HAP: Efficient Quantization Harnessing Adaptive Precision for DNN Hardware Acceleration

Erjing Luo
University of Alberta
Edmonton, Canada
erjing1@ualberta.ca

Xinkuang Geng
Shanghai Jiao Tong University
Shanghai, China
xinkuang@sjtu.edu.cn

Honglan Jiang*
Shanghai Jiao Tong University
Shanghai, China
honglan@sjtu.edu.cn

Leibo Liu
Tsinghua University
Beijing, China
liulb@tsinghua.edu.cn

Jie Han*
University of Alberta
Edmonton, Canada
jhan8@ualberta.ca

Abstract

Post-training quantization (PTQ) avoids expensive training processes and is effective in accelerating deep neural networks (DNNs). Since the representation capacity of a quantizer decreases exponentially with precision, PTQ encounters fundamental limitations in the low-precision domain. As an alternative, adaptive-precision quantization (APQ) leverages the coding redundancy in binary representations to reduce precision without sacrificing resolution. This is especially suitable for activations, as they exhibit greater exploitable redundancy and lesser robustness than weights in PTQ. However, existing APQ approaches require data to heavily cluster near zero, which is not guaranteed in asymmetric quantization that is preferred for activations. Moreover, the precision variability in APQ leads to an unbalanced computational workload, making it difficult to effectively harvest the theoretical performance gain. To address these challenges, a novel quantization and accelerator design harnessing adaptive-precision (HAP), leverages a per-group dynamic zero-point to generalize APQ. Its intra-channel grouping strategy further enables balancing workloads through reordering. Leveraging a dual-precision vulnerable channel protection scheme for weights, it achieves superior accuracy compared with existing PTQ solutions at the level of 4 or 5 bits for a variety of DNN families. A novel bit-serial accelerator features a lightweight reorder engine, achieving a 2.65× speedup and a 55% energy reduction on average compared to existing designs.

1 Introduction

Quantization has become the *de-facto* technique to accelerate the inference of deep neural networks (DNNs). Recent applications show an inclination towards post-training quantization (PTQ), which does not involve expensive retraining processes or proprietary dataset problems [12, 16, 23]. However, the available coding space of a fixed-width quantizer decreases exponentially as bit-width reduces. In the sub-8-bit domain, this significantly limits the numerical resolution and may lead to catastrophic degradation in model accuracy [3, 5, 7, 17].

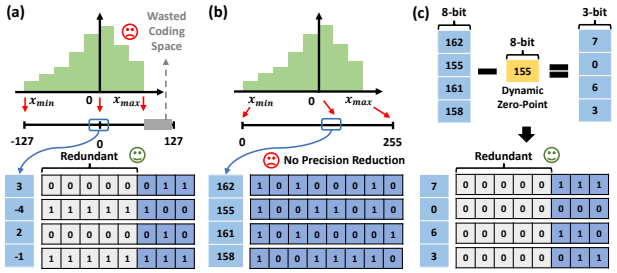

Figure 1: Applying APQ to a data group sampled from an asymmetric distribution under: (a) SQ; (b) AQ; (c) AQ + DAR.

An alternative to achieve aggressive compression without sacrificing resolution is adaptive-precision quantization (APQ) [2, 11, 14, 20]. APQ is motivated by the prevalent long-tailed distribution of DNN data, where small-magnitude data occur much more frequently [4, 6, 24]. As an example in Fig. 1 (a), even if high precision is selected for PTQ, the more significant bits in binary representations tend to repeat and fail to encode valid information. Exploiting this opportunity, APQ adjusts the encoding bit-widths of localized data to reduce bit-width on average. However, such reductions highly rely on data distribution characteristics and quantization settings. Moreover, a challenging aspect of APQ is its implementation. The variable precision not only requires dedicated arithmetic units but also, more importantly, can lead to imbalanced computational workloads in parallel architectures, making it challenging to achieve actual performance gains.

In this article, we still argue that APQ offers a promising PTQ solution for the activations in DNNs. On one hand, activations are typically disadvantaged in PTQ due to coarse quantization granularity. As a result, the low-precision PTQ approaches for them are less robust than those for weights [13, 15, 21]. APQ essentially avoids resolution loss for activations. As for weights, the PTQ methods tend to incur less accuracy drop, even at the 4-bit level [12, 16], or they can be substantially enhanced by algorithmic optimizations [5, 17, 19]. On the other hand, from an information theory perspective, activations exhibit higher exploitable coding redundancy. Fig. 2 plots the Shannon entropy of weights and activations by layer for

*Corresponding authors. The work at the University of Alberta was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326) and Alberta Innovates (Project Number: RES0053965).

DAC '26, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804255

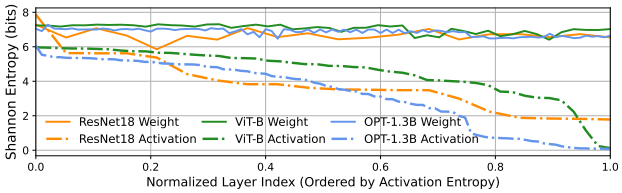

Figure 2: Layer-wise Shannon entropy of 8-bit DNNs.

three typical 8-bit models. As can be seen, the activation entropy is conspicuously lower, even falling to less than 4 bits in many layers.

