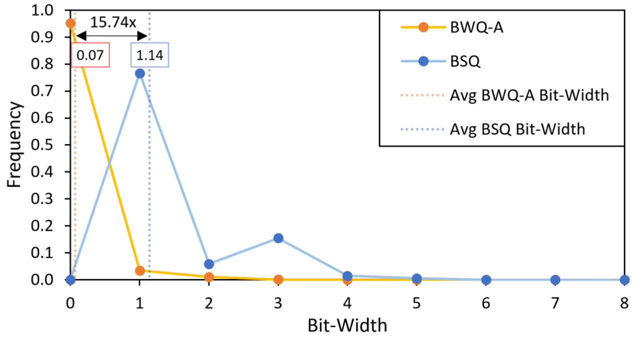


Fig. 8. Bit-width distribution of the VGG19-BN model trained on CIFAR-10 with BWQ-A and BSQ.

VI. Experimental Results

A. Algorithm

In this section, we compare the performance of BWQ-A with 32-bit floating-point baseline models and BSQ. The experimental results on CIFAR-10, CIFAR-100, and ImageNet are presented in Table II. Here, 'Act. Prec.' refers to activation precision, 'Acc' is the test accuracy, and 'Comp' is the model compression ratio. Typically, BWQ-A is able to achieve a higher weight and activation compression ratio with more redundant models. On the CIFAR-10 dataset, BWQ-A achieves an average (geometric mean) weight compression ratio of $81.98\times$ and an average activation compression ratio of $10.17\times$, all while maintaining an accuracy degradation within 1% compared to the floating-point baseline models. Image classification tasks on CIFAR-100 and ImageNet are more complicated. However, on average, BWQ-A is still able to attain a $41.42 \times /13.05\times$ weight compression ratio and an $8.81 \times /8.00\times$ activation compression ratio, respectively.

Compared with BSQ, BWQ-A is able to achieve a higher model compression ratio with similar accuracy, as BWQ-A leverages a finer mixed-precision quantization granularity. Fig. 7 shows the quantization results of 7 layers of VGG19-BN trained on CIFAR-10 with BSQ and BWQ-A. The first and the second rows of heatmaps present the quantization results of BWQ-A and BSQ, respectively. Each column contains two bit-width maps corresponding to the same weight layer. According to Fig. 7, in BWQ-A, as the WBs contribute differently to the training objective, they can be assigned flexible precisions based on their individual significance. In contrast, BSQ restricts the weights in the same layer to have the same precision. In BSQ, although many of the WBs may have low significance, all of the weights are designated with the maximum bit-width of the WB for that specific layer. The effects of this issue become more pronounced in deeper layers that contain a larger number of parameters. This drawback prevents BSQ from achieving the optimal trade-off between compression ratio and accuracy, rendering BSQ more resource-intensive when deployed on ReRAM-based accelerators.

Fig. 8 presents the bit-width distribution of the entire VGG19-BN model trained on CIFAR-10. We observe an evident shift in the distribution of these two model compression schemes. Notably, BWQ-A achieves a substantial reduction in average bit-width compared to BSQ, with an average $15.74\times$ lower bit-width.

B. Accelerator

Fig. 9 shows the normalized speedup and energy efficiency of all accelerators considered in this work normalized with respect to ISAAC. The performance of the accelerators is simulated under the OU-based operation scheme to ensure high computational accuracy and manageable peripheral overhead. Our proposed architecture BWQ-H is able to achieve

TABLE II
PERFORMANCE COMPARISON OF THE COMPRESSED MODELS.

Dataset	Model	#Param (M)	FP32 Acc (%)	BSQ			BWQ-A		
				Act. Prec.	Acc (%)	Comp. (×)	Act. Prec.	Acc (%)	Comp. (×)
CIFAR-10	ResNet-18	11.17	95.38	4	95.38	26.05	3	94.59	56.46
	ResNet-34	21.28	95.61	4	94.66	83.86	4	94.73	117.52
	VGG16-BN	33.65	92.60	3	92.45	26.59	3	92.60	136.01
	VGG19-BN	38.96	92.94	3	92.84	28.15	3	91.96	443.01
	ResNet-20	0.27	92.61	3	92.16	13.76	3	92.07	16.04
	MobileNetV2	2.30	94.43	4	94.24	5.73	3	93.54	47.34
CIFAR-100	ResNet18	11.22	75.61	4	76.35	6.21	4	75.31	45.97
	ResNet34	21.33	76.41	4	76.37	33.40	4	76.16	63.93
	VGG16-BN	34.02	72.93	4	72.37	19.88	3	72.87	50.42
	VGG19-BN	39.33	72.94	4	72.05	23.09	3	72.04	78.56
	MobileNetV2	2.41	75.54	8	75.13	6.22	4	74.57	18.35
	DenseNet121	7.00	75.99	5	75.18	7.26	4	75.07	23.65
ImageNet	ResNet34	21.80	73.55	4	72.62	9.48	4	72.56	13.55
	DenseNet121	7.98	74.65	5	73.70	7.47	4	73.68	12.56

