Algorithm/Hardware Co-Design for In-Memory Neural Network Computing with Minimal Peripheral Circuit Overhead

Hyungjun Kim, Yulhwa Kim, Sungju Ryu, Jae-Joon Kim
Pohang University of Science and Technology, Pohang, Korea
{hyungjun.kim, yulhwa.kim, sungju.ryu, jaejoon}@postech.ac.kr

***Abstract*—We propose an in-memory neural network accelerator architecture called MOSAIC which uses minimal form of peripheral circuits; 1-bit word line driver to replace DAC and 1-bit sense amplifier to replace ADC. To map multi-bit neural networks on MOSAIC architecture which has 1-bit precision peripheral circuits, we also propose a bit-splitting method to approximate the original network by separating each bit path of the multi-bit network so that each bit path can propagate independently throughout the network. Thanks to the minimal form of peripheral circuits, MOSAIC can achieve an order of magnitude higher energy and area efficiency than previous in-memory neural network accelerators.**

I. Introduction

Recently, in-memory neural network accelerators have been attracting attentions for energy-efficient Deep Neural Network (DNN) computing [1], [2]. However, previous in-memory DNN accelerators suffer from significant overhead of peripheral circuits [2], [3]. Since typical in-memory DNN hardware is based on analog computing, analog-to-digital (or digital-to-analog) conversion is required near the memory array. It is known that such peripheral circuits typically dominate the power and area consumption of the entire system [2], [3]. In addition, multiple arrays are often needed to complete a weighted sum computation for an output neuron due to the limited array size. This requires higher resolution peripheral circuits to handle high-precision partial sum generation, thereby aggravating the overhead of peripheral circuits.

Meanwhile, low-precision DNNs have been actively studied to reduce computation overhead without too much accuracy degradation [4]–[7]. Both weights and activations can be quantized to have reduced precision so that parameter size and computational cost of a DNN model can be significantly reduced. DNN quantization also helps to reduce peripheral circuit overhead of in-memory DNN accelerators. In particular, binary neural networks (BNNs) [4] which use 1-bit for both weight and activation can minimize the complexity of peripheral circuits. However, BNNs typically suffer from considerable accuracy loss. Therefore, it would be desirable if multi-bit low-precision neural networks can be computed with in-memory DNN accelerators which conserve the benefits of 1-bit peripheral circuits in BNN accelerators.

In this paper, we propose an in-memory neural network accelerator architecture, MOSAIC, that can compute multi-bit networks with minimum peripheral circuits. Unlike previous designs, MOSAIC utilizes 1-bit input drivers and 1-bit read-out circuits only. Therefore, MOSAIC computes a network in a bitwise manner. In order to compute general networks on MOSAIC, we propose to split multi-bit networks and train them to achieve the accuracy value close to original one. We compare the MOSAIC architecture with other in-memory neural network accelerator designs using image classification benchmarks and show that MOSAIC can achieve the higher energy efficiency and the smaller area. The main contributions of this paper are as follows.

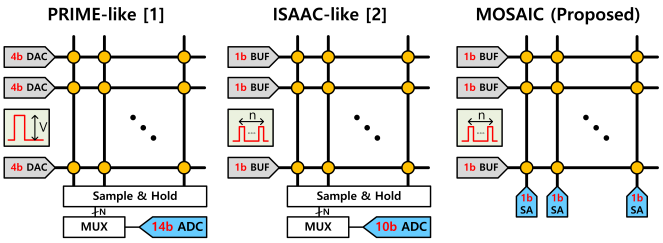

Fig. 1. In-memory computation schemes with different peripheral circuits. DAC or buffer (BUF) can be used as input driver. ADC or sense amplifier (SA) can be used as read-out circuit. Yellow circles at crosspoints represent memory cells.

1) We introduce the BitSplit-Net in which multi-bit activations are approximated by separate bit paths using bit-splitting method.
2) We present an in-memory computing architecture (MOSAIC) on which the BitSplit-Net can be mapped with minimal peripheral circuit overhead.
3) We evaluate the hardware efficiency of MOSAIC and previous architectures and show that MOSAIC can achieve better energy efficiency and smaller area.