Nonetheless, existing APQ methods have non-negligible limitations for activations. First, to efficiently reduce precision, symmetric quantization (SQ) is commonly assumed to ensure that the frequent near-zero values are projected as small-magnitude integers. However, as shown in Fig. 1 (b), this condition may not hold in asymmetric quantization (AQ), where most values are mapped to integers with large magnitudes. Compared to SQ, AQ can fit the input interval perfectly into the coding space with no waste. Therefore, it is preferred for activations [9], whose distributions exhibit pronounced asymmetry [4]. Second, existing APQ methods rarely address the workload imbalance problem, let alone activations, whose distributions can substantially change across input.

To bridge this gap, we present a novel PTQ and accelerator solution, aiming to **H**arness the **A**daptive-**P**recision (HAP) for efficient DNN inference. HAP adopts 8-bit AQ for activations and employs dynamic asymmetric re-quantization (DAR), a novel lossless APQ algorithm, to achieve low precision. DAR circumvents the limitation of existing APQ methods with a group-wise dynamic zero-point. Its intra-channel grouping strategy further exploits the value locality for lower average precision and enables balancing workload through reordering. In addition, HAP provides a vulnerable channel protection (VCP) scheme for weights, which targets 4-bit PTQ while allowing a fraction of channels to remain at 8-bit to preserve accuracy. VCP clusters channels with the same precision at compile-time and applies an orthogonal transformation to the subsequent weight tensor to counteract the induced effects. Experiments on a variety of DNNs including convolutional neural networks (CNNs), vision transformers (ViTs), and large language models (LLMs), show that HAP achieves superior accuracy compared with existing PTQ solutions at 4 or 5 bits. On the hardware side, we design a bit-serial accelerator that features a lightweight reorder engine to effectively improve workload balance. It achieves a 2.65× speedup and reduces energy by 55% on average compared to existing designs. The novel contributions are summarized below:

- A general and novel APQ algorithm, DAR, is proposed for activations, addressing the limitations of existing approaches.
- A dual-precision VCP scheme enhances weight quantization with compile-time optimizations for channel permutation.
- A dedicated bit-serial accelerator provides fine-grained precision adaptability and workload balancing capabilities to convert APQ into actual performance gains.

2 Preliminaries

2.1 Uniform Quantization Basics

AQ is the general form of uniform quantization. Essentially, it projects the input data x from a pre-calibrated interval $[x_{lb}, x_{ub}]$ into $[0, 2^B - 1]$ and rounds ($\lfloor \cdot \rceil$) it to a B-bit integer q, by

$$q = clip(\lfloor \frac{x}{\Delta} \rceil + Z; 0, 2^B - 1), \tag{1}$$

where $\Delta = \frac{x_{ub} - x_{lb}}{2^B - 1}$ is the scaling factor, the zero-point $Z = -\lfloor \frac{x_{lb}}{\Delta} \rceil$ is obtained by solving $q = 0$ at $x = x_{lb}$, and $clip(q; \alpha, \beta)$ further limits q to the nearest boundary if it exceeds $[\alpha, \beta]$. When the input interval is well calibrated, errors are mainly incurred by rounding, with an absolute upper-bound of $\frac{\Delta}{2}$ [10, 18]. Unlike weights that are calibrated at the finer-grained channel-level, activation PTQ suffers from larger scaling factors, thus it is typically less robust.

A quantized dot-product of activations and weights (denoted with superscripts a and w) can be computed in integer domain,

$$\begin{aligned} y_{i,j} &\approx \sum_{k=0}^{K-1} \Delta^a (q^a_{i,k} - Z^a) \Delta^w (q^w_{k,j} - Z^w) \\ &= \Delta^a \Delta^w \sum_{k=0}^{K-1} (q^a_{i,k} q^w_{k,j} - q^a_{i,k} Z^w - Z^a q^w_{k,j} + Z^a Z^w). \end{aligned} \tag{2}$$

In practice, SQ is more widely adopted, which further constrains the zero-points as zeros to simplify computation. SQ achieves this by extending the input interval symmetrically (i.e. $[-x_{sb}, x_{sb}]$, where $x_{sb} = max\{|x_{ub}|, |x_{lb}|\}$), and projecting data to a signed integer,

$$q = clip(\lfloor \frac{x}{\Delta} \rceil; -2^{B-1} + 1, 2^{B-1} - 1). \tag{3}$$

However, if data distribution is not nearly symmetric, the extended interval not only amplifies rounding errors but also leaves non-negligible coding space unused. As a result, AQ is favorable for activations due to the prevalent asymmetry [4, 9].

2.2 Adaptive-Precision Quantization

There have been several works exploring APQ. ShapeShifter [11] and Drift [14] adopt group-wise bit-width encoding to amortize overhead and adjust activation precision according to the largest data magnitude. BitWave [20] and BBS [2] can be considered as variants of APQ targeting weights. Instead of letting the precisions vary across data groups, they adopt bit-pruning algorithms to unify the number of effective bits before inference.

A salient limitation of existing APQ methods lies in the assumption that the frequent near-zero data in DNNs are always mapped to low-magnitude integers. This assumption empirically holds in SQ, as it only introduces scaling without any shift ($Z = 0$). However, in AQ, these data can be mapped to large-magnitude integers that require higher encoding bit-widths. Furthermore, for activations, few effective approaches have been proposed to mitigate the workload imbalance arising from irregular-precision data flow, which can severely degrade the achievable performance.