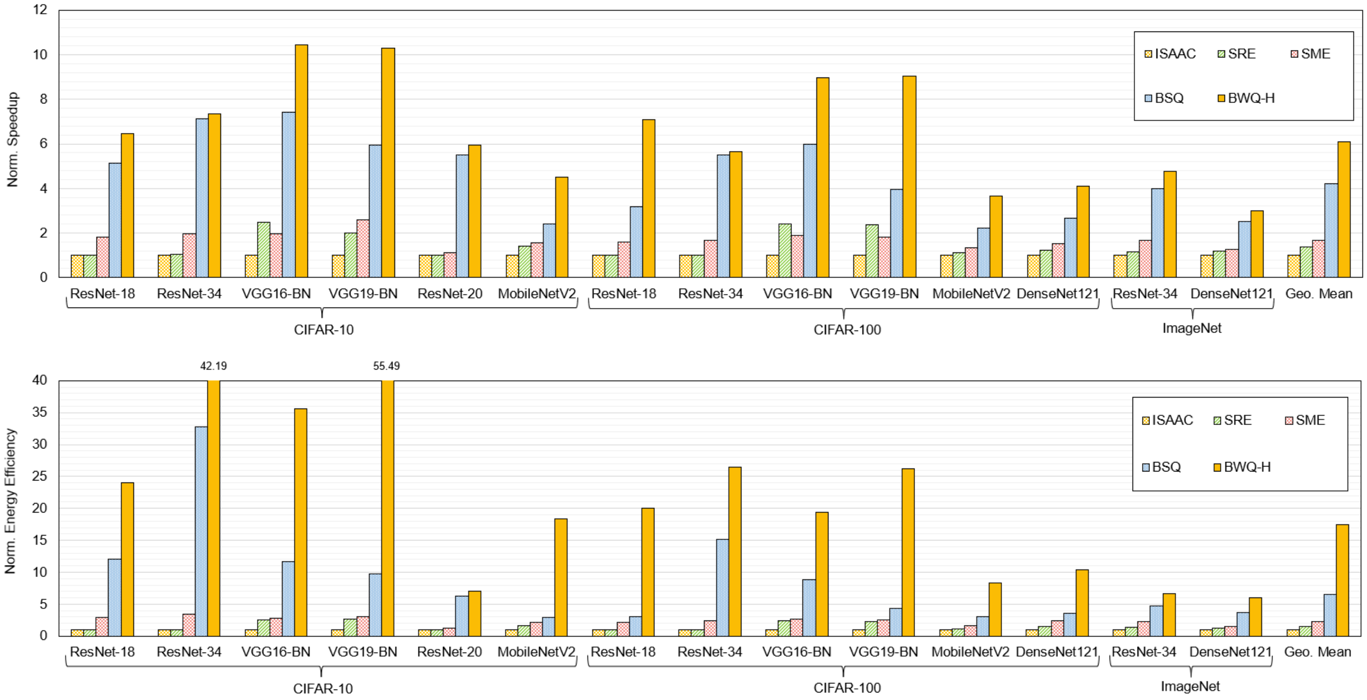


Fig. 9. The normalized speedup and energy efficiency (over ISAAC) of BWQ-H and other baseline accelerators.

the highest speedup and energy efficiency for all models and datasets considered in this work. On average, BWQ-H achieves 6.08× speedup and 17.47× energy saving over ISAAC. Fig. 10 illustrates the breakdown of BWQ-H's energy saving over ISAAC. The primary source of energy saving is attributed to the high weight compression ratio achieved through BWQ-A, while activation compression and the precision-aware mapping scheme also contribute to reducing BWQ-H's energy consumption. In the OU-based operation scheme, the increased number of cycles needed to complete the computation with the entire crossbar leads to a significant rise in ADC energy, which becomes the dominant part of energy consumption in the ReRAM-based accelerators. Therefore, applying models with a high weight compression ratio can effectively reduce the number of computation cycles, thus greatly enhancing energy efficiency.

A notable observation from Fig. 9 is that on both CIFAR-10 and CIFAR-100 datasets, despite achieving a significantly higher weight compression ratio with BWQ-A, BWQ-H's speedup for VGG19-BN is not higher than that of VGG16-BN. This is due to the fact that BWQ-H optimizes specific parts of the system, and there exists a speedup limit determined by the latency of the unoptimized components. BWQ-H is reaching the speedup limit on both models. However, being a deeper model, VGG19-BN introduces more layers with large input feature maps. As a result, the latency of relatively lower optimized parts, such as buffers and accumulation circuits, is increased.

