Algorithm-hardware co-design for Energy-Efficient A/D conversion in ReRAM-based accelerators

Chenguang Zhang[1,2], Zhihang Yuan[1,2], Xingchen Li[1,2], Guangyu Sun[1,3,*]
[1] School of Integrated Circuits, Peking University, Beijing, China
[2] School of Computer Science, Peking University, Beijing, China
[3] Beijing Advanced Innovation Center for Integrated Circuits, Beijing, China
zhangchg@stu.pku.edu.cn, {yuanzhihang, lixingchen, gsun}@pku.edu.cn

***Abstract*—Deep neural networks are widely deployed in many fields. Due to the in-situ computation (known as processing in memory) capacity of the Resistive Random Access Memory (ReRAM) crossbar, ReRAM-based accelerator shows potential in accelerating DNN with low power and high performance. However, despite power advantage, such kind of accelerators suffer from the high power consumption of peripheral circuits, especially Analog-to-Digital Converter (ADC), which account for over 60 percent of total power consumption. This problem hinders the ReRAM-based accelerator to achieve higher efficiency.**

Some redundant Analog-to-Digital conversion operations have no contribution to maintaining inference accuracy, and such operations can be eliminated by modifying the ADC searching logic. Based on such observations, we propose an algorithm-hardware co-design method and explore the co-design approach in both hardware design and quantization algorithms. Firstly, we focus on the distribution output along the crossbar's bit-lines and identify the fine-grained redundant ADC sampling bits. To further compress ADC bits, we propose a hardware-friendly quantization method and coding scheme, in which different quantization strategy was applied to the partial results in different intervals. To support the two features above, we propose a lightweight architectural design based on SAR-ADC. It's worth mentioning that our method is not only more energy efficient but also retains the flexibility of the algorithm. Experiments demonstrate that our method can reduce about $1.6 \sim 2.3\times$ ADC power reduction.

I. Introduction

With the wide adoption of deep neural networks (DNNs), an increasing number of applications utilize multiple DNNs to achieve state-of-the-art performance. The general and energy-efficient DNN inference acceleration demands surge at the edge such as automotive, robotics, and IoT devices. Among various accelerator architectures [1], [2], Processing-In-Memory (PIM) can avoid frequent data movement between memory and computing units showing great potential in breaking the bandwidth limitation and enhancing power efficiency. Additionally, the ReRAM-based analog computation [3] provides further advantages to PIM due to its features of nonvolatile, high density, and in-situ matrix-vector-multiplications (MVMs).

However, the overall energy efficiency of ReRAM-based PIM accelerators is impaired by the high overhead of analog-to-digital converters (ADCs) that digitize the analog results from crossbars (XBs). High-resolution ADC even consumes much more power and area than the ReRAM crossbar itself [4]. To eliminate the ADC bottleneck, existing solutions can be broadly classified into two types: algorithm-level techniques exploit weight sparsity and reduce the computation amount to reduce ADC usage [5], [6]. However, annoying retraining limits the range of applicable models at deployment. Architecture-level techniques replace high-resolution ADC with other highly customized circuits [7], [8] or ADC designs [9]. These adaptations are not so feasible in practice due to the consideration of manufacturing and analog circuit design complexity.

Targeting to improve energy efficiency while retaining the flexibility of the algorithm, in this paper, we adopt an approach at different levels: we find that a large proportion of ADC output bits are redundant during each A/D conversion process (Section II-E), and contribute little to the inference accuracy, due to the imbalanced distribution and inherent fault tolerance of DNNs. Based on this observation, we propose a configurable coding scheme to avoid redundancy generation and thereby improve energy efficiency, and implement the coding-decoding by a modified successive approximation register (SAR) ADC design. Our key contributions are:

- We analyze the distribution of analog values from crossbar bit-lines and quantify the redundancy in ADC output coding (Section III-A).
- We proposed a hardware-friendly quantization method, Twin range quantization, that can flexibly adjust quantization intervals based on the value distribution, and an efficient coding scheme for ADC outputs (Section III).
- We customize the SAR ADC control logic to support configurable quantization levels without changing analog circuits (Section III-D).
- An algorithm-hardware co-optimization scheme is proposed to search for the optimal parameters for each DNN layer, maximizing energy saving while minimizing accuracy loss (Section IV).

Our techniques require *no DNN retraining* and *no customization of ADC's analog part*. The proposed ADC modification is transparent to DNN models and is orthogonal to any other approaches based on model compression and auto-ML methods.

