Mapping Binary ResNets on Computing-In-Memory Hardware with Low-bit ADCs

Yulhwa Kim, Hyungjun Kim, Jihoon Park, Hyunmyung Oh, Jae-Joon Kim
Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea
{yulhwa.kim, hyungjun.kim, jihoon.park, hyunmyung.oh, jaejoon}@postech.ac.kr

***Abstract*—Implementing binary neural networks (BNNs) on computing-in-memory (CIM) hardware has several attractive features such as small memory requirement and minimal overhead in peripheral circuits such as analog-to-digital converters (ADCs). On the other hand, one of the downsides of using BNNs is that it degrades the classification accuracy. Recently, ResNet-style BNNs are gaining popularity with higher accuracy than conventional BNNs. The accuracy improvement comes from the high-resolution skip connection which binary ResNets use to compensate the information loss caused by binarization. However, the high-resolution skip connection forces the CIM hardware to use high-bit ADCs again so that area and energy overhead becomes larger. In this paper, we demonstrate that binary ResNets can be also mapped on CIM with low-bit ADCs via aggressive partial sum quantization and input-splitting combined with retraining. As a result, the key advantages of BNN CIM such as small area and energy consumption can be preserved with higher accuracy.**

***Index Terms*—NN accelerator, computing in memory, analog computing, hardware-NN co-design**

I. Introduction

Computing-in-Memory (CIM)-based hardware has been drawing attention as an efficient computing platform for Neural Networks (NNs) by enabling embarrassingly parallel processing of multiply-accumulate (MAC) operations [1], [2]. CIM typically utilizes analog signals for the computation so that Digital-to-Analog Converter (DAC)/Analog-to-Digital Converter (ADC) are often required to interface with the external digital logic. The area and energy of the interface circuits increase significantly with respect to the increase of bit resolution [3], and hence it is important to lower the resolution of the interfaces to improve the area/energy efficiency of the CIM-based NN accelerators.

To mitigate the ADC/DAC overhead, a large number of CIM-based NN accelerators started to adopt Binary Neural Networks (BNNs), which use 1-bit neuron and weight values [4]. Thanks to the extremely quantized inputs and weights, CIM-based BNN accelerators can use 1-bit word-line (WL) drivers as the input interface circuits and low precision ADCs as the output interface circuits [5]–[10].

On the other hand, BNNs usually suffer from accuracy degradation compared to the higher-precision models. To improve the accuracy of BNNs, various strategies have been proposed [11]–[13]. Among them, the skip connection used by ResNet greatly improves the performance of BNNs in particular [12]–[14]. [12] reported that the top-5 accuracy of ImageNet classification was 79.9% for binary ResNet-18 and 71.0% for plain BNN. Therefore, adopting high-resolution skip connection is very important for improving the accuracy of BNNs, and hence recent BNN algorithms have been extensively using the skip connection to achieve acceptable accuracy.

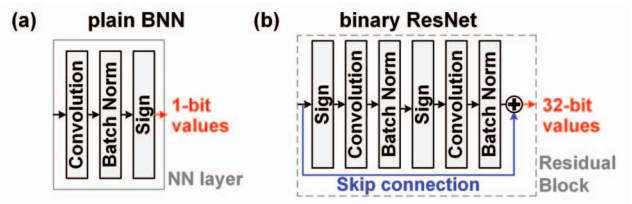

Fig. 1. The network architecture of (a) plain BNN (b) binary ResNet.

Despite the algorithmic advantage of the skip connection in BNNs, the skip connection puts a burden on the design of CIM hardware as it requires high-resolution values even in BNNs. In conventional BNNs (Fig. 1(a)), binary activation values are passed to the next layer so that implementation of simple input/output interfaces is possible for CIM BNN hardware. In contrast, a skip connection accumulates 32-bit pre-activation values over the entire forward propagation (Fig. 1(b)) so that it requires high-precision values even in BNNs.

This paper proposes a hardware/algorithm co-design methodology to implement binary ResNet on CIM-based accelerators. The proposed method aims at reducing the required resolution of output interface as much as possible to increase the area/energy efficiency of CIM-based accelerators. The main contribution of this paper is as follows:

1. We analyze the issues in mapping binary ResNets on CIM hardware and compare them with the case of plain BNNs.
2. We demonstrate that binary ResNets can be mapped on CIM hardware with low-bit ADCs via aggressive partial sum quantization and input-splitting with retraining.
3. We show that the proposed design saves area and energy compared to conventional designs.