Since the baselines represent different types of architectures, we discuss the comparison between BWQ-H's design with each baseline architecture separately.

ISAAC: To guarantee high computational accuracy, the 2-bit ReRAM cells assumed in ISAAC are replaced by 1-bit cells in our simulations. In ISAAC, both weights and activations are represented with 16-bit. While this configuration delivers nearly lossless accuracy, it's deemed excessive and resource-demanding for edge devices. The primary reason for BWQ-H's superior performance over ISAAC is its superior weight and activation compression rates. On average, BWQ-H achieves a speedup of $6.08\times$ and saves $17.47\times$ more energy than ISAAC.

SRE: SRE applies a HW-based optimization approach, exploring very fine-grained structured sparsity on the OU-row level from both the weight and the activation side. This method omits zero-valued OU rows, replacing them with subsequent non-zero values. However, this optimization approach only leverages the inherent sparsity of neural networks but does not proactively compress the models. Thus, its weight compression performance is much lower than what BWQ-A can achieve. For example, the highest average compression ratio that SRE can achieve is about $10\times$, and this is accomplished with the smallest 2×2 OUs. For 9×8 OUs, the compression ratio is only about $3.3\times$. On the other hand, to leverage fine-grained sparsity, SRE imposes substantial peripheral overheads for indexing control, including index storage and complex indexing logic. Therefore, BWQ-H is able to achieve $4.44\times$ average speedup and $11.98\times$ energy saving over SRE.

SME: SME proposes a HW/SW co-design approach to design a DNN acceleration framework for ReRAM devices. At the algorithm level, SME performs post-training-quantization to compress the model. In the proposed quantization scheme, at most 3 consecutive bits of the weights can be non-zero. To leverage the bit-level sparsity obtained from the compression algorithm, SME proposes a bit-wise inter-crossbar mapping scheme and a squeeze-out scheme. In the mapping scheme, the different bits of a weight matrix are mapped to different crossbars. In the squeeze-out process, if a crossbar row representing the LSB only contains zeros, then the zeros are squeezed out and the row is replaced with the non-zero values from the other crossbars with a right-shift. This requires doubling the corresponding activation values. However, SME's quantization scheme can significantly compromise model accuracy due to its constraints on the number of non-zero bits. Moreover, the mapping and squeeze-out methods only leverage the sparsity to a very limited extent. Only if an entire crossbar row consists of zero-weight values then the data in that row can be squeezed out. Thus, the de facto model compression ratio is low when large crossbars are utilized. Since SME requires doubling the activation data on specific rows, it also induces additional overhead to record the index of the activation data that is required to be doubled. Therefore, BWQ-H is able to achieve $3.66\times$ average speedup and $7.65\times$ energy saving over SME.

BSQ: As introduced in Section II-B, BSQ performs layer-wise mixed-precision quantization to compress both the weights and activations. Since all the weights within a layer have a uniform precision, BSQ doesn't require a LUT in the memory controller to track the bit-width for each WB as BWQ-H does. Under the OU-based operation scheme, BSQ simply performs sequential computation on each OU and fetches the activation data sequentially from the IR. The indexing overhead for BSQ under the OU-based operation scheme is negligible. Nonetheless, the speedup and energy performance of BSQ is lower than that of BWQ-H due to the fact that BWQ can achieve a higher model compression ratio with the fine-grained quantization granularity. On average, BWQ-H is able to achieve $1.45\times$ speedup and $2.66\times$ energy saving over BSQ.

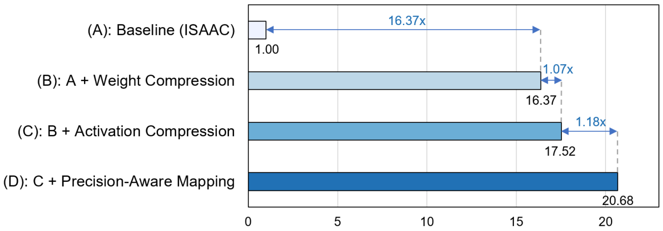


Fig. 10. Breakdown analysis for BWQ-H's energy saving over ISAAC.