II. Background and Motivation

A. Analog PIM basic

PIM macro: An analog PIM macro is the basic unit of a ReRAM-based accelerator, consisting of a crossbar array

This work is supported by Key-Area Research and Development Program of Guangdong Province (2021B0101310002), NSF China(62032001), 111 Project (B18001) *Corresponding author.

979-8-3503-4859-0/DATE24/© 2024 EDAA

and peripheral circuits. The ReRAM crossbar functions as an MVM engine. The voltages applied to the word lines (WLs) multiply with the conductance of the cells. Currents of each cell accumulate along the bit lines (BLs), producing analog the MVM result, i.e., $\mathbf{I}_i = \sum_j \mathbf{G}_{i,j}\mathbf{V}_j$. ADC converts the currents at each BL to digital outputs.

Map Neural Networks to Accelerator: To map a neural network to the PIM macro, elements in input features are fed to DACs, and weights are stored as the conductance of the ReRAM cells. As shown in Fig. 1, a convolutional kernel (with $k \times k$ kernels, C_i input channels) is transformed to MVM in different sliding windows [3]. Resolution limitation of DAC and ReRAM cells [10] necessitate bitwise sliced mapping: a weight value is partitioned and mapped to cells on different BLs, input vectors are fed to DAC as bit slice cycle by cycle. When not fitting into a crossbar pair, a layer is partitioned and mapped to multiple crossbars. This sliced **intermediate** MVM results along BLs, cycle, and crossbars are merged by shift-and-add and accumulator in the digital domain, getting the final MVM result, which is sent to other functional units or crossbars for further processing.

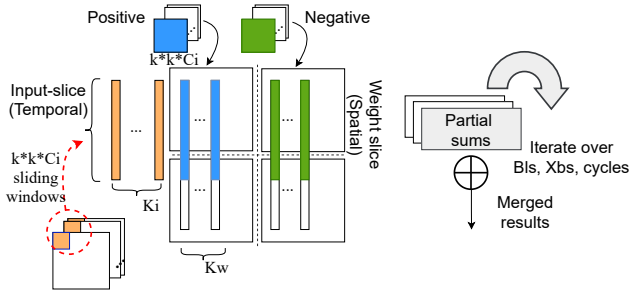

Fig. 1. Mapping convolutional layers. K_w, K_i are the bit-width of the weight and input activation respectively.

B. Quantization in Datapath

Quantization can effectively compress neural networks. The uniform quantization projects a real number x to a k-bit integer value x_q:

$$x_q = Q_k(x, \Delta) = \Delta_x \text{clamp}(\text{round}(\frac{x}{\Delta_x}), 0, 2^k - 1), \quad \text{where, } \Delta_x = \frac{(b-a)}{2^k - 1} \tag{1}$$

where the scaling factor Δ_x divides a given range of real values into several partitions, round$(\cdot)$ approximates a value, and clamp$(\cdot)$ limits x_q to the range $[a, b]$. For the ReRAM-based accelerator, quantization happens in the stage of algorithm mapping and inference stages, for better performance, low power, and less area. Note that ADC also functions as a quantizer and is possible to introduce extra quantization errors, which accumulate during the merging process. This behavior was not the designer's intent and needed to be specifically considered in the DNN quantization algorithm design.

C. ADC Preliminaries and Previous Works

Lossless conversion requires ADC with a resolution no less than $R_{ADC,ideal}$, i.e.,

$$R_{ADC,ideal} = \log_2(S) + R_{DA} + R_{cell} + \delta, \tag{2}$$

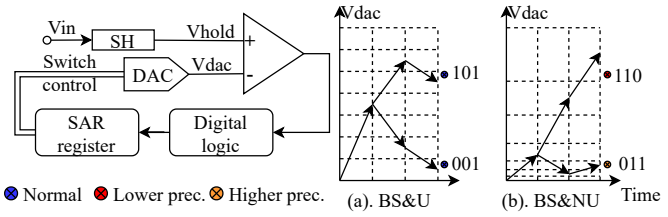

Fig. 2. Conventional SAR ADC with (a) uniform and (b) non-uniform grid

$$\delta = 0 \text{ if } R_{DA} \geq 1 \text{ and } R_{cell} \geq 1 \text{ else } -1,$$

where R_{cell}, R_{DA}, S are the number of bits a cell represents, the resolution of DAC, and the size of the crossbar, respectively. The overall ADC energy is formulated as:

$$E_{ADC,tot} = \overbrace{\text{len (input)} \times \text{size}_w}^{\#\text{ of MVMs / inference}} \times \overbrace{\frac{K_w}{R_{cell}} \times \frac{K_i}{R_{DA}}}^{\#\text{ of A/D conversions / MVM}} \tag{3}$$

$$\times E_{convert}, \tag{4}$$

where $E_{convert}$ is the energy per A/D conversion. As the power of ADC scales exponentially with ADC resolution R_{ADC} high-resolution ADCs are not competitive in terms of power and area. To overcome this challenge, solutions proposed by recent works can be categorized into three orthogonal levels: circuit, architecture, mapping, and algorithm.