II. Preliminaries

A. Categorization of in-memory DNN hardware depending on peripheral circuit style

Previous in-memory neural network accelerators can be categorized into a few groups depending on peripheral circuit styles [1], [2] (Fig. 1). The first type of in-memory neural network accelerators use high resolution Digital-to-Analog converters (DACs) and Analog-to-Digital converters (ADCs) as their peripheral circuits. Different input values are mapped to different voltage amplitudes by DACs and high-precision (i.e. 14-bit) ADCs are used as read-out circuits. Since both of input voltages and conductance values of memory cells have multi-bit resolutions, this type of architecture requires the highest resolution for ADCs. Since PRIME [1] is based on this scheme, let us call this type as the PRIME-like scheme for convenience.

On the other hand, ISAAC-like schemes compute each bit of input values at each time step [2]. Hence, ISAAC-like schemes need to use multiple time steps to represent multi-bit input values instead of using DACs. Each bit of an input value is provided by a single bit buffer (BUF) which is much simpler than a DAC. While ISAAC-like schemes can utilize ADCs with lower resolution than PRIME-like ones due to the lower input precision, ADCs need to run multiple times to deal with multi-bit inputs.

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

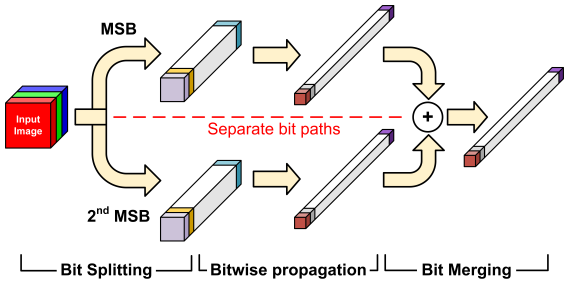

Fig. 2. An example of BitSplit-Net where 2-bit activations are split into two independent bits. Each 3D feature map is composed of binary pixel data.

Fig. 3. An image from ImageNet dataset in (a) 8-bit, (b) 1-bit (MSB), and (c) 1-bit (2nd MSB) format.

III. Bit-splitting methodology

In this section, we propose the bit-splitting methodology which approximates the original network and achieves similar accuracy. The methodology enables bitwise computations of multi-bit DNN and provides a foundation for mapping neural networks on MOSAIC architecture which relies on 1-bit input/output peripheral circuits.

A. Overall network configuration

Fig. 2 shows an example of the BitSplit-Net with 2-bit precision. At the beginning of the network, the feature map data are split into multiple paths. For example, when the network uses 2-bit precision for activation, the most significant bits (MSBs) of the feature map follow the first bit path while the second most significant bits (2nd MSBs) of the feature map follow the second bit path. Then, each bit of feature map data propagates throughout the network separately until the end of the network. While each bit path does not interact with each other, all the bit paths share the same weight parameters. At the end of the network, a bit merging layer merges the separated bits into one feature map and this feature map is used for the classifier.

Fig. 3 shows an image sampled from ImageNet dataset [8]. Each image is the result of quantization to (a) 8-bit, (b) 1-bit (MSB), and (c) 1-bit (2nd MSB). MSBs in the image represent the boundary between the dog and the background (Fig. 3b). On the other hand, 2nd MSBs in the image show more detailed information such as the eyes of the dog (Fig. 3c). This observation supports that training with each bit of activations separately and then combining the results may generate similar accuracy to the conventional case.

B. Layers in BitSplit-Net

In this section, we discuss the considerations required for forward and backward propagation in BitSplit-Net and provide details about each layer. For each layer, X/Y represent the input/output of the layer. Other layers that are not described below are implemented in the same way as done in conventional networks.