3 HAP Quantization

3.1 Dynamic Asymmetric Re-Quantization

Essentially, AQ does not change the coding redundancy in activations. It reduces the effectiveness of APQ, as the presence of Z can cause the frequent values to be mapped as large-magnitude integers. For a quantized data group that falls in $[q_{glb}, q_{gub}]$, we address this by further subtracting the minimum $\tilde{Z} = q_{glb}$. As illustrated in Fig. 1(c), this operation ensures that each quantized data group is remapped as small-magnitude integers (i.e. $\tilde{q} \in [0, q_{gub} - q_{glb}]$) that require lower encoding bit-widths,

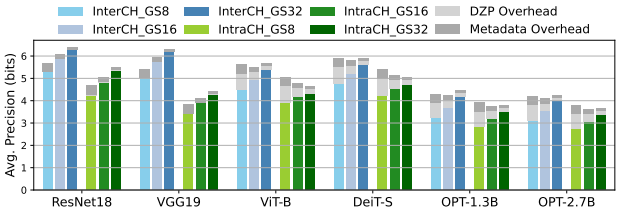

Figure 3: Average precision for DAR, under varied grouping strategies (intra- or inter-channel) and group sizes.

$$\tilde{q} = q - \tilde{Z} = clip(\lfloor \frac{x}{\Delta} \rceil + Z - \tilde{Z}; 0, 2^{\tilde{B}} - 1), \quad (4)$$

where $\tilde{B} = \lceil \log_2(q_{gub} - q_{glb} + 1) \rceil$. We name this method DAR, as an extra group-wise dynamic zero-point (DZP) $\tilde{Z}$ is introduced for precision adaptation. Notably, the B-bit DZP and $\lceil log_2 B \rceil$-bit metadata for indexing precision need to be encoded as well, but these overheads are amortized by the entire group.

For appropriately applying DAR in practice, we have two considerations. ① **DZP redundancy.** In some layers, due to the presence of ReLU and GELU functions, activations may natively cluster at the lower input boundary x_{lb}, making the overhead of DZP exceed its benefit. Hence, prior to inference, we decide whether to enable the DZP for a layer based on profiling. ② **Intra-channel value locality.** As indicated in [16, 22, 23], activations within a channel tend to have higher value locality. Unlike previous work [11, 14], we adopt an intra-channel grouping strategy to reduce the average precision. In Fig. 3, we profile the average adaptive precision achieved by DAR, along with its overhead. The results under different grouping strategies and group sizes, GS, consistently justify our considerations. We eventually choose $GS = 16$, which leads to an average precision of 4.39 bits.

3.2 GEMM Dataflow for DAR

The major layers in DNNs are commonly transformed as tiled general matrix multiplication (GEMM) tasks for processing. We focus on designing an efficient dataflow for a GEMM tile of activation and weights, $A \times W$, where $A \in \mathbb{R}^{M \times K}$ and $W \in \mathbb{R}^{K \times N}$. In particular, we assume the number of rows in A is equal to DAR group size (i.e. $M = GS$). Before being fetched on-chip, activations are pre-processed as four components: a) a variable-precision matrix where each column shares the same precision, (b) a DZP vector, c) a shared static zero-point, and d) a precision metadata vector, as shown in Fig. 4 (a). Assuming SQ for weights, DAR reformulates the dot-product equation using three terms,

$$\hat{y}_{i,j} \approx \Delta^a \Delta^w \sum_{k=0}^{K-1} (\tilde{q}^a_{i,k} - Z^a + \tilde{Z}^a_k) q^w_{k,j} = \Delta^a \Delta^w (P_A + P_S + P_D), \quad (5)$$

where

$$P_A = \sum_{k=0}^{K-1} \tilde{q}^a_{i,k} q^w_{k,j}, P_S = -Z^a \sum_{k=0}^{K-1} q^w_{k,j}, P_D = \sum_{k=0}^{K-1} \tilde{Z}^a_k q^w_{k,j}.$$

We focus on the variable-precision term P_A and dynamic zero-point term P_D, as P_S only involves static data and can be pre-computed.

To efficiently process P_A, we propose the outer-product reordering dataflow (ORD), as shown in Fig. 4 (b). ORD is essentially based on a bit-serial output-stationary (OS) dataflow that is widely adopted in bit-serial architectures [2, 20]. Specifically, both dataflows target broadcasting the GEMM tile onto an $M \times N$ processing array, where each processing engine (PE) contains L bit-serial multipliers and serves to accumulate the results. This is achieved by folding both matrices along the dimension K into L parallel submatrices and processing them as K/L iterations in a bit-serial fashion. Although this dataflow provides fine-grained precision adaptability, the time interval between iterations must be determined by the highest precision for synchronization. For DAR, such a workload imbalance occurs when the precisions of the L activation groups in an iteration do not exactly match. To mitigate this, ORD seeks to permute the order of activation groups to match precision. The permutation will remain error-free if the weight matrix is reordered accordingly, which is mathematically equivalent to reordering the outer-products in GEMM, as permitted by the commutative property. The implementation of ORD will be detailed in Section 4.

The DZP term P_D is a vector-matrix multiplication. For reusing hardware, the DZP vector is decomposed bit-wise and expanded as a bit matrix, such that each bit significance can be mapped to a row of the array. In particular, we assume $M = GS$ is a multiple of B^a and map GS/B^a DZP vectors each time. After computing, the partial results from every B^a PEs in a column need to be added. As the example shows in Fig. 4 (c), we provide additional aggregation paths between PEs to perform summation in $log_2(B^a)$ cycles.

At execution time, the P_A and P_D terms of a tile are scheduled alternately, such that they can be summed up on-chip. Supposing the average precision for P_A is $\bar{B}^a$ and the achieved average PE utilization is $\gamma \in [0, 1]$. The computing cycles of a tile can be derived as $(\frac{\bar{B}^a}{\gamma} + \frac{B^a}{GS})\frac{K}{L} + log_2(B^a)\frac{B^a}{GS}$, with the first term dominating.