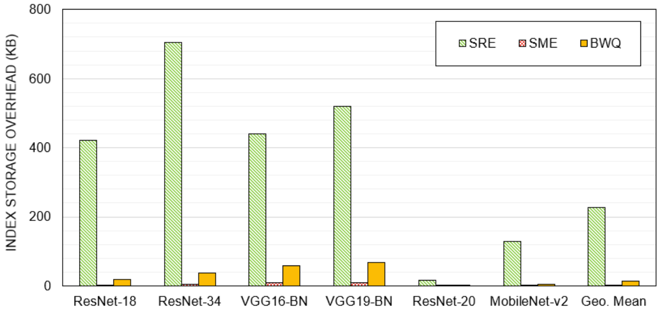


Fig. 11. Indexing overhead comparison of CIFAR-10 models.

C. Indexing Overhead Analysis

In this section, we evaluate the index overhead of BWQ-H, SRE, and SME when using the OU-based scheme for CIFAR-10 models which are mentioned in Table II. Here, 9×8 OUs are assumed. As shown in Fig. 11, on average, BWQ-H's indexing overhead is $17.38\times$ lower than SRE and $4.46\times$ higher than SME. As discussed in the previous section, as SRE exploits the extremely fine-grained OU-row level sparsity for both weight and activation, it suffers from significant index storage overhead. SRE can require as much as 704KB to store the index for the ResNet-34 model. On the other hand, the memory consumption for storing the index in SME is much smaller, as it only explores the sparsity at the crossbar-row level. However, this coarse-grained compression granularity hinders SME from achieving the optimal model compression ratio. As a result, BWQ-H can attain higher speedup and reduced energy consumption than SME.

TABLE III
WEIGHT SPARSITY ON THE CROSSBARS IN BWQ-H.

Dataset	Model	Weight Sparsity (%)
CIFAR-10	ResNet-18 VGG16-BN	21.51 11.67
CIFAR-100	ResNet-18 VGG16-BN	16.67 17.50
ImageNet	ResNet-34 DenseNet121	5.21 22.81

D. Analysis of the Weight Sparsity on the Crossbars

In this section, we evaluate the weight sparsity on the crossbars under the precision-aware mapping scheme and the impact of the spare ReRAM cells. Spare ReRAM cells exist when the total bit-width for each row of WBs varies. For instance, in Figure 6, the sum of the bit-width of the WBs in the first row is 2, whereas it is 3 for the second row. Therefore, there is a spare OU of four ReRAM cells at the end of the first OU row. Table III shows the weight sparsity of several representative models for various datasets in BWQ-H. The weight sparsity in the table is defined as the ratio of the total number of ReRAM cells that are not activated to the crossbar capacity. In general, the weight sparsity ranges from 5% to 22% for different models. A lower sparsity level indicates a more uniform distribution of precision among the WBs. However, it is worth noting that the existence of unused ReRAM cells does not impact the throughput of the crossbar. This is attributed to the fact that only one OU can be activated per cycle, and the OU's throughput is maximized under the precision-aware mapping scheme. Moreover, owing to the high density of the ReRAM crossbars in contrast to the peripheral circuits, the spare OUs would only result in negligible area overhead [5]. Unused ReRAM cells also do not result in an increased peripheral circuit overhead, as the capacity of the peripherals in each PIM bank is designed to handle the operations of a single OU.

E. Ablation Studies

1) Regularization Strength and Re-quantization Interval: These are the two most important hyperparameters in BWQ-A that affect the model's accuracy and compression ratio. As shown in Fig. 12, we evaluate the performance of BWQ-A with VGG19-BN on the CIFAR-10 dataset. Each data series represents 5 experiments, each with a different regularization strength: [5e-4, 1e-3, 3e-3, 5e-3, 1e-2]. It is understandable that applying larger regularization strength and more frequent re-quantization can enhance the model's compression ratio at the expense of accuracy. It can be concluded from Fig. 12 that the model's compression ratio is more influenced by the re-quantization interval than the regularization strength. It is evident from Fig. 12 that using shorter re-quantization intervals consistently yields a better trade-off between compression ratio and accuracy. However, for VGG-19BN trained on CIFAR-10, using a re-quantization interval shorter than 200 introduces excessive quantization noise, making the training process unstable.

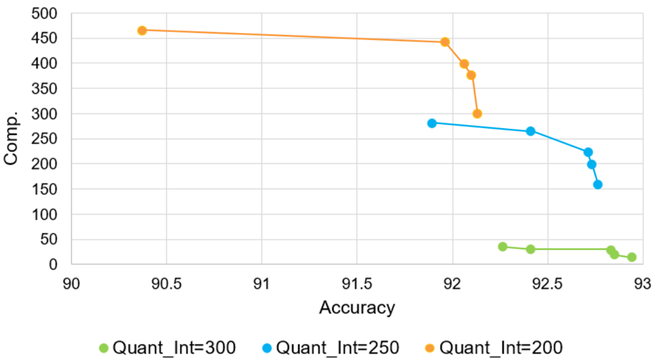


Fig. 12. BWQ-A's accuracy and model compression ratio against varying regularization strength and re-quantization intervals.