- At the algorithm level, low-bit quantization [5], [6], [9] is widely used to reduce the bit-width of the weight and input activation (K_w, K_i); and pruning techniques [5], [11], [12] are used to reduce required MVMs per inference.
- At the mapping stage, weight encoding is introduced to obtain smaller conductance values (K_w) [3], [13]; weight slicing and input slicing is explored in RAELLA [13] to trade off ADC resolution (i.e., $E_{convert}$) and the number of A/D conversions per MVM.
- At the architecture level [3], [14], 1-bit DAC and 1-bit ReRAM cells ($R_{DA}, R_{cell} = 1$) are widely adopted, and $R_{DAC,ideal} = \log_2(S) + 1$. Multi-bit input fed to DAC as bit series, aggregated by shift-and-add operations.
- At the circuit level, to reduce $E_{convert}$, ADCs are customized with a non-uniform quantization scheme [9]; or discard, with all the operations conducted in the analog domain.

D. A/D operations and coding scheme

SAR ADC is the most suitable ADC for analog PIM macro, which is more energy-efficient than other ADCs in the target resolution and frequency [15], and strongly benefits from process technology scaling [16].

SAR logic performs a Binary Search (BS) to approximate the analog input (V_{hold}) with K reference voltages $V_{ref,idx(k)}$, represented by a K-bit output code, by comparing V_{hold} with $K-1$ predefined threshold voltages $V_{th,idx(k)}$ generated by a digital-to-analog converter (DAC). As shown in Fig. 2, the horizontal dash grid represents reference voltages, points ($\otimes$) represent the sampled values to be converted, and arrows consist of searching traces. Starting from $(10\ldots0)_2$th voltage level, output code is iteratively generated from MSB to LSB,

and depending on the previous bit, V_{hold} will be approximated to the upper or lower half of the voltage range as Eq. 5,

$$idx(k) = \begin{cases} (10,\ldots,0)_2, & k=1 \\ (\underbrace{D_{K-1}\cdots D_{K-k+1}}_{k-1}1\cdots 0)_2, & otherwise \end{cases} \quad (5)$$

where D_i is the i-th bit of the output code, $(\cdot)_2$ is the binary representation of ith reference voltage at k-th step. The conversion is completed in K cycles, forming a K-bit output code. We refer to each step as an *A/D operation*, and full progress as an *A/D conversion*. For a SAR ADC with K-bit resolution, energy consumption is proportional to the number of A/D operations,

$$E_{convert} = e_{\text{op}} N_{A/D\_ops} \quad (6)$$

where e_{op}, and $N_{A/D\_ops,i}$ represent energy per A/D operation, and the number of A/D operations required by each conversion, respectively.

For conventional Uniform (U) ADCs, threshold voltages are equally spaced ($(k-\frac{1}{2})\cdot \text{LSB}, k=1,\ldots,2^K-1$). In contrast, Non-uniform ADC (NU) performs BS on a customized grid, which has a higher density in the range with more values, as shown in Fig. 2 (b). With lower R_{ADC}, NU ADC achieves similar accuracy as U ADC, but with a large reduction in e_{op}, also lower ADC power consumption. Both of them have fixed N_{A/D_ops} for each conversion.

E. Motivation

Customizing ADCs and retraining DNN models restrict the algorithm flexibility of the accelerator. Therefore, we avoid modifying the analog characteristics of ADC. From the above dive into the ADC and related works, we notice that the factor, N_{A/D_ops}, which is determined by the searching process (i.e., coding) of SAR logic, is an indicator of A/D conversion efficiency. BS is a good choice for numerical comparison in general but may be not the optimal choice for a given distribution. This motivates us to explore the optimal coding scheme (with least N_{A/D_ops}) and configurable SAR logic to achieve adaptive energy-efficient A/D conversions, which is orthogonal to the above techniques.

III. Twin Range Quantization

In this section, we first identify the redundancy in the output coding scheme, then we introduce our Twin Range quantization and coding scheme to remove such redundancy. Finally, we present the implementation of configurable ADC coding/decoding.

A. Value distribution at BLs and redundancy

While it is commonly assumed that the activation in neural networks follows a Gaussian distribution [17], the value distribution of the ReRAM crossbar's BLs has not been thoroughly investigated. Fig. 3a illustrates the distribution of the BLs, revealing a highly imbalanced distribution where the majority of samples are concentrated in a small interval close to zero.