3.3 Vulnerable Channel Protection

Although literature has demonstrated the feasibility of 4-bit PTQ for weights, adopting such aggressive precision inevitably incurs accuracy drop. Given that the output-channels of weights are quantized separately, HAP provides an alternative 8-bit precision to protect vulnerable channels (VCs) suffering from worse fidelity degradation. To keep weight precision consistent across the PEs, similar to [2], we limit the number of VCs to a multiple of the array column count N and cluster them at compile-time. As shown in Fig. 4 (d), this permutation changes the input-channel order of the activation matrix in the next layer. To ensure correctness, we apply an orthogonal permutation to the next weight matrix to counteract this effect. This compile-time optimization is applicable for all of the single-path layers in DNNs. If the model contains self-attention modules, we additionally impose constraints to the query, key, and value layers to ensure: a) they have identical permutations; and b) all channels in each attention head remain clustered. If two-path residual structures exist, we assume a direct memory access (DMA) to explicitly permute the secondary path, whose latency can be hidden by the computation of the major path.

As there is no absolute metric to compare vulnerability in PTQ, we propose a Pareto-based multi-metric evaluation approach to empirically identify VCs. The insight is to comprehensively evaluate all output-channels with multiple PTQ metrics, each capturing distinct aspects of sensitivity. Specifically, we evaluate them using MSE, as well as the KL-divergence between the 8- and 4-bit quantized distributions. While MSE captures the average rounding errors, KL-divergence can provide a unified measurement of information

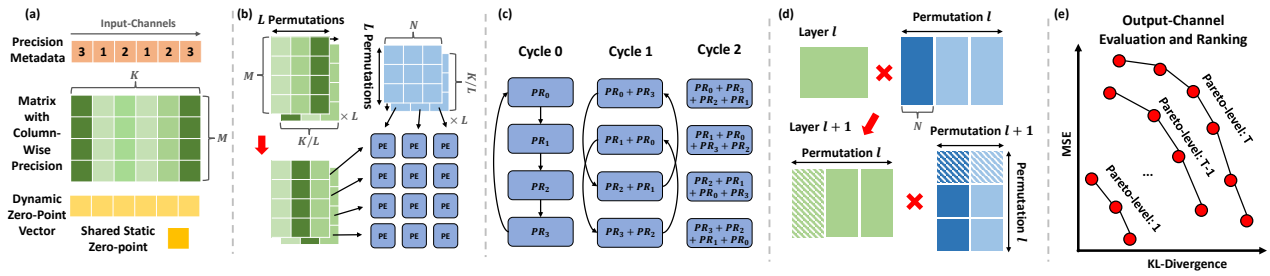

Figure 4: Quantization illustrations with $M = GS = 4$, $N = 3$, $K = 6$, $L = 2$, and $B^a = 4$. (a) Intra-channel DAR; (b) ORD; (c) partial result aggregation; (d) VC clustering and orthogonal permutation; (e) Pareto-based weight channel evaluation and ranking.

loss. As detailed in Algo. 1, it starts by uniformly adopting 4-bit quantization. Then, it evaluates all channels and globally performs non-dominated sorting (*Line 2-3*) to derive Pareto-levels as shown in Fig. 4(e). Based on them, it sorts the channels of each layer in descending order and partitions them into chunks of size N (*Line 5-8*). Finally, it sorts all chunks based on average Pareto-levels, and greedily promotes vulnerable chunks to 8-bit quantization under a pre-defined constraint for the total bit operations (*Line 9-14*).

Algorithm 1: Channel-protected Weight Quantization

Input: model $\mathcal{M}$, target bit operations TarBitOps, array column N
1 $\mathcal{M} \leftarrow$ Quant($\mathcal{M}$, 4b), TotalBitOps $\leftarrow$ BitOps($\mathcal{M}$, 4b)
2 MSE, KL $\leftarrow$ Evaluation($\mathcal{M}$.Channels)
3 ParetoLevel $\leftarrow$ NonDominatedSort(MSE, KL)
4 ChunkList, ScoreList $\leftarrow \emptyset$
5 **for** Layer in $\mathcal{M}$ **do**
6 ChannelList, LevelList $\leftarrow$ DescentSort(Layer.Channels, ParetoLevel)
7 Chunk, ChunkScore $\leftarrow$ Partition(ChannelList, LevelList, N)
8 ChunkList.append(Chunk), ScoreList.append(ChunkScore)
9 ChunkList $\leftarrow$ DescentSort(ChunkList, ScoreList)
10 **while** ChunkList $\neq \emptyset$ **do**
11 Chunk $\leftarrow$ ChunkList.pop(0)
12 $\Delta_{ops} \leftarrow$ BitOps(Chunk, 8b - 4b)
13 **if** TotalBitOps + $\Delta_{ops} \leq$ TarBitOps **then**
14 Chunk $\leftarrow$ QuantPromote(Chunk, 8b), TotalBitOps += Δ_{ops}
15 **Return** quantized model $\mathcal{M}$

4 HAP Architecture

4.1 Overall Architecture

Fig. 5 illustrates the overall architecture of the HAP accelerator, which includes a processing array, a reorder engine (RE), SRAM buffers, and a controller. The RE is the key module of HAP, which schedules the permutations in ORD. As described in Section 3.2, it seeks to reorder the activation groups within each submatrix, such that the L precisions in each iteration are matched as closely as possible. To achieve this, each submatrix is equipped with a cache-like register page consisting of R entries, where an entry contains a valid tag, a precision tag, and the addresses of the corresponding weight rows and activation columns. Once the dispatcher detects a match (or an exception), it returns the permuted addresses and precisions. The array then retrieves the data from SRAMs for execution. The register pages, dispatcher, and array work in a pipelined fashion, ensuring continuous computing iterations.