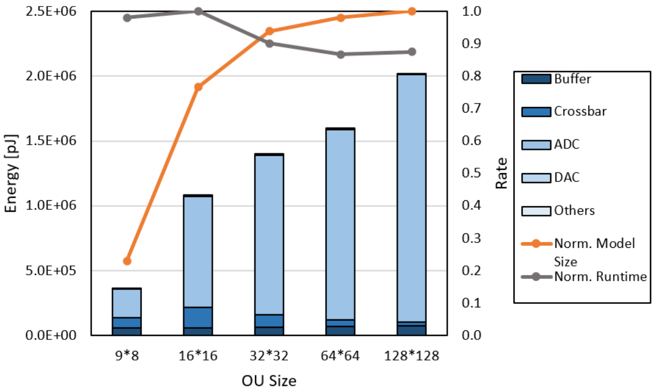


Fig. 13. BWQ's energy consumption, normalized model size, and normalized runtime against varying OU sizes. The bars correspond to the y-axis on the left, while the orange and black lines correspond to the y-axis on the right.

2) OU Size: In spite of the fact that the OU size for practical ReRAM-based accelerators is currently limited by the manufacturing technology of ReRAM devices, it is highly plausible that future advancements in manufacturing technology will enable the support of larger OU sizes. Therefore, we provide a future roadmap for practical ReRAM-based accelerators by examining the scalability of BWQ with varying OU sizes, from 9×8 to 128×128. Expanding the OU size results in a decrease in the number of WBs that can be mapped to the crossbar, thus reducing the size of the LUT in the memory controller which stores the bit-width of the WBs. However, this expansion also involves an increase in the number of required ADCs and DACs within each PIM Bank.

Fig. 13 shows BWQ's performance with the ResNet-18 model tested on CIFAR-10 for varying OU sizes. Other models exhibit similar trends. As coarser quantization granularity results in a lower model compression ratio, the model size achieved with the BWQ-A quantization scheme increases with OU size. The model's inference runtime can be affected by a number of factors when OU size varies. When a larger OU is considered, then the computation with the entire crossbar can be completed within fewer clock cycles, thus reducing the runtime. However, a larger OU leads to a lower model compression ratio. Thus, more clock cycles are required to finish the computation with a larger model. As a larger OU contains more BLs, it also requires higher ADC precision, which leads to increased ADC latency. Overall, the runtime only shows minor variations with increasing OU size, reaching its minimum value at an OU size of 64×64. A similar analysis can be made for energy consumption. Fig. 13 also shows the energy consumption breakdown of different components. ADC energy steadily increases with OU size since larger OUs require ADCs with higher precisions, and ADC energy scales up significantly with its precision. As ADC energy constitutes the majority of the energy consumption in our OU-based scheme, the overall energy consumption increases as the OU size grows. Therefore, if the primary goal is minimizing the runtime, a medium-sized OU would be preferable. On the other hand, if the task has a limited energy consumption budget, then the 9×8 OU configuration would be the most beneficial design option.

VII. Conclusion

In this paper, we present BWQ, an algorithm-architecture co-design framework to enable highly efficient ReRAM-based DNN accelerators. Due to the practical concerns, BWQ adopts the OU-based operation scheme. At the algorithm level, we introduce BWQ-A, a block-wise mixed precision quantization scheme. The size of the weight block aligns with the size of the OU. By employing finer quantization granularity, a high average model compression ratio of $58.27\times$ is achieved within 1% accuracy degradation compared to the floating-point baseline models. At the architecture level, we present the design of BWQ-H, which enables the efficient inference of the models quantized using BWQ-A. BWQ-H incorporates a novel precision-aware mapping scheme to increase memory utilization with an OU for the mixed-precision quantized models. Additionally, a memory controller is designed to generate control signals to the peripheral circuits to enable the computation of mixed-precision weights on the same crossbar. Leveraging the high model compression ratio, on average, BWQ-H achieves a $17.47\times$ speedup and a $6.08\times$ energy efficiency over existing ReRAM-based architectures. We also analyze the scalability of BWQ-A and BWQ-H with varying OU sizes. This serves as a future road map for practical ReRAM-based accelerators depending on the evolution of the manufacturing technology of ReRAM devices.

References

[1] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *2017 IEEE international symposium on high performance computer architecture (HPCA)*, pp. 541–552, IEEE, 2017.

[2] X. Yang, B. Yan, H. Li, and Y. Chen, "Retransformer: Reram-based processing-in-memory architecture for transformer acceleration," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

[3] T.-H. Yang, H.-Y. Cheng, C.-L. Yang, I.-C. Tseng, H.-W. Hu, H.-S. Chang, and H.-P. Li, "Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks," in *Proceedings of the 46th International Symposium on Computer Architecture*, pp. 236–249, 2019.