II. Preliminaries

A. Binary Neural Networks

BNNs construct the network with binary neurons and weights. Binary values are generated from the following sign function:

$$Sign(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \quad (1)$$

Binary neurons are obtained by applying batch normalization followed by the sign function (Eq. (1)) to the MAC results (x)

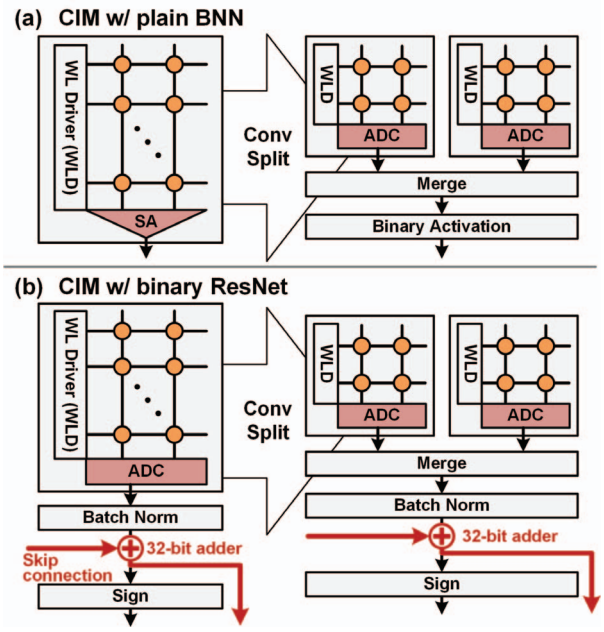

Fig. 2. CIM implementation of (a) plain BNN and (b) binary ResNet. We use single array to implement convolution when the array is large enough (left). Otherwise, convolution is split and multiple arrays are used for the convolution (right).

of the convolution. The batch normalization calculates mean (μ) and std (σ) of MAC results and conducts the following operation [15]:

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{2}$$

Here, γ and β are trainable parameters, and ϵ is a constant for numerical stability. In this paper, we call the combination of sign function (Eq. (1)) and batch normalization (Eq. (2)) as binary activation. Binary activation generates binary neuron values using the output of convolutions, and the equation of the binary activation is as follows:

$$BinAct(x) = \begin{cases} +1, & \text{if } x \geq th \\ -1, & \text{if } x < th \end{cases}, \quad th = \mu - \beta \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma} \tag{3}$$

Since the binary activation is a simple thresholding function, it can be implemented on the memory array using sense amplifiers (SAs) [16].

III. Characteristics of BNN Mapping on CIM

A. Mapping plain BNNs on CIM

CIM accelerates MAC operations inside memory array. If single array is large enough to implement the entire convolution, the array can generate 1-bit output neuron values with the MAC results. In this case, memory array can use SAs which can be regarded as 1-bit ADCs to generate the outputs (Fig. 2(a)-left) [16]. However, due to physical limit, the number of rows for an array is usually smaller than 1K, and hence the number of input data which can be processed by the array is also limited. If the fan-in, the number of inputs for a dot product to generate an output neuron, exceeds this limit, the convolution is split so that each split can fit on a single array, and multiple arrays are used to implement the convolution (Fig. 2(a)-right) [9].

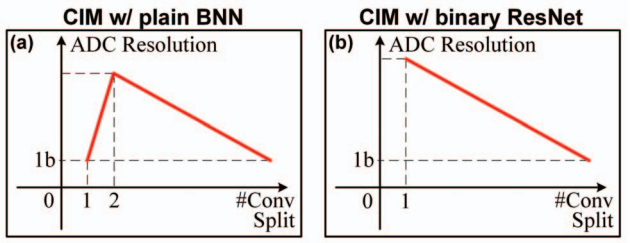

Fig. 3. The required resolution of ADCs for (a) CIM with plain BNN and (b) CIM with binary ResNet. The ADC resolution becomes the maximum value when the number of convolution split is 2 for plain BNN and 1 for binary ResNet.