Generally, achieving lossless data bit width compression on such a skew distribution is challenging: we need to retain both the *numerical range* of large values while minimizing the *quantization error* of small values. Huffman coding, which is a classical idea of using variable-length codes to improve the coding efficiency of skewed data, inspires us to explore more efficient intermediate coding schemes used in A/D conversion. Additionally, the inherent fault tolerance of DNN enlightens us to reduce the precision of values that are not sensitive to inference accuracy. To this end, we identify samples of two kinds in the skewed distribution: (**T1**) the majority that concentrate in a narrow range (R1) and (**T2**) the minority that scatter in a wide range (R2), and correspondingly, two compression strategies.

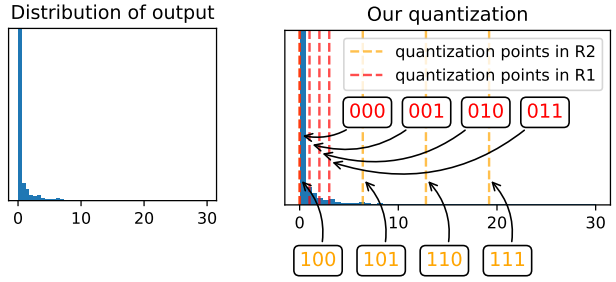

Fig. 3. (a) Distribution of the output of crossbar's BLs. (b) Twin ranges quantization.

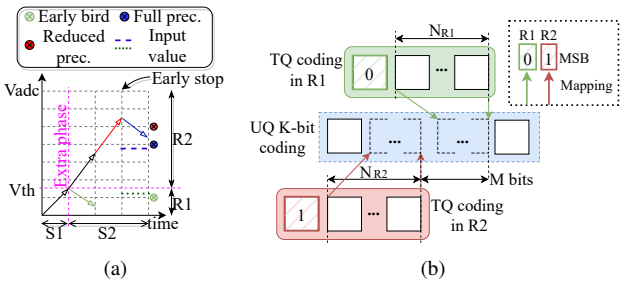

Fig. 4. (a) DAC output with two searching strategies. (b) The bit mapping of the ADC output code.

B. Quantization abstraction

From the perspective of A/D conversion, we can apply different SAR logic to the two ranges for less A/D operations:

- "Early birds": With the expense of a 1-bit comparison for each conversion, we can apply a biased search: if R1 is small and dense enough ("sweet spot"), most conversions can be completed with fewer A/D operations, but without precision loss. As shown in Fig 4a, "early bird" (green) in R1 is approximated just in $1+1$ step.
- "Early stopping": If T2 is sparse and not sensitive to inference accuracy, we can stop the conversion progress even though the output code is not fully determined, resulting in a reduced sensing precision but retaining the numerical range. In figure 4a, "early stopping" is made (red) after two steps before a full precision (blue) conversion completed.

Two strategies play different roles in balancing power and computation accuracy: "Early birds" earns the N_{A/D_ops} without precision degradation, but is required to find the "sweet spot". "Early stopping", has to determine whether to force a stopping or continue with full precision. Therefore, systematic analysis

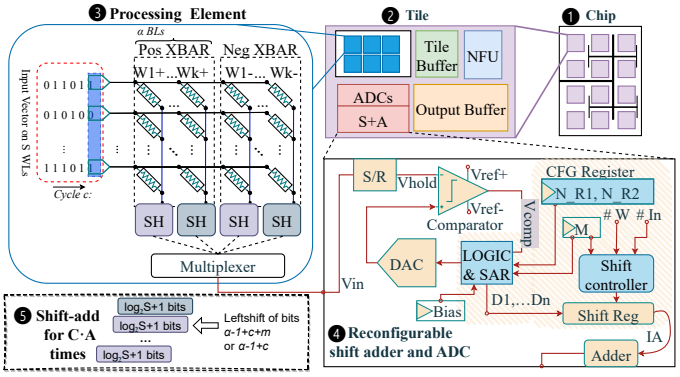

Fig. 5. Overall architecture

and calibration are required to maximize the gain in power and latency (Section IV).