[4] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.

[5] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[6] F. Tu, Y. Wang, L. Liang, Y. Ding, L. Liu, S. Wei, S. Yin, and Y. Xie, "Sdp: Co-designing algorithm, dataflow, and architecture for in-sram sparse nn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 1, pp. 109–121, 2022.

[7] Q. Zheng, X. Li, Z. Wang, G. Sun, Y. Cai, R. Huang, Y. Chen, and H. Li, "Mobilatice: a depth-wise dcnn accelerator with hybrid digital/analog nonvolatile processing-in-memory block," in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

[8] C. Ogbogu, M. Soumen, B. K. Joardar, J. R. Doppa, D. Heo, K. Chakrabarty, and P. P. Pande, "Energy-efficient reram-based ml training via mixed pruning and reconfigurable adc," in *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, 2023.

[9] M.-Y. Lin, H.-Y. Cheng, W.-T. Lin, T.-H. Yang, I.-C. Tseng, C.-L. Yang, H.-W. Hu, H.-S. Chang, H.-P. Li, and M.-F. Chang, "Dl-rsim: A simulation framework to enable reliable reram-based accelerators for deep learning," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, ACM, 2018.

[10] S. Yang, W. Chen, X. Zhang, S. He, Y. Yin, and X.-H. Sun, "Auto-prune: Automated dnn pruning and mapping for reram-based accelerator," in *Proceedings of the ACM International Conference on Supercomputing*, pp. 304–315, 2021.

[11] W.-H. Chen, K.-X. Li, W.-Y. Lin, K.-H. Hsu, P.-Y. Li, C.-H. Yang, C.-X. Xue, E.-Y. Yang, Y.-K. Chen, Y.-S. Chang, *et al.*, "A 65nm 1mb nonvolatile computing-in-memory reram macro with sub-16ns multiply-and-accumulate for binary dnn ai edge processors," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 494–496, IEEE, 2018.

[12] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, *et al.*, "15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 244–246, IEEE, 2020.

[13] H. Ji, L. Song, L. Jiang, H. Li, and Y. Chen, "Recom: An efficient resistive accelerator for compressed deep neural networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 237–240, IEEE, 2018.

[14] S. Huang, A. Ankit, P. Silveira, R. Antunes, S. R. Chalamalasetti, I. El Hajj, D. E. Kim, G. Aguiar, P. Bruel, S. Serebryakov, *et al.*, "Mixed precision quantization for reram-based dnn inference accelerators," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pp. 372–377, 2021.

[15] Z. Zhu, H. Sun, Y. Lin, G. Dai, L. Xia, S. Han, Y. Wang, and H. Yang, "A configurable multi-precision cnn computing framework based on single bit rram," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[16] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq: Hessian aware quantization of neural networks with mixed-precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 293–302, 2019.

[17] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

[18] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of convnets via differentiable neural architecture search," *arXiv preprint arXiv:1812.00090*, 2018.

[19] H. Yang, L. Duan, Y. Chen, and H. Li, "Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization," *arXiv preprint arXiv:2102.10462*, 2021.

[20] S. Dash, Y. Luo, A. Lu, S. Yu, and S. Mukhopadhyay, "Robust processing-in-memory with multibit reram using hessian-driven mixed-precision computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 1006–1019, 2021.

[21] E. Hanson, S. Li, H. Li, and Y. Chen, "Cascading structured pruning: enabling high data reuse for sparse dnn accelerators," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pp. 522–535, 2022.

[22] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[23] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features

from tiny images," 2009.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[29] L. Xiao, H. Yang, Z. Dong, K. Keutzer, L. Du, and S. Zhang, "Csq: Growing mixed-precision quantization scheme with bi-level continuous sparsification," *arXiv preprint arXiv:2212.02770*, 2022.

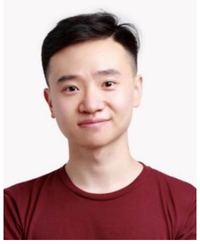
[30] T. maintainers and contributors, "Torchvision: Pytorch's computer vision library." https://github.com/pytorch/vision, 2016.

[31] F. Liu, W. Zhao, Z. He, Z. Wang, Y. Zhao, T. Yang, J. Feng, X. Liang, and L. Jiang, "Sme: Reram-based sparse-multiplication-engine to squeeze-out bit sparsity of neural network," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pp. 417–424, IEEE, 2021.

[32] Z. Zhu, H. Sun, K. Qiu, L. Xia, G. Krishnan, G. Dai, D. Niu, X. Chen, X. S. Hu, Y. Cao, *et al.*, "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, pp. 83–88, 2020.