With convolution split, analog partial sums need to be converted to digital values by ADCs because partial sums should be merged to complete the entire convolution. The required resolution of ADC depends on the range of partial sums. When the fan-in constraint of the array is N_{in}, the possible range of partial sum is as follows:

$$[-N_{in}, +N_{in}] \tag{4}$$

This is because the multiplication result between a binary neuron and weight is -1 or +1 and the array accumulates N_{in} number of multiplication results. Therefore, the array requires ADCs with $log_2(N_{in}) + 1$ bit for the exact conversion. The number of input processed by each array decreases as the convolution split increases, so the required resolution of ADC becomes the maximum value when the convolution split is 2 and keeps decreasing as the convolution split increases (Fig. 3(a)).

The area and energy overhead of ADC increase significantly as the bit resolution increases, so previous works tried to lower the resolution of ADC by adopting partial sum quantization and retraining [6]–[8]. Recently, RRAM-based CIM chips with 1-bit ADC has been reported for plain BNNs so ADC resolution requirement for plain BNNs can be basically reduced to the minimal value [7], [8]. In [7], [8], plain BNNs achieved acceptable accuracy with 1-bit partial sum quantization. It is relatively easy to lower ADC resolution for plain BNNs because a convolution output is solely used as the input of the binary activation block which generates a binary neuron value. However, in binary ResNets, the high precision convolution output is also required for the skip connection, and hence it is not straightforward to lower the resolution of ADCs. In the next section, we discuss the new challenges in mapping binary ResNets on CIM hardware and highlight the differences from mapping plain BNNs.

B. Mapping binary ResNets on CIM

In binary ResNet, the skip connections are used to propagate 32-bit information throughout the network. This high-resolution

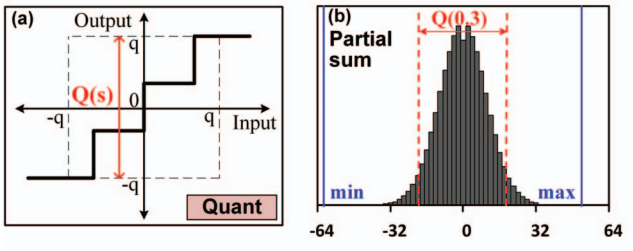

Fig. 4. (a) Linear quantization function used for partial sum quantization. Q(s) indicates quantization range parameterized with Q-scale (s). (b) Distribution of partial sums generated from CIM arrays with fan-in constraint 64. (Bold-blue line: true min/max of parital sum, dashed-red line: quantization bound when s is 0.3)

Algorithm 1 Algorithm of Q-scale optimization

Input: pre-trained binary ResNet (Net), fan-in constraint of array (N_{in}), bit-resolution of ADC (Bit_{ADC})
Output: best accuracy (acc_{best}), optimized Q-scale (s_{opt})
$acc_{best} = 0$, $s_{opt} = 0$
for $s \in (0, 1]$ **do**
  inference $Net(N_{in}, Bit_{ADC}, s)$
  $acc \leftarrow$ accuracy of $Net(N_{in}, Bit_{ADC}, s)$
  if $acc \geq acc_{best}$ **then**
    $acc_{best} \leftarrow acc$, $s_{opt} \leftarrow s$
  end if
end for
Return: acc_{best}, s_{opt}

data path substantially increases the network accuracy [13], but it leads binary ResNets to have different characteristics from plain BNNs. While the convolution output of plain BNN is only used to generate binary neurons, in binary ResNet, the convolution output also needs to be added to the high-resolution skip connection. Therefore, even when a single array is large enough to complete a convolution operation, binary ResNet still requires high-resolution ADC. In such a case, the required resolution of ADC is $log_2(K) + 1$ (K: fan-in of the convolution). Then, the convolution result is converted to a digital value and is added with the 32-bit skip connection value (Fig. 2(b)-left). ADCs are also required for binary ResNet with convolution splits to generate partial convolution values as well (Fig. 2(b)-right).