From the perspective of the quantization algorithm, the above two strategies can be abstracted as a twin-range quantization (**TRQ**), a narrow range R1 to precisely quantize the small dense values, and a wider range R2 to cover the large sparse values; each range is quantized uniformly but is assigned a distinct scaling factor. Different from previous works, this non-uniform stems from the biased searching strategy, resulting in grids that align with the full precision searching grids, as seen in Fig. 3b (red dashed vertical grid for R1 and orange for R2). And the quantization function T_k can be formulated as Eq 7:

$$\mathrm{T}_k(x, \Delta_{\mathrm{R1}}, \Delta_{\mathrm{R2}}, N_{R1}, N_{R2}) = \begin{cases} Q_{k-1}(x, \Delta_{\mathrm{R1}}), x \le \theta \\ Q_{k-1}(x, \Delta_{\mathrm{R2}}), otherwise, \end{cases}$$

$$\theta = 2^{N_{R1}}\Delta_{\mathrm{R1}}, \mathrm{R1} = [0, \theta), \mathrm{R2} = [\Delta_{\mathrm{R2}}, +\infty), \quad (7)$$

where $Q_k(x, \Delta)$ performs k bit uniform quantization, with the scaling factor Δ (mentioned in Eq. 1). N_{R1}, N_{R2} and $\Delta_{\mathrm{R1}}, \Delta_{\mathrm{R2}}$ are quantization bits and scaling factors of the two ranges. **Note** that, as a type of post-training quantization scheme (PTQ) [18], the parameters can be easily calibrated to adapt to different DNNs by our algorithm-hardware co-design method (in Section IV), **no retraining is required!** Our quantizer is the *behavior abstraction of A/D conversion of SAR-ADC at BLs* and is orthogonal to the other quantization on W/A/I.

C. Coding scheme

We devise a bitmap to encode the values quantized by our TRQ. As shown in Fig. 4b, The most significant bit (MSB) indicates which range a value belongs to, "0" for R1 and "1" for R2. The remaining N_{Rx} bit(s) is an unsigned uniform coding for the approximated value in each range. To make the quantization grid align with the full precision grid, we make

$$\Delta_{\mathrm{R2}} = 2^M \Delta_{\mathrm{R1}}, \quad (8)$$

where m is an unsigned integer parameter. In the decoding stage, the output code with MSB=1 is shifted left by m bits to align with the value from R1.

As can be seen, **TRQ requires neither codebook nor analog DAC modification**, which greatly simplifies the hardware design, which will be discussed in Section III-D.

D. Hardware design

1) Overall architecture: In this paper, we follow the overall architecture of ISAAC [3], shown in Fig. 5. The accelerator (❶) consists of multiple tiles connected by a global bus. Each tile (❷) contains Neural Function Units (NFU), PE array, ADCs, Shift and Add (S+A) module, input buffer, and output buffer. The PE (❸) is used for accelerating the MVM operation. The accumulated product current at the BL end is converted to voltage values by the trans-impedance amplifiers (TIAs) and retained by the Sample-and-Hold (SH) circuit. We implement lightweight modifications to the SAR logic (❹) and the S+A module (❺). Here, the voltage vectors are initially encoded into compact digital codes by ADCs and subsequently decoded and accumulated by the S+A module in a column-wise and cyclical manner. ADCs and S+A modules operate in a time-division manner, shared by the PEs. The output feature maps are stored in the output buffer before being routed to other tiles or off-chip memory. As both the *coding* and *decoding* steps are handled by hardware, no additional *software overhead* is required. The *analog components* of the ADC remain unchanged, and the *original resolutions* of the SAR ADC are preserved.

2) Pipeline with Configurable Resolution:

*a) **SAR logic with twin ranges**:* We apply TRQ, i.e., customized searching strategy, by modifying the SAR logic. As described in Section III-B, first, the approximation logic checks whether the sampled voltage is located in R1 i.e., $\left[0, 2^{N_{R1}}\Delta_{\mathrm{R1}}\right)$, with an *extra detection phase*, then, BS is performed in R1 and R2, with step size $\Delta_{R1} = V_{ref}/2^{N_{R1}+M}$ and $\Delta_{R2} = V_{ref}/2^{N_{R2}}$, respectively, as shown in fig. 4a. We achieve this by simply adding extra approximation logic.

*b) **Shift and Add module**:* The modified ADC generates a compact digital code in the format mentioned in Section III-C, which can not be accumulated directly by an arithmetic adder. The radixes of the output code with MSB of 1 is 2^M times the output code with MSB equal to 0, which requires decoding before arithmetic operations. Thanks to the hardware-friendly encoding scheme, decoding requires only shift operations. We add an extra shift control to the existing S+A module to support the above decoding: Determining the MSB, before adding to the partial sum, the ADC's output code is shifted left by M bits (❺) (MSB is 1) or not (MSB is 0), and the MSB is discarded.