4.2 Processing Engine & Array

There are L bit-serial multipliers in each PE. An adder tree sums up the results and passes the summation to a common left-shifter. To support the dual-precision VCP, the primitive arithmetic precision

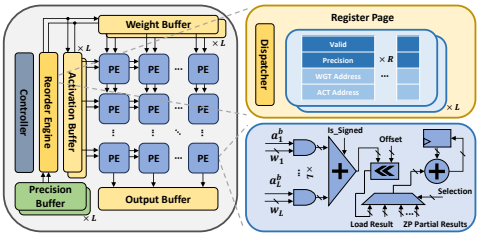

Figure 5: HAP architecture overview.

is set to 4-bit. In the 8-bit mode, an operand is sliced into a signed more significant part and an unsigned less significant part to be processed in two continuous cycles. As there is no fundamental difference between unsigned and signed additions, this is achieved with configurable sign-extension logic. The 32-bit accumulator is reused to accumulate the local result and aggregate the partial results from other PEs, which are selected by a multiplexer. For hyper-parameters, we empirically set $L = 16$ to match the throughput of a PE to an 8-bit multiplier. The dimension of the processing array is configured as 16×32, equivalent to 512 8-bit multipliers.

4.3 Matching Mechanism Analysis

As a cache-in operation is usually faster than a dispatch due to bit-serial processing, a larger register page can collect more data, thus it ensures a higher chance of detecting at least a match. However, given that registers and associative matching logic are area-consuming, the register page size is limited to $R = 8$.

The match rate can be analyzed using a probability model. We use E_b^l to denote that there is at least a b-bit group on the l^{th} page. A precision is regarded as matched (E_b) if it exists on all pages. As each page is independent, this probability can be calculated as

$$P(E_b) = P(E_b^1 \cap \cdots \cap E_b^L) = \prod_{l=1}^{L} P(E_b^l). \tag{6}$$

Finally, the match rate is the probability that at least one of the precisions is matched. An approximate equation can be used for analysis, where we assume the independence between $\overline{E}_b$,

$$\begin{aligned} P(E_1 \cup \cdots \cup E_B) =& 1 - P(\overline{E}_1 \cap \cdots \cap \overline{E}_B) \\ \approx& 1 - \prod_{b=1}^{B} [1 - P(E_b)]. \end{aligned} \tag{7}$$

Considering a uniform distribution for the precisions, we can derive $P(E_b^l) = 1 - (1 - \frac{1}{B})^R$. However, this leads to an unacceptable match rate of 0.946% for our case ($B = 8$, $L = 16$, $R = 8$).

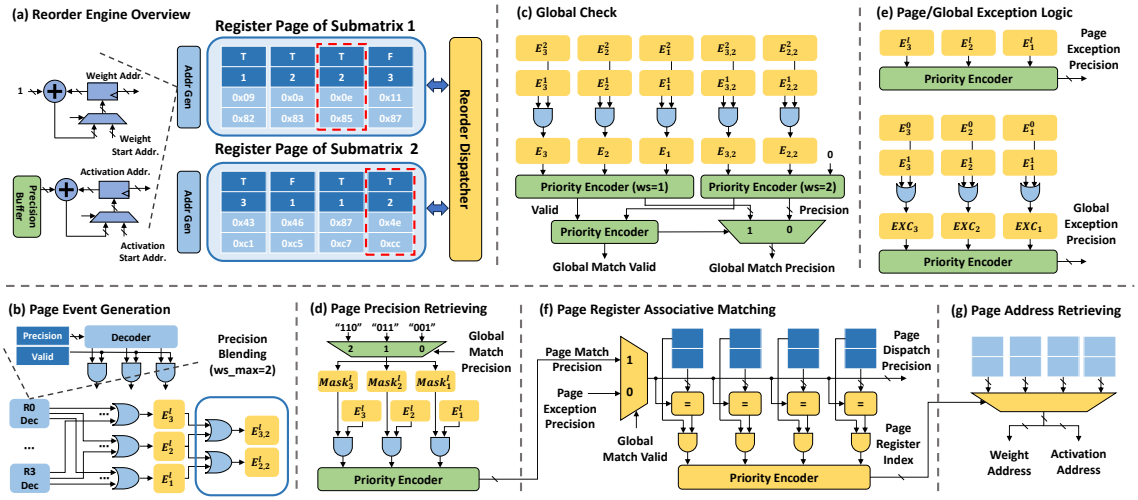

Figure 6: Reorder engine datapath example ($B = 3$, $L = 2$, $R = 4$, $ws_{max} = 2$). (a) Overview; (b) event generation; (c) event matching; (d) dispatch precision retrieving; (e) exception logic; (f) page register associative matching; (g) page address retrieving.

We further find that relaxation on match conditions can effectively solve this problem. We name this relaxed mechanism as precision blending (PB). Given a blending window size ws, if any group precision b' satisfies $0 \le b - b' < ws$, we consider that it can be matched on b as well, which leads to $ws - 1$ idling cycles at most. Essentially, PB generalizes the definition of the page match event as $E^l_{b,ws} = E^l_b \cup \cdots \cup E^l_{b-(ws-1)}$, where $ws = 1$ is the exact mechanism. Under the same assumptions, PB modifies this probability to $P(E^l_{b,ws}) = 1 - (1 - \frac{ws}{B})^R$. With $ws = 2$, the match rate is drastically increased to $P(E_{2,2} \cup \cdots \cup E_{B,2}) \approx 76.10\%$.