Since ResNets improve the accuracy using the high-resolution skip data paths, the accuracy is sensitive to the changes in the skip connection values. As a result, quantization of the skip connection values is harder than the quantization of neuron and weight values [17]. In other words, as convolution output is fed to skip connection, it is hard to lower the resolution of ADCs for the binary ResNets. We, however, made an interesting observation that using multiple arrays with convolution split could help to reduce the ADC resolution in binary ResNet CIM designs unlike plain BNN CIM designs in which the required ADC resolution is minimum when the number of convolution split is 1. As the number of convolution splits increases, the ADC output from each array is merged together to form the final convolution output, which effectively increases the resolution of convolution output. In addition, the ADC resolution for accurate partial sum generation decreases with the increased number of convolution splits (Fig. 3(b)). Based on the observations, we discuss several partial sum quantization methodologies to map binary ResNet on CIM hardware in the next section.

IV. Mapping binary ResNet with Low-bit ADC

A. Partial sum quantization with scaling and retraining

In each CIM array, an analog partial sum output is linearly quantized with an ADC in general. The quantization function with variable resolution and range is shown in Fig. 4(a). To maximize the hardware efficiency, the resolution of the quantization needs to be minimized as it decides the resolution of ADC. To find the best quantization range to maintain the network accuracy with low-bit ADCs, we first observed the distribution of partial sums generated from memory arrays (Fig. 4(b)). The distribution of partial sums is close to the gaussian distribution with mean 0. In addition, most of the partial sum values are within the tighter range than the theoretical maximum range. Therefore, we used the tighter quantization range to reduce the ADC resolution [6].

In this work, we tried to find the best range empirically. First, we introduced Q-scale (s), a scaling factor for the quantization range. The quantization range (Q) with Q-scale is as follows:

$$Q(s) \in [-N_{in} \cdot s, +N_{in} \cdot s], \quad s \in (0, 1] \tag{5}$$

When the Q-scale is 1, the quantization range covers all possible values of partial sums, and the range becomes narrower as the Q-scale decreases. Note that the partial sums which are out of the range are mapped to the boundary values. To find the best Q-scale, we simulated target tasks (e.g. image classification) with different Q-scales and resolutions (Algorithm 1).

The use of the Q-scale reduces the ADC resolution but the changes made for the binary ResNets to map them on multiple arrays using convolution split and partial sum quantization degrade the accuracy of the network significantly. To compensate the accuracy loss, we retrain the network [1]. In the retraining process, using the appropriate Q-scale was critical to achieve sufficient accuracy. Without setting a good quantization range, the accuracy was not fully recovered even with retraining, but the quantization-aware training with the optimized Q-scale fully recovered the accuracy even with the 2-bit partial sum quantization. Meanwhile, our experiments showed that 1-bit partial sum quantization caused large accuracy loss even with the optimal quantization-aware retraining.

B. Modified input splitting

To enable 1-bit partial sum quantization on binary ResNet, we adopted the input splitting (IS) methodology [8], which enabled the use of 1-bit ADCs in the CIM hardware for plain BNNs. The IS methodology completes a binary activation (Eq. (3)) in each array and recovers the accuracy using retraining. When the IS methodology is applied to binary ResNet

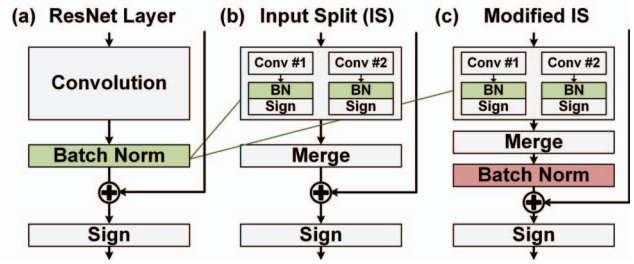

Fig. 5. (a) ResNet layer with skip connection. (b) Input-split layer with skip connection. (c) Modified input-split layer with skip connection.