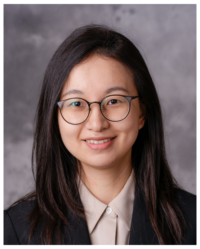
Xueying Wu received her B.Eng. degree in Microelectronics from Fudan University, Shanghai, China, in 2021. She is currently pursuing a Ph.D. degree with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA, supervised by Prof. Hai (Helen) Li. Her research interests include computer architecture, neural network compression, and hardware-software co-design of Processing-in-Memory systems.

Edward Hanson received his Ph.D. in Computer Engineering from Duke University in 2023 under the guidance of Prof. Yiran Chen. Previously, he received a B.S. degree in Computer Engineering from the University of Maryland, Baltimore County (UMBC) in 2019 and was awarded the Meyerhoff Premier Scholarship. His research interests lie in optimizing machine learning systems by co-designing algorithm, compile-time software, and architecture.

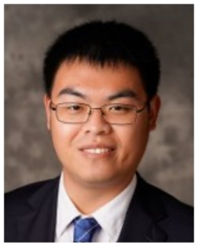


Nansu Wang received the bachelor of engineering and master of engineering degrees from China University of Petroleum, Beijing, China in 2015 and 2018 respectively. He also received the master of science degree in interdisciplinary data science from Duke University, Durham, NC, USA, in 2023. He is currently working as an engineer in data analysis at Lenovo. His research interest lies in neural network compression methodologies and applied machine learning.

Qilin Zheng received the B.S. degree from Peking University, Beijing, China, in 2019, and the M.S. degree from KU Leuven, Leuven, Belgium, in 2022. He is currently pursuing the Ph.D. degree in ECE with Duke University, Durham, NC, USA. His current research interests include computer architecture, non-volatile memory, and compute-in-memory design.



Xiaoxuan Yang (Member, IEEE) Xiaoxuan Yang is an Assistant Professor in the Electrical and Computer Engineering Department at the University of Virginia. She received her Ph.D. degree in Electrical and Computer Engineering at Duke University. Previously, she received the B.S. degree in Electrical Engineering from Tsinghua University and the M.S. degree in Electrical Engineering from the University of California, Los Angeles. Her research interests include emerging nonvolatile memory technologies, robustness and reliability enhancement in processing-in-memory designs, and hardware accelerators for deep learning applications. Her research work won Third Place of ACM Student Research Competition SRC at International Conference on Computer-Aided Design (ICCAD) and Best Research Award at ACM SIGDA Ph.D. Forum at Design Automation Conference (DAC). She is also selected as a Rising Star in EECS, an NSF iREDEFINE Fellow, and a Rising Scholars Postdoc Fellow by the School of Engineering & Applied Science, University of Virginia.



Huanrui Yang (Member, IEEE) is Postdoctoral Scholar in the EECS department of UC Berkeley, supervised by Prof. Kurt Keutzer. Before joining Berkeley he earned PhD from Duke University in 2022, supervised by Prof. Hai Li and Yiran Chen, and earned Bachelor's degree from Tsinghua University in 2017. His main research interest lies in compressing neural network models with methods like sparsity and quantization, and to evaluate and enhance the robustness of deep learning models. He published multiple papers at conferences such as NeurIPS, ICLR, CVPR, KDD, DAC, ICCAD, etc, and served as reviewer for multiple journals and conferences.

Shiyu Li (Graduate Student Member, IEEE) received the B.Eng. degree in automation from Tsinghua University, Beijing, China, in 2019. He is currently pursuing a Ph.D. degree with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA, supervised by Prof. Yiran Chen. His research interests include computer architecture, algorithm-hardware co-design of deep learning systems, and near-data processing.

Feng Cheng graduated with a bachelor's degree from the City University of Hong Kong, Hong Kong SAR, in 2022. He is now working towards a PhD in the Electrical and Computer Engineering Department at Duke University, Durham, advised by Dr. Y. Chen and Dr. H. Li. His research is focused on in-/near-memory computing and hardware-software co-design.

Partha Pratim Pande (Fellow, IEEE) is a professor and holder of the Boeing Centennial Chair in computer engineering at the school of Electrical Engineering and Computer Science, Washington State University, Pullman, USA. He is currently the director of the school. His current research interests are novel interconnect architectures for manycore chips, on-chip wireless communication networks, heterogeneous architectures, and ML for EDA. Dr. Pande currently serves as the Editor-in-Chief (EIC) of IEEE Design and Test (D&T). He is on the editorial boards of IEEE Transactions on VLSI (TVLSI) and ACM Journal of Emerging Technologies in Computing Systems (JETC) and IEEE Embedded Systems letters. He was/is the technical program committee chair of IEEE/ACM Network-on-Chip Symposium 2015 and CASES (2019-2020). He also serves on the program committees of many reputed international conferences. He has won the NSF CAREER award in 2009. He is the winner of the Anjan Bose outstanding researcher award from the college of engineering, Washington State University in 2013.