4.4 Reorder Engine

Fig. 6 shows an example for the RE datapath design based on PB. It is noteworthy that the hyper-parameters can be easily generalized.

4.4.1 Precision Loading & Address Generation. The datapath starts by fetching the precision metadata from the precision buffer and generating both addresses. We assume that both weights and activations are stored continuously. After the start addresses are set, the next addresses are computed by incrementally adding an offset. While it is a constant for weights, the activation addresses are offset by the loaded precisions. Once there are empty entries, new addresses and precisions are cached.

4.4.2 Match Path. The dispatcher implements logic gates to generate and detect matched events. In each page, a decoder-OR network decodes the precision registers and performs PB (Fig. 6 (b)). To ensure a high match rate while mitigating the inaccurate matching incurred by relaxation, we concurrently evaluate matched events under multiple PB window sizes (i.e. $ws \in \{1, \ldots, ws_{max}\}$). These events are summarized in a global check module (Fig. 6 (c)), where a two-level priority encoding strategy is employed to select the best dispatch precision. The priority is determined based on the insight that the matched events with smaller window size but higher precision generally lead to a more balanced workload. Based on this, in the first stage, the ws_{max} priority encoders separately encode their highest precisions. The second stage then prioritizes the one with the smallest window size, and uses a multiplexer to select the global dispatch precision. It is noteworthy that the global dispatch precision may not exist exactly in each page due to PB. Therefore, for each page, we use a B-bit mask to AND possible signals, and encode the highest satisfied precision (Fig. 6 (d)).

4.4.3 Exception Path. There are two types of exceptions to be handled: a) the register pages are full, but no match event occurs; b) there are no incoming data, but no match event occurs. Under such cases, we empirically enforce each page to dispatch its highest precision, which are detected using priority encoders (Fig. 6 (e)).

4.4.4 Register Match and Address Retrieving. The matching logic and the exception logic run concurrently in the dispatcher datapath. A multiplexer controlled by a global match valid signal selects the match or the exception precision. Next, associative matching is performed to retrieve the register index corresponding to this dispatch precision. Weight and activation buffer addresses are then accessed. Finally, the valid flags of these registers are deasserted.

5 Experiments

5.1 Experimental Setup

To demonstrate the generality and scalability of HAP, we evaluate it on a variety of DNN architectures/tasks, including CNNs (ResNet18, VGG19) and ViTs (ViT-B, Deit-S) on ImageNet-1K, and LLMs (OPT-1.3B, OPT-2.7B) on WikiText-2. To study accuracy, the algorithm is implemented and verified in Pytorch, and it is compared with recent PTQ solutions: SQuant [5], NoisyQuant [15], SmoothQuant [21], and OmniQuant [19]. For QmniQuant, we only consider the learnable weight clipping (LWC), as we find the learnable equivalent transformation (LET) does not always converge on our models. By default, we assume per-layer AQ for activations and per-channel SQ for weights. All activations are calibrated based on 128 random samples following [12].

For hardware evaluation, we compare HAP with three bit-serial accelerators. While we run baseline 8-bit models on Stripes [8], ShapeShifter [11] and BitWave [20] leverage precision adaptability in activations and weights, respectively. For a fair comparison, all accelerators are scaled to match the computing capability of 512 8-bit multipliers and are configured with an equal SRAM size of 640 KB in total. We implement HAP and model other accelerators

in SystemVerilog, and synthesize them with Synopsys Design Compiler under 65-nm TSMC process at the frequency of 500 MHz. For end-to-end performance and energy evaluation, we develop a cycle-accurate simulator to precisely model the behaviors of all accelerators, in which we assume a 128 bits/cycle DRAM bandwidth and use CACTI [1] to estimate the energy cost of memories.

5.2 Accuracy Results

We empirically adopt two configurations for VCP, equivalent to average weight precisions of 4.6 and 5.6 bits. Table 1 reports the accuracy (or perplexity) results. Comparing W4A8 and W8A4, it can be verified that activations are typically more sensitive to quantization precision. HAP leverages APQ to compress activations, thus fundamentally preserving most of the accuracy. In addition, VCP effectively enhances pure 4-bit quantization, providing the flexibility to trade off accuracy and performance. Since it just changes the channel precision, it is completely compatible with weight-specific PTQ methods to achieve better results. As can be seen, even in the aggressive setting (W4.6/A4~), HAP shows better results than existing PTQ solutions at W5A5, with a non-trivial margin.

Table 1: Top-1 Accuracy (%) & Perplexity (for LLMs)

Prec. (W/A)	32/32	8/8	4/8	8/4	5/5	4.6/4~	5.6/4~
CNNs	FP	SQuant [5]				HAP+SQuant	
ResNet18 ↑	69.76	69.65	68.49	63.41	68.09	68.84	**69.21**
VGG19 ↑	74.23	74.16	73.61	69.22	72.95	73.91	**74.09**
ViTs	FP	NoisyQuant [15]				HAP	
ViT-B ↑	84.54	84.31	81.62	72.42	79.04	82.98	**83.47**
Deit-S ↑	79.83	79.24	76.50	48.32	61.05	78.26	**78.84**
LLMs	FP	SmoothQuant [21]				HAP	
OPT-1.3B ↓	14.62	16.77	29.15	675.80	34.07	23.77	19.13
OPT-2.7B ↓	12.47	13.90	34.17	380.78	35.05	22.66	16.13
LLMs	FP	OmniQuant-LWC [19]				HAP+LWC	
OPT-1.3B ↓	14.62	15.36	17.15	321.43	92.70	16.01	**15.81**
OPT-2.7B ↓	12.47	13.44	14.42	1573.49	50.90	14.29	**13.96**