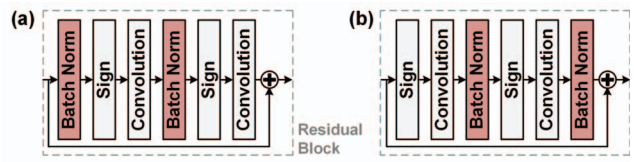

Fig. 6. Two typical residual blocks with different positions of skip connection.

mapping, the batch normalization after the convolution in the original binary ResNet (Fig. 5(a)) is merged with binary sign function in each convolution split array as shown in Fig. 5(b). However, the conventional IS-based binary ResNet showed poor accuracy even with retraining. To identify the reason for the failure, we compared two typical residual block design styles (Fig. 6) [13], [14]. In the first design (Fig. 6(a)), the convolution output is directly added to the skip connection, but in the second design (Fig. 6(b)), the convolution output is followed by the batch normalization before being added to the skip connection. When the classification accuracy was measured for the CIFAR-10 dataset, the accuracy of the structure in Fig. 6(b) was 0.04-1.84% higher than that of the structure in Fig. 6(a). The results provide an intuition that it is important to apply a batch normalization to the convolution output before adding it to the skip connection in binary ResNets. Therefore, we propose a modified input splitting methodology which uses an additional batch normalization before the addition with the skip connection value (Fig. 5(c)). The proposed architecture successfully recovered the accuracy with retraining. Detailed results will be shown in the experimental results (Section V).

V. Experimental Results

In this section, we evaluate the proposed CIM implementation methodologies for binary ResNets in terms of classification accuracy and area/energy efficiency.

A. Accuracy evaluation setups

We designed the binary ResNet based on the ResNet-20 architecture for accuracy evaluation with the CIFAR-10 dataset. The network consists of 3 types of residual blocks; the block information is summarized in Table I. The first input convolution layer and the output fully-connect layer were not included in the CIM implementation. We compared several mapping algorithms: partial sum quantization with and without Q-scale optimization, the quantization-aware retraining (Q-train), and the modified IS. CIM arrays with fan-in constraints 256, 128, and 64 were evaluated. The number of convolution splits for each fan-in constraint is summarized in Table I.

TABLE I
Block structure and convolution split of binary ResNet. (N_{in}: fan-in constraint of CIM-unit)

block type	output size	#layers /block	kernel size	#convolution split per layer		
				N_{in}=256	N_{in}=128	N_{in}=64
block1	32×32	2	3×3×64	3	6	9
block2	16×16	2	3×3×128	6	9	18
block3	8×8	2	3×3×256	9	18	36

TABLE II
Classification accuracy of BNNs on CIFAR-10.

network	accuracy	# parameters	OPs
binary ResNet	90.96%	4.3M	0.64G
plain BNN [4]–[8]	88.60%	14M	0.61G

For all training cases, the same training conditions were used. Input images were pre-processed with whitening; weights of the network were initialized with values sampled from uniform distribution in $[-1, 1]$. The straight-through estimator was adopted to estimate the gradient of discrete sign function [4]. We used the adam optimizer with initial learning rate 0.06. Learning rate was decayed by 0.5 every 50 epochs. The size of a mini batch was 200 and the networks were trained for 500 epochs.

B. Classification accuracy results

The binary ResNet showed 90.96% classification accuracy. It is 2.36% higher than that of the plain BNN used in [4]–[8] with 70% less parameters and similar OPs (Table II).

Partial sum quantization and Q-scale (without retraining). Table III shows the accuracy and the optimal Q-scale (s) of the binary ResNet with partial sum quantization. The exact analog-to-digital conversion requires 7-9 bit ADCs for arrays with fan-in constraint 64-256, but the partial sum quantization reduced it to 5 bit with network accuracy above 90% (less than 1% accuracy drop compared to the non-split model). If we use the optimized Q-scale based on the Algorithm 1, 4 bit was sufficient to maintain the accuracy. Optimal s does not exceed 0.50, so the quantization range is much tighter than the entire range of partial sums. In addition, optimal s tends to become smaller as the ADC resolution decreases. It indicates that the Q-scale tends to abandon the outliers when the resolution of ADCs is insufficient. Without retraining, the accuracy became lower than 90% when the resolution is under 3 bit, so retraining is required to further reduce the ADC resolution.