Bit splitting. The bit splitting layer is the first layer in which BitSplit-Net starts to differ from conventional networks. In this layer, high-precision inputs are split into multiple paths by different activation functions. Fig. 4 shows how the bit splitting layer mimics a conventional ReLU-based quantization layer. Fig. 4(a) shows a ReLU-based 2-bit quantization function when the input range is limited from 0 to 1. The 2-bit quantization produces 4 different output levels; {00, 01, 10, 11} indicate $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$, respectively. In this case, the first bit has a weight of $\frac{2}{3}$ and the second bit has weight of $\frac{1}{3}$. Let us call this, the bit-weight factor, β. We designed each activation function in the bit splitting layer based on this quantization scheme and Fig. 4(b) and (c) show two activation functions. Without loss of generality, one can design a k-bit splitting layer following the Algorithm 1. As described in the Algorithm 1, we constrained the activations to the range of [0,1]. Then, the binary code for each bit is determined and each of them is scaled by β_i. The output of the bit splitting layer will be $\{Y_i\}_{i=1}^k = \{\beta_i Y_i^B\}_{i=1}^k$ where $Y_i^B \in \{0, 1\}$.

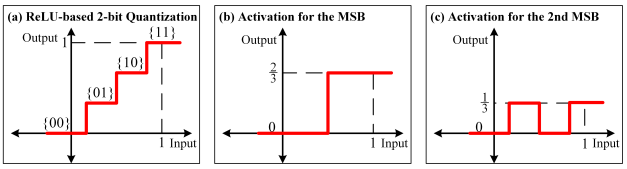

Fig. 4. (a) ReLU-based 2-bit quantization function used in conventional multi-bit networks. Activation function for the first (b) and the second (c) bit in (a). The combined result from (b) and (c) is identical to the result from (a).

Algorithm 1 Bit splitting algorithm

In: High-precision input $\mathbf{x} \in \mathbb{R}^n$, number of bits k
Out: k binary activations $\{\mathbf{y}_i\}_{i=1}^k$
Initialization: $\lambda_1 \leftarrow 2^k - 1$, $\lambda_2 \leftarrow 0$, $\beta_i \leftarrow 0$
$\mathbf{x} \leftarrow \text{ReLU1}(\mathbf{x})$ *equals to clamp(0,1)
for $i = 1$ **to** k **do**
  $\lambda_2 \leftarrow 2^{k-i}$, $\beta_i \leftarrow \frac{\lambda_2}{\lambda_1}$
  $\mathbf{y}_i \leftarrow \text{Modulo}\Big(\text{Floor}\big(\frac{1}{\lambda_2}\text{Round}(\lambda_1 \mathbf{x})\big), 2\Big)$
  $\mathbf{y}_i \leftarrow \beta_i \mathbf{y}_i$
end for
Return: $\{\mathbf{y}_i\}_{i=1}^k$

Gradients for each activation functions of bit splitting layer can be derived using straight-through-estimator (STE) enabling back-propagation [4]. Then, gradient values from each bit are accumulated as described in (1).

$$\frac{\partial \mathbf{C}}{\partial \mathbf{X}} = \sum_{i=1}^{k} \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i} \frac{\partial \mathbf{Y}_i}{\partial \mathbf{X}} \overset{\text{STE}}{=} \sum_{i=1}^{k} \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i} \beta_i \odot \mathbb{I}_{0 \le X \le 1} \tag{1}$$

Here, $\mathbf{C}$, $\odot$, and $\mathbb{I}$ denote for the cost function, element-wise multiplication and indicator function, respectively.

Bitwise convolution. After passing through the bit splitting layer, feature maps from each bit are convoluted separately, but with same weight parameters. Let us assume $\{\mathbf{X}_i\}_{i=1}^k \in \{0, \beta_i\}^{w \times h \times c_{in}}$ as k binary inputs to the convolution layer and $\mathbf{W}$ as weight matrix. Since each bit of activation propagates independently, we do not need to use conventional multi-bit multiplication for convolutions. Instead, we utilize convolution with bitwise input feature map. The forward propagation of the bitwise convolution is given by:

$$\mathbf{Y} = [\mathbf{Y}_1, ..., \mathbf{Y}_k], \quad \mathbf{Y}_i = Conv(\mathbf{X}_i, \mathbf{W}) \tag{2}$$