*c) **Configurable register**:* The configuration, including ADC output bit-width (N_{R1}, N_{R2}), non-uniform degree (M), an offset of $R1$ ($Bias$, used to adapt outlier conditions, as discussed in Section IV) are restored in the resister near the ADC and the Shift and Add module. The sensing precision can be configured as any bits below ADC resolution (R_{ADC}), and the non-uniform can be configured from 0 to $R_{ADC} - N_{R2}$. Such flexibility enables our design can adapt to various DNNs and be compatible with other algorithm-level compression techniques. Besides, our ADC design can be configured as either twin ranges mode or U ADC mode.

IV. Algorithm and Hardware Co-optimization

Section III introduces our TRQ algorithm and its corresponding hardware implementation. Our design is capable of compressing redundant A/D operations, thereby reducing ADC energy consumption and improving power efficiency. In this

section, we propose a parameter-searching algorithm to determine the configuration with the maximum energy reduction while meeting accuracy constraints.

A. Parameter Calibration

To pinpoint the optimal values for R1 (N_{R1}) and establish an 'early stop' threshold for R2 (N_{R2}, M, and Δ_{R2}), we employ a layer-by-layer parameter search as detailed in Algorithm 1. For each layer, the algorithm first determines the distribution type of the layer's BL output, then identifies the best M and Δ_{R2} for that layer. We sample C candidates uniformly from the interval $[\alpha\frac{y_{\max}}{2^{N_{R2}}-1}, \beta\frac{y_{\max}}{2^{N_{R2}}-1}]$, with α and β defining the search range. Given Δ_{R2}, M is iteratively tested over integers from 0 to $R_{ADC} - N_{R2}$. The optimal M minimizes energy consumption satisfies:

$$\begin{aligned} &M = \arg\min_{M} e_{\text{op}}(N \cdot \nu + \sum_{i=1}^{N_D} N_{A/D\_ops,i}), \\ &N_{A/D\_ops,i} = N_{R1} \text{ if } i \in R1, \text{else } N_{R2}, \\ &\nu = 1 \text{ if } bias = 0, \text{else } \nu = 2. \end{aligned} \tag{9}$$

Here, N_D, e_{op}, and $N_{A/D\_ops,i}$ represent the number of samples, energy per A/D operation, and the number of A/D operations required by each sample, respectively. The item $N \cdot \nu$ represents the overhead of the pre-detection phase. For each (M, Δ_{R2}) combination, quantization MSE is evaluated as

$$\min_{\Delta_{\text{R2}}} \text{MSE}(\text{T}_k(x, \Delta_{\text{R1}}, \Delta_{\text{R2}}, N_{R1}, N_{R2}), y). \tag{10}$$

to find the optimal Δ_{R2}. End-to-end accuracy is checked after optimization of all layers, the search process iterates over descending N_{R2} until the accuracy drops below the threshold θ. Finally, the resulting twin range quantization is compared with standard uniform quantization to select the best approach for each layer.

N_{R1} is deduced regarding the distribution type of the layer's BL output. For the *ideal cases* in Section III-B, which performs lossless A/D conversion in $R1$, We can deduce,

$$\Delta_{\text{R1}} = 1, N_{R2} + M = R_{ideal}, bias = 0, \nu = 1. \tag{11}$$

B. Compatibility for various types of distribution

The normal-like distribution with strong unimodality (i.e., low-variance), is also treated as the ideal case, except for Offset $= bias \cdot \Delta_{\text{R2}}$ is introduced as an extra threshold to identify the range $R1$ and $bias$, which is searched over the integer from 0 to $2^{N_{R1}} - 1$, is concatenated to the left side of the coding from $R1$ in the decoding progress. In other cases, such as weak unimodal, multi-modal, and flattened distribution, if a "sweet spot", $R1$ may not be found, "early stop" strategy is performed in both ranges. To reduce the search space, N_{R1} and N_{R2} set to the same values, and both Δ_{R1} and Δ_{R1} are searched to minimize quantization error Eq. 10.

Algorithm 1 Parameter Searching

Input: A pre-tuned quantized DNN model, prediction accuracy threshold θ
Output: $M, R1$

```
N_R1, N_R2 ← R_ADC − 1
Acc ← to evaluate the inference accuracy of a well-tuned quantized model
while True do
    for l = 1 → N do
        Judge the distribution types T of current layer
        for Δ_R2 ∈ range(α y_max/(2^N_R2 − 1), β y_max/(2^N_R2 − 1)) do
            if T is case_ideal or case_N then
                search for N_R1 and bias minimize Eq. 9
                set M according to Eq. 11
            else
                set N_R1 = N_R2
                search for M and bias minimize Eq. 9
            Execute T_k, and record MSE loss as Eq. 10
        Find optimal Δ_R2 as Eq. 10
    Acc' ← to evaluate the inference accuracy
    if Acc − Acc' > θ then
        Break
    else
        N_R2 ← N_R2 − 1
    Compare with uniform quantization with N_R2 bits
```