Q-Train. The quantization-aware training results are summarized in Fig. 7. The Q-Train significantly increased the accuracy with Q-scale s playing a critical role to achieve sufficiently high accuracy. While the Q-Train with s=1 failed to achieve the target accuracy even with 3-bit quantization, the Q-Train with optimal s successfully achieved 90% classification accuracy

TABLE III
ACCURACY OF BINARY RESNET WITH PARTIAL SUM QUANTIZATION.
(BOLD TEXT: OPTIMAL Q-SCALE VALUES)

ADC Resol	N_{in}=256		N_{in}=128		N_{in}=64	
	s=1	s=opt	s=1	s=opt	s=1	s=opt
5b	90.0%	90.5%(**0.50**)	90.5%	90.9%(**0.35**)	90.0%	90.8%(**0.50**)
4b	85.9%	90.6%(**0.30**)	88.5%	90.6%(**0.40**)	88.5%	90.2%(**0.45**)
3b	29.9%	89.0%(**0.30**)	54.7%	89.2%(**0.40**)	53.9%	89.5%(**0.40**)
2b	12.1%	80.1%(**0.20**)	15.6%	81.0%(**0.25**)	16.2%	81.6%(**0.30**)
1b	11.0%	17.7%(**0.20**)	10.7%	27.1%(**0.20**)	11.0%	24.9%(**0.25**)

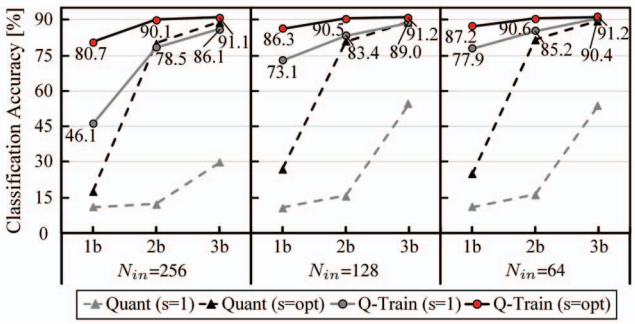

Fig. 7. Comparison of classification accuracy of binary ResNet with (without) the Q-scale (s) optimization and with (without) the retraining. The Q-train and the Quant represent the partial sum quantization with and without retraining, respectively. (numbers: accuracy obtained with Q-train)

with 2-bit quantization only. However, for 1-bit quantization, Q-train failed to recover accuracy loss.

Modified IS. Although the IS scheme enabled to use 1-bit quantization for plain BNNs [8], it failed on binary ResNet with 6-15% accuracy drop (Fig. 8). On the other hand, modified IS scheme accomplished 1-bit quantization with 90.0-90.7% classification accuracy (Fig. 8). The only difference between the IS and the modified IS is the addition of the batch normalization before the skip connection. This result confirms that a careful design is needed for the input which is added to the high-resolution skip connection in binary ResNets.

The simulation results show that the proposed methodologies successfully lowered the ADC requirement; 1-bit ADCs can be used for binary ResNets as well as plain BNNs without noticeable accuracy degradation.

C. Hardware evaluation setups

The CIM architecture for processing binary ResNets consists of chain of layer blocks (Fig. 9). A layer block produces binary neuron values and 32-bit skip connection values. The skip connection data is bypassed in certain blocks when it is not needed in the blocks. Within a layer block, CIM-units conduct MAC operations for convolution splits. Then, digital logic post-processes the output of the CIM-units or conducts the computations for skip connection. Final computation results are propagated to the next layer block.

We used the NeuroSim simulator [18] for energy and area analysis in 32nm technology. Multiple sense-amp based flash-type ADC was used and every 8 columns share an ADC [7].

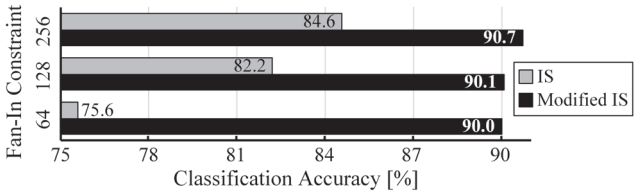

Fig. 8. Classification accuracy of binary ResNet with IS and modified IS.

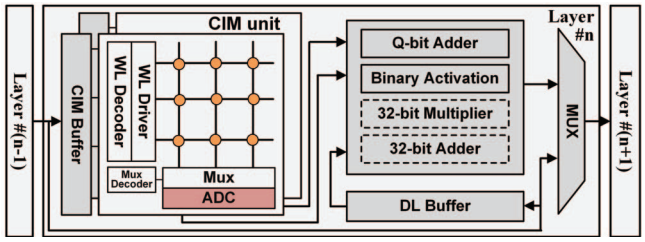

Fig. 9. The overall architecture of CIM hardware for binary ResNet. (Q: ADC resolution, Dashed box: components for the skip connection)

A couple of 1T1R RRAM cells are used to represent a binary weight [5]. The modified IS used 12.5% of total memory cells to implement threshold values for the binary activation [16]. We used flip-flops for CIM buffers and SRAM for Digital Logic (DL) buffers. In addition, additional 32-bit arithmetic unit and DL buffer were implemented for the two 1x1 convolutions for the skip connections to deal with channel expansion.