5.3 Load Balance Analysis

We take ViT-B as an example to study workload balance. Fig. 7 reports the match rate and PE utilization under varied ws_{max} and R. Without reordering, the naive utilization is only about 66.8%, leading to 49.7% more processing cycles. Exact RE indeed improves PE utilization, but the match rate grows moderately as R increases. With relaxation, PB drastically boosts the match rate. Although it degrades the utilization for the matched cases, the overall utilization is enhanced under the same R. We finally select $ws_{max} = 3$ and $R = 8$, as utilization almost saturates at 91.0%.

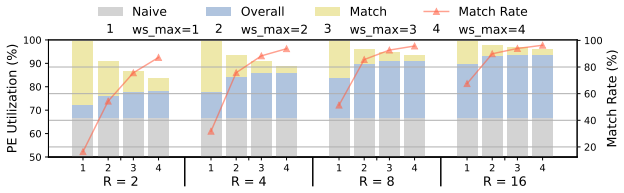

Figure 7: Match rate and PE utilization based on ViT-B.

5.4 Accelerator Performance & Energy

Performance. HAP reveals superior performance as it compresses both activations and weights to lower precisions. Fig. 8 reports the detailed speedup normalized to Stripes. Ablating from weight VCP, HAP_W8.0 achieves a 1.63× speedup due to DAR and RE. In particular, the two LLMs exhibit even higher speedups (2.06× and 2.09×) due to lower activation precisions. In contrast, ShapeShifter leads to a moderate result of 1.10× because of workload imbalance and limitations on AQ. Reducing the precision of non-vulnerable weight channels brings incremental gains. The achieved speedups of 2.25× (HAP_W5.6) and 2.65× (HAP_W4.6) showcase superior performance compared to their equivalent BitWave counterparts.

Energy. Fig. 9 shows the normalized energy breakdown, including the static energy and the dynamic energy for DRAM, SRAM, and the compute core. HAP increases SRAM accesses for weights, as P_A and P_D are separated. However, notable energy savings are achieved, attributed to reduced computations and DRAM accesses.

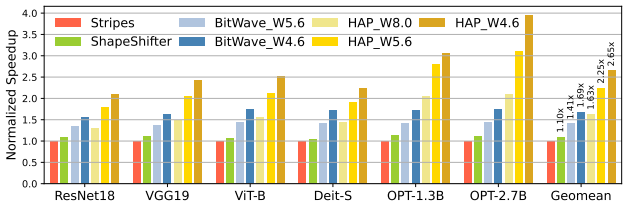

Figure 8: Normalized speedup.

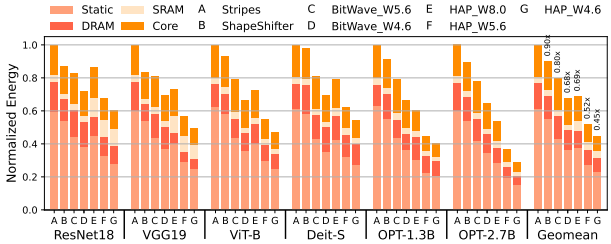

Figure 9: Normalized energy breakdown.

5.5 Core Area & Power Breakdown

For simplicity, we adopt the flip-flop-based register pages, where each entry includes a 4-bit tag and a 24-bit data. Table 2 reports the area & power breakdown of the compute core. Overall, the RE only takes 4.83% of area and 6.05% of power because its datapath mainly consists of simple logic gates, decoders, multiplexers, and priority encoders, which are well optimized by synthesis tools. The register pages occupy most of the area and power of RE, which can be implemented with SRAMs for further optimizations.

Table 2: Core Area & Power Breakdown

Module	Area (mm^2)		Power (mW)	
PE Array	1.0034	93.72%	254.7820	92.15%
RE Datapath Base	0.0080	0.75%	0.3180	0.12%
RE Precision Blending	0.0009	0.08%	0.1710	0.06%
Register Pages	0.0428	4.00%	16.2397	5.87%
Others	0.0156	1.45%	4.9880	1.80%
Total	1.0707	100.00%	276.4987	100.00%

6 Conclusion

We present HAP, a quantization and accelerator design for low-precision PTQ. HAP harnesses the inherent precision adaptability in activations to achieve low-precision while leveraging dual-precision VCP to enhance aggressive weight quantization. A novel bit-serial accelerator leverages a reorder engine to address the workload imbalance problem incurred by variable-precision computing. Our experiments reveal that HAP achieves superior accuracy, speedup, and energy reduction compared with related works.