Janardhan Rao Doppa (Senior Member, IEEE) is the Huie-Rogers Endowed Chair Associate Professor at Washington State University. He received his PhD in computer science from Oregon State University. His research interests are at the intersection of machine learning and computing systems design. He won NSF CAREER award, Outstanding Paper Award from AAAI conference (2013), Best Paper Award from ACM Transactions on Design Automation of Electronic Systems (2021), IJCAI Early Career Award (2021), Best Paper Award from Embedded Systems Week Conference (2022, 2023), Best Paper Award from International Symposium on Low-Power Electronic Design (2023), Outstanding Junior Faculty in Research Award (2020) and Reid-Miller Teaching Excellence Award (2018) from the College of Engineering, Washington State University.

Krishnendu Chakrabarty (Fellow, IEEE) received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, respectively. He is now the Fulton Professor of Microelectronics in the School of Electrical, Computer and Energy Engineering at Arizona State University (ASU). He is also the Director of the ASU Center on Semiconductor Microelectronics and CTO of the South-West Advanced Prototyping (SWAP) Hub for the Department of Defense Microelectronics Commons. Before joining ASU, he was the John Cocke Distinguished Professor and Department Chair of Electrical and Computer Engineering (ECE), and Professor of Computer Science, at Duke University. His current research projects include: design-for-test of 2.5D/3D integrated circuits and heterogeneous integration; hardware security; AI accelerators; microfluidic biochips; AI for healthcare.

Prof. Chakrabarty is a recipient of the National Science Foundation CAREER award, the Office of Naval Research Young Investigator award, the Humboldt Research Award from the Alexander von Humboldt Foundation, Germany, the IEEE Transactions on CAD Donald O. Pederson Best Paper Award, the IEEE Transactions on VLSI Systems Prize Paper Award, the ACM Transactions on Design Automation of Electronic Systems Best Paper Award, multiple IBM Faculty Awards and HP Labs Open Innovation Research Awards, and over a dozen best paper awards at major conferences. He is also a recipient of the IEEE Computer Society Technical Achievement Award, the IEEE Circuits and Systems Society Charles A. Desoer Technical Achievement Award, the IEEE Circuits and Systems Society Vitold Belevitch Award, the Semiconductor Research Corporation Technical Excellence Award, the Semiconductor Research Corporation Aristotle Award, the IEEE-HKN Asad M. Madni Outstanding Technical Achievement and Excellence Award, and the IEEE Test Technology Technical Council Bob Madge Innovation Award. He is a Fellow of ACM, IEEE, and AAAS, and a Golden Core Member of the IEEE Computer Society.

Hai (Helen) Li (Fellow, IEEE) is Clare Boothe Luce Professor and Chair of the Electrical and Computer Engineering Department at Duke University. She received her B.S. and M.S. degrees from Tsinghua University, Beijing, China, and Ph.D. degree from the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, in 2004. Prior to joining Duke University, she worked with Qualcomm Inc., Intel Corporation, Seagate Technology, the Polytechnic Institute of New York University, and the University of Pittsburgh.

Prof. Li served as Associate Editor-in-Chief of IEEE Transactions on Circuits and Systems I (TCAS-I), Senior Editorial Board member of IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS), and Associate Editors of multiple IEEE and ACM journals. She has served as general chair and technical program chair of multiple IEEE conferences, including DAC, ISLPED, ISVLSI, SoCC, and ISQED, and the Technical Program Committee members of over 30 international conference series. She has been on the steering committee of ISVLSI and iNIS since 2016. Dr. Li serves on the IEEE Fellow committee.

Prof. Li's research interests include neuromorphic computing systems, machine learning and deep neural networks, memory design and architecture, and cross-layer optimization for low power and high performance. She has authored or co-authored more than 300 technical papers in peer-reviewed journals and conferences and a book entitled Nonvolatile Memory Design: Magnetic, Resistive, and Phase Changing (CRC Press, 2011). She received 9 best paper awards from international conferences. Prof. Li is a Distinguished Lecturer of the IEEE CAS society (2018-2019) and a distinguished speaker of ACM (2017-2020). Prof. Li is a recipient of the NSF Career Award, DARPA Young Faculty Award (YFA), TUM-IAS Hans Fischer Fellowship from Germany, and ELATE Fellowship (2022). She is a fellow of ACM and IEEE.