D. Hardware evaluation results

Array-level energy and area. Before evaluating the area and energy of the entire CIM hardware, we first analyze the area and energy of an array (the CIM-unit in Fig. 9). In the experiment, size of the memory array is fixed to 128×64. As shown in Fig. 10, ADC is the major source of area and energy consumption when the resolution of ADC is 5 bit. The ADC consumes 83.8% and 96.1% of the CIM-unit area and energy, respectively. However, the reduced ADC resolutions by the proposed methodologies alleviate the ADC overhead substantially. The reduction in the ADC resolution from 5 bit to 1 bit saved up to 79.2% and 93.0% of CIM-unit area and energy consumption, respectively.

Total Area and Energy. We analyze the total area and energy consumption of the entire binary ResNet CIM hardware using single image inference (Fig. 11 and Fig. 12). 3 different fan-in constraints (64, 128, 256) for memory arrays are evaluated. As expected, as the ADC resolution decreases, the total area and energy are also reduced significantly. In case of area, the CIM-unit area is much larger than the area of digital logic part for the higher ADC resolution cases because of the ADC area overhead. For the lower-bit ADC designs (1 or 2-bit), ADC area becomes smaller so that the area of CIM-unit and digital logic part becomes comparable. The energy consumption of CIM-unit and digital logic is comparable when ADC resolution is high but the digital logic tends to consume much larger energy than the CIM-unit for lower bit ADC cases.

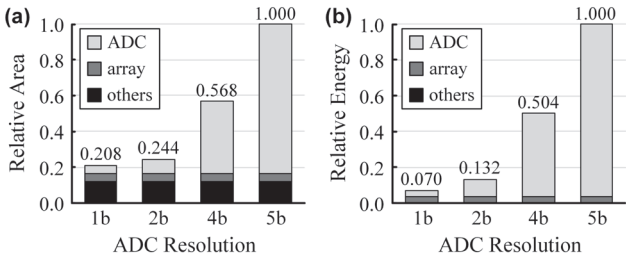

Fig. 10. (a) Area and (b) energy breakdown of CIM-unit with different ADC resolution (1b: Modified IS, 2b: Q-Train (s=opt), 4b: Quant w/o retraining (s=opt), 5b: Quant w/o retraining (s=1)). The size of memory array is fixed to 128×64.

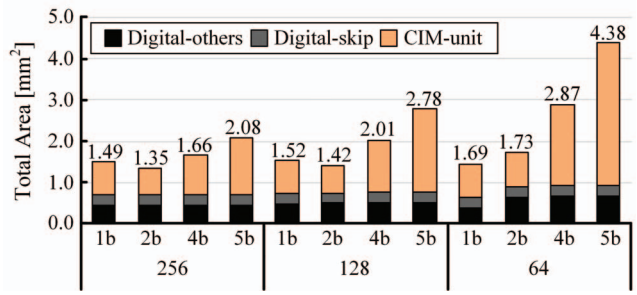

Fig. 11. Total area of CIM hardware with different ADC resolution (1b: Modified IS, 2b: Q-Train (s=opt), 4b: Quant w/o retraining (s=opt), 5b: Quant w/o retraining (s=1)). Memory arrays with fan-in constraint 256, 128, and 64 are adopted.

In most cases, CIM hardware with 1 bit ADC (modified IS) achieved the best hardware efficiency in terms of area and energy. However, when the fan-in constraint of each memory array is 256 or 128, CIM hardware with 2 bit ADC (Q-Train) achieved the comparable area and energy to those of 1 bit ADC-based design. It is because the modified IS used 12.5% of memory cells for implementing the in-memory batch normalization [16] to enable 1 bit ADC-based design.