Since each bit path shares the same weight parameters, back-propagation is given by:

$$\frac{\partial \mathbf{C}}{\partial \mathbf{W}} = \sum_{i=1}^{k} \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i} \frac{\partial \mathbf{Y}_i}{\partial \mathbf{W}} = \sum_{i=1}^{k} \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i} \mathbf{X}_i$$

$$\frac{\partial \mathbf{C}}{\partial \mathbf{X}_i} = \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i} \frac{\partial \mathbf{Y}_i}{\partial \mathbf{X}_i} = \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i} \mathbf{W} \tag{3}$$

Note that fully-connect layer can be considered as a special case of convolution layer, so same approach can be applied.

Binary activation function. Except the first activation layer which is the bit splitting layer, all the other activation layers use the thresholding function as an activation function. While the threshold is fixed at 0.5, its output value varies depending on the bit position. The bit-weight factor, β, is again used to determine the output values of each thresholding function. For example, when k=2, the feature map of MSB path can have value of 0 or $\frac{2}{3}$ and that of 2nd MSB path can have value of 0 or $\frac{1}{3}$. This output value can be found based on the total number of bits (k) and which bit (i) the function is looking at (see Algorithm 1). Then, the binary activation is given by:

$$Y_i = \begin{cases} \beta_i & \text{if } x \geq 0.5 \\ 0 & \text{if } x < 0.5 \end{cases}$$

For back-propagation, the STE method was used similar to the case in the bit splitting layer as follows:

$$\frac{\partial \mathbf{C}}{\partial \mathbf{X}_i} = \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i}\frac{\partial \mathbf{Y}_i}{\partial \mathbf{X}_i} \overset{\text{STE}}{=} \frac{\partial \mathbf{C}}{\partial \mathbf{Y}_i}\beta_i \odot \mathbb{I}_{0\leq X\leq 1} \tag{4}$$

Bit merging. At the end of the network, the bit merging layer accumulates the results from different bit paths to produce the last feature map for classification. The result from each bit path has a different significance; for example, the result from the MSB path may be twice as important as the result from the 2nd MSB path. Note that this has been already considered by bit-weight factor, β, in the binary activation layer. The bit-weight factors can be simply computed, or can be set to learnable parameters so that optimal values can be found during training. Therefore, the output of the bit merging layer can be seen as the simple weighted sum of the results from different bit paths.

$$\text{Forward:} \quad \mathbf{Y} = \sum_{i=1}^{k} \beta_i \mathbf{X}_i^B$$

$$\text{Backward:} \quad \frac{\partial \mathbf{C}}{\partial \mathbf{X}_i^B} = \frac{\partial \mathbf{C}}{\partial \mathbf{Y}}\frac{\partial \mathbf{Y}}{\partial \mathbf{X}_i^B} = \frac{\partial \mathbf{C}}{\partial \mathbf{Y}}\beta_i \tag{5}$$

Experimental results showed that the proposed BitSplit-Net achieved accuracy comparable to conventional multi-bit networks. Detailed results are given in section V-A.

Meanwhile, one of the most important benefits in the BitSplit-Net is that the hardware-friendly characteristics of BNN (e.g. binary activation function) can be extended to the implementation of multi-bit neural network hardware.

In next section, we propose a new in-memory DNN computing architecture called MOSAIC which utilizes 1-bit precision peripheral circuits based on the bit-splitting methodology (Fig. 1 right).

IV. MOSAIC ARCHITECTURE

A. BitSplit-Net with in-memory computing

We first show how the BitSplit-Net scheme works with in-memory neural network accelerators. Among previous in-memory neural network accelerator designs, ISAAC-like schemes use 1-bit buffer as input driver while others use multi-bit DACs because ISAAC-like schemes can compute each bit of input values in different times steps. Fig. 5(a) shows an example of memory array operation for conventional multi-bit network in an ISAAC-like architecture. In the first cycle, an input block labeled as ① is provided to the memory array and a multiplication result which is also labeled as ① is produced. The output from the first input block should be converted to high precision intermediate data and intermediate data from each cycle is accumulated in the output buffer until the accumulated