V. Evaluation

A. Experiment Settings

We adopt the ISAAC [3] as our baseline, and employ 128×128 crossbars with single-bit ReRAM cells, along with readily available data paths supporting 8b inputs/weights and 16b partial sums. The parameters of ReRAM and ADC are obtained from [19] and [20], respectively. The system operates at a clock frequency of 100MHz. For the digital part, we evaluate our design using data from CACTI6.5 with a 45 nm process for buffers and interconnects, The customized peripheral circuits are synthesized with FreePDK 45 nm library [21] using Design Compiler. We build a simulation based on DNN+NeuroSim [22], an open-source tool used to evaluate the performance of the neural network on PIM architecture.

To evaluate the accuracy and energy, we use ResNet-20 on CIFAR-10, ResNet-18, SqueezeNet1.1 on the ImageNet dataset and LeNet-5 on the MNIST dataset. We randomly select 32 images from the training dataset as calibration images to fine-tune the ADC configurations. The input activations and weights are applied with 8-bit symmetric uniform quantization to accommodate the hardware design. Their scaling factors are determined based on the maximum absolute values. For search space generation, we set $\alpha = 0.1$, $\beta = 1.2$, the number of range dividing candidates $C = 50$, m varies in $[0, 7]$. This results in up to 400 search candidates.

B. Algorithm Evaluation

Fig. 6 (a-b) shows the prediction accuracy using different ADC quantization bit-width, where "f/f" and "8/f" represent the model with no quantization (float point) and 8-bit quantization on *weights* and *activations*, respectively. And "8,7,6,5,4," represents the maximum allowed length of *ADC*, i.e., upper bound of N_{R1}, N_{R2}. Compared with low-bit uniform quantization, TRQ can achieve better prediction accuracy. For instance, on ResNet-20 and CIFAR-10 datasets, TRQ achieves 91.09% prediction

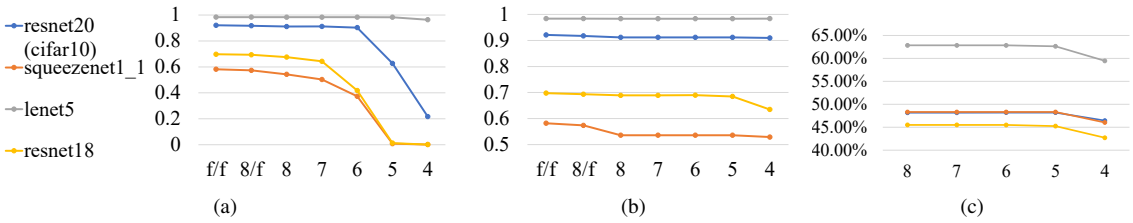

Fig. 6. Evaluation of algorithm (a) Accuracy w.r.t. ADC resolution without TRQ and (b) with TRQ; (c) Remained A/D operations with TRQ.

accuracy at 4-bit ADC, while to achieve similar accuracy, U ADC should be at least with 7-bits resolution.

C. Hardware Evaluation

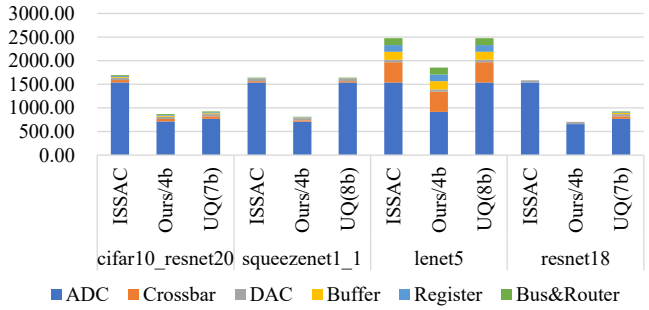

Fig. 7. Power breakdown of ReRAM-based accelerator.

In this part, we study how the percentage of ADC dynamic reading efficiency is reduced by TRQ. Fig. 6 (c) shows the percentage of ADC dynamic reading energy is reduced by TRQ compared with A/D operations required by original full precision ADC (8bit/conversion). The batch size is rescaled for each model across DNNs to keep overall energy in the same range. The percentage of the ADC energy consumption is reduced to $42\% \sim 62\%$ on average (i.e., $1.6 \sim 2.3\times$ improvement) with TRQ. Fig. 7 shows the overall power breakdown of model inference for the above four DNN workloads (the upper bound of $N2$ is set as 4bit). We compare 4bit TRQ ADC and U ADC that use the minimum bit-width to achieve similar accuracy. It can be seen although the original ADC bit-width is unchanged, TRQ can reduce the ADC power consumption significantly.