VI. Conclusion

In this paper, we investigated the unique challenges in mapping binary ResNets to CIM hardware, which occurs due to high-precision skip connections. Based on the analysis, we proposed new mapping methodologies and demonstrated that the ADC resolution in the binary ResNet CIM hardware could be reduced to very low bits (1-2 bits). The proposed methodologies saved the area and energy consumption of CIM hardware up to 67.0% and 65.5% compared to naive implementation with higher ADC resolution.

Acknowledgment

This work was in part supported by Samsung Research Funding Center under Project Number SRFC-TC1603-51 and Samsung Electronics Co., Ltd., and the Nano-Material Technology Development Program through the National Research Foundation of Korea (NRF) funded by the MSIT (NRF2016M3A7B4910249).

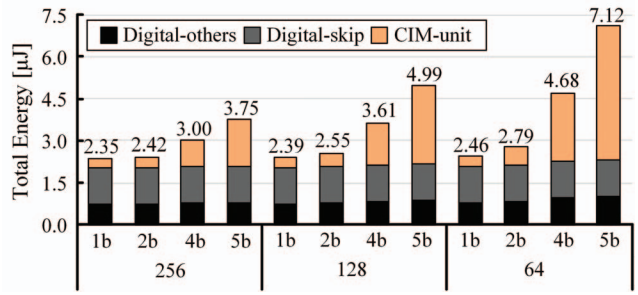

Fig. 12. Total energy of CIM hardware for single image inference with different ADC resolution (1b: Modified IS, 2b: Q-Train (s=opt), 4b: Quant w/o retraining (s=opt), 5b: Quant w/o retraining (s=1)). Memory arrays with fan-in constraint 256, 128, and 64 are adopted.

References

[1] Y. Cai, et al., "Low bit-width convolutional neural network on rram," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019.
[2] Z. Jiang, et al., "C3SRAM: In-Memory-Computing SRAM Macro Based on Capacitive-Coupling Computing," IEEE Solid-State Circuits Letters, vol. 2, pp.131–134, 2019.
[3] Q. Wang, et al., "Neuromorphic processors with memristive synapses: Synaptic interface and architectural exploration," ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 12, no. 4, 2016.
[4] M. Courbariaux, et al., "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," arXiv preprint arXiv:1602.02830, 2016.
[5] X. Sun, et al., "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018.
[6] S. Yin, et al., "High-Throughput In-Memory Computing for Binary Deep Neural Networks With Monolithically Integrated RRAM and 90-nm CMOS," IEEE Transactions on Electron Devices, 2020.
[7] S. Yin, et al., "Monolithically integrated RRAM-and CMOS-based in-memory computing optimizations for efficient deep learning," IEEE Micro, vol. 39, no. 6, 2019.
[8] Y. Kim, et al., "Neural Network-Hardware Co-design for Scalable RRAM-based BNN Accelerators," arXiv preprint arXiv:1811.02187, 2018.
[9] T. Tang, et al., "Binary convolutional neural network on RRAM," Asia and South Pacific Design Automation Conference (ASP-DAC), 2017.
[10] J. Kim, et al., "Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array," IEEE Symposium on VLSI Circuits, 2019.
[11] A. Mishra, et al., "WRPN: wide reduced-precision networks," arXiv preprint arXiv:1709.01134, 2017.
[12] A. Bulat, et al., "XNOR-Net++: Improved binary neural networks," British Machine Vision Conference (BMVC), 2019.
[13] Z. Liu, et al., "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," European conference on computer vision (ECCV), 2018.
[14] K. He, et al., "Identity Mappings in Deep Residual Networks," European conference on computer vision (ECCV), 2016.
[15] S. Ioffe, et al., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," International Conference on Machine Learning (ICML), 2015.
[16] H. Kim, et al., "In-memory batch-normalization for resistive memory based binary neural network hardware," Asia and South Pacific Design Automation Conference (ASP-DAC), 2019.
[17] E. Park, et al., "Precision highway for ultra low-precision quantization," arXiv preprint arXiv:1812.09818, 2018.
[18] P.-Y. Chen, et al., "NeuroSim+: An Integrated Device-to-Algorithm Framework for Benchmarking Synaptic Devices and Array Architectures," IEEE International Electron Devices Meeting (IEDM), 2017.