References

[1] Rajeev Balasubramonian, Andrew B. Kahng, Naveen Muralimanohar, Ali Shafiee, and Vaishnav Srinivas. 2017. CACTI 7: New Tools for Interconnect Exploration in Innovative Off-Chip Memories. *ACM Trans. Archit. Code Optim.* 14, 2, Article 14 (June 2017), 25 pages. doi:10.1145/3085572

[2] Yuzong Chen, Jian Meng, Jae-sun Seo, and Mohamed S. Abdelfattah. 2024. BBS: Bi-Directional Bit-Level Sparsity for Deep Learning Acceleration. In *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 551–564. doi:10.1109/MICRO61859.2024.00048

[3] Xinkuang Geng, Siting Liu, Jianfei Jiang, Kai Jiang, and Honglan Jiang. 2024. Compact Powers-of-Two: An Efficient Non-Uniform Quantization for Deep Neural Networks. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–6. doi:10.23919/DATE58400.2024.10546652

[4] Xinkuang Geng, Siting Liu, Leibo Liu, Jie Han, and Honglan Jiang. 2024. QUQ: Quadruplet Uniform Quantization for Efficient Vision Transformer Inference. In *Proceedings of the 61st ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) *(DAC '24)*. Association for Computing Machinery, New York, NY, USA, Article 272, 6 pages. doi:10.1145/3649329.3656516

[5] Cong Guo, Yuxian Qiu, Jingwen Leng, Xiaotian Gao, Chen Zhang, Yunxin Liu, Fan Yang, Yuhao Zhu, and Minyi Guo. 2022. SQuant: On-the-Fly Data-Free Quantization via Diagonal Hessian Approximation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=JXhROKNZzOc

[6] Cong Guo, Jiaming Tang, Weiming Hu, Jingwen Leng, Chen Zhang, Fan Yang, Yunxin Liu, Minyi Guo, and Yuhao Zhu. 2023. OliVe: Accelerating Large Language Models via Hardware-friendly Outlier-Victim Pair Quantization. In *Proceedings of the 50th Annual International Symposium on Computer Architecture* (Orlando, FL, USA) *(ISCA '23)*. Association for Computing Machinery, New York, NY, USA, Article 3, 15 pages. doi:10.1145/3579371.3589038

[7] Cong Guo, Chen Zhang, Jingwen Leng, Zihan Liu, Fan Yang, Yunxin Liu, Minyi Guo, and Yuhao Zhu. 2022. ANT: Exploiting Adaptive Numerical Data Type for Low-bit Deep Neural Network Quantization. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1414–1433. doi:10.1109/MICRO56248.2022.00095

[8] Patrick Judd, Jorge Albericio, Tayler Hetherington, Tor M. Aamodt, and Andreas Moshovos. 2016. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–12. doi:10.1109/MICRO.2016.7783722

[9] Dongyun Kam, Myeongji Yun, Sunwoo Yoo, Seungwoo Hong, Zhengya Zhang, and Youngjoo Lee. 2025. Panacea: Novel DNN Accelerator using Accuracy-Preserving Asymmetric Quantization and Energy-Saving Bit-Slice Sparsity. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 701–715. doi:10.1109/HPCA61900.2025.00059

[10] Raghuraman Krishnamoorthi. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342* (2018).

[11] Alberto Delmás Lascorz, Sayeh Sharify, Isak Edo, Dylan Malone Stuart, Omar Mohamed Awad, Patrick Judd, Mostafa Mahmoud, Milos Nikolic, Kevin Siu, Zissis Poulos, and Andreas Moshovos. 2019. ShapeShifter: Enabling Fine-Grain Data Width Adaptation in Deep Learning. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture* (Columbus, OH, USA) *(MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 28–41. doi:10.1145/3352460.3358295

[12] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. 2023. RepQ-ViT: Scale Reparameterization for Post-Training Quantization of Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 17227–17236.

[13] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems*, P. Gibbons, G. Pekhimenko, and C. De Sa (Eds.), Vol. 6. 87–100.

[14] Lian Liu, Zhaohui Xu, Yintao He, Ying Wang, Huawei Li, Xiaowei Li, and Yinhe Han. 2024. Drift: Leveraging Distribution-based Dynamic Precision Quantization for Efficient Deep Neural Network Acceleration. In *Proceedings of the 61st ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) *(DAC '24)*. Association for Computing Machinery, New York, NY, USA, Article 140, 6 pages. doi:10.1145/3649329.3655986

[15] Yijiang Liu, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. 2023. NoisyQuant: Noisy Bias-Enhanced Post-Training Activation Quantization for Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 20321–20330.

[16] Jaehyeon Moon, Dohyung Kim, Junyong Cheon, and Bumsub Ham. 2024. Instance-Aware Group Quantization for Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16132–16141.

[17] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. 2020. Up or Down? Adaptive Rounding for Post-Training Quantization. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 7197–7206. https://proceedings.mlr.press/v119/nagel20a.html

[18] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295* (2021).

[19] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=8Wuvhh0LYW

[20] Man Shi, Vikram Jain, Antony Joseph, Maurice Meijer, and Marian Verhelst. 2024. BitWave: Exploiting Column-Based Bit-Level Sparsity for Deep Learning Acceleration. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 732–746. doi:10.1109/HPCA57654.2024.00062

[21] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 38087–38099. https://proceedings.mlr.press/v202/xiao23c.html

[22] Chenhao Xue, Chen Zhang, Xun Jiang, Zhutianya Gao, Yibo Lin, and Guangyu Sun. 2024. Oltron: Algorithm-Hardware Co-design for Outlier-Aware Quantization of LLMs with Inter-/Intra-Layer Adaptation. In *Proceedings of the 61st ACM/IEEE Design Automation Conference* (San Francisco, CA, USA) *(DAC '24)*. Association for Computing Machinery, New York, NY, USA, Article 31, 6 pages. doi:10.1145/3649329.3656221

[23] Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. 2023. SPIQ: Data-Free Per-Channel Static Input Quantization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 3869–3878.

[24] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. 2020. GOBO: Quantizing Attention-Based NLP Models for Low Latency and Energy Efficient Inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 811–824. doi:10.1109/MICRO50266.2020.00071