VI. Conclusion

This paper presents an energy-efficient quantization scheme for ReRAM-based neural network accelerators, including the TRQ algorithm, hardware implementation, and the algorithm-hardware co-design. Our design is capable of compressing redundant A/D operations, thereby improving power efficiency with negligible accuracy loss. Our method can be easily integrated into existing RRAM-based neural network accelerators, requiring no modification of the analog part of the ADC, but only a lightweight modification of the digital logic of the ADC. And the design is transparent to the DNN models, requiring no DNN retraining, no overhead for en-/decoding, and *original resolutions* of the ADC is preserved, Such flexibility enables our design can adapt to various DNNs and other hardware optimizations and model compression techniques without any modification.

References

[1] M. Gobulukoglu, C. Drewes et al., "Classifying Computations on Multi-Tenant FPGAs," in DAC, 2021, pp. 1261–1266.

[2] A. Boroumand, S. Ghose et al., "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks," PACT, vol. 2021-Septe, pp. 159–172, 2021.

[3] A. Shafiee, A. Nag et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ISCA, vol. 44, no. 3, pp. 14–26, 2016.

[4] X. Li, Z. Yuan et al., "Tailor: Removing Redundancy in Memristive Analog Neural Network Accelerators," in DAC. IEEE, 2022.

[5] S. Qu, B. Li et al., "ASBP: Automatic Structured Bit-Pruning for RRAM-based NN Accelerator," in DAC. IEEE, 2021, pp. 745–750.

[6] Z. Zhu, H. Sun et al., "A Configurable Multi-Precision CNN Computing Framework Based on Single Bit RRAM," in DAC. IEEE, 2019, pp. 1–6.

[7] T. Chou, W. Tang et al., "CASCADE: Connecting RRAMs to Extend Analog Dataflow In An End-To-End In-Memory Processing Paradigm," in Proc. MICRO, New York, NY, USA, 2019, pp. 114–125.

[8] P. Chi, S. Li et al., "Prime: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," ISCA, vol. 44, no. 3, pp. 27–39, 2016.

[9] H. Sun, Z. Zhu et al., "An energy-efficient quantized and regularized training framework for processing-in-memory accelerators," in DAC. IEEE, 2020, pp. 325–330.

[10] IRDS. (2021) INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS, MORE THAN MOORE WHITE PAPER. [Online]. Available: https://irds.ieee.org/editions/2021/more-moore

[11] X. Ma, G. Yuan et al., "Tiny but accurate: A pruned, quantized and optimized memristor crossbar framework for ultra efficient dnn implementation," in DAC. IEEE, 2020, pp. 301–306.

[12] H. Shin, R. Park et al., "Effective zero compression on ReRAM-based sparse dnn accelerators," in DAC, New York, NY, USA, 2022, p. 949–954.

[13] T. Andrulis, J. S. Emer, and V. Sze, "Raella: Reforming the arithmetic for efficient, low-resolution, and low-loss analog PIM: No retraining required!" in ISCA, 2023.

[14] Q. Liu, B. Gao et al., "33.2 A fully integrated analog ReRAM based 78.4 TOPS/W compute-in-memory chip with fully parallel MAC computing," in ISSCC. IEEE, 2020, pp. 500–502.

[15] B. Murmann, "Energy limits in current a/d converter architectures," ISSCC Short Course, 2012.

[16] ——, "Energy limits in a/d converters," in 2013 IEEE Faible Tension Faible Consommation, 2013, pp. 1–4.

[17] Y. Choukroun, E. Kravchik et al., "Low-bit Quantization of Neural Networks for Efficient Inference," in ICCV Work. 2019, 2019, pp. 3009–3018.

[18] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in NeurIPS '2019, H. Wallach, H. Larochelle et al., Eds., vol. 32, 2019.

[19] P. Yao, H. Wu et al., "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641–646, 2020.

[20] H. Chen, X. Zhot et al., "A> 3ghz erbw 1.1 gs/s 8b two-sten sar adc with recursive-weight DAC," in 2018 IEEE Symp. VLSI Circuits. IEEE, 2018, pp. 97–98.

[21] J. E. Stine, I. Castellanos et al., "FreePDK: An open-source variation-aware design kit," in 2007 IEEE Int. Conf. Microelectron. Syst. Educ. IEEE, 2007, pp. 173–174.

[22] X. Peng, S. Huang et al., "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies," in 2019 IEDM, 2019, pp. 32.5.1–32.5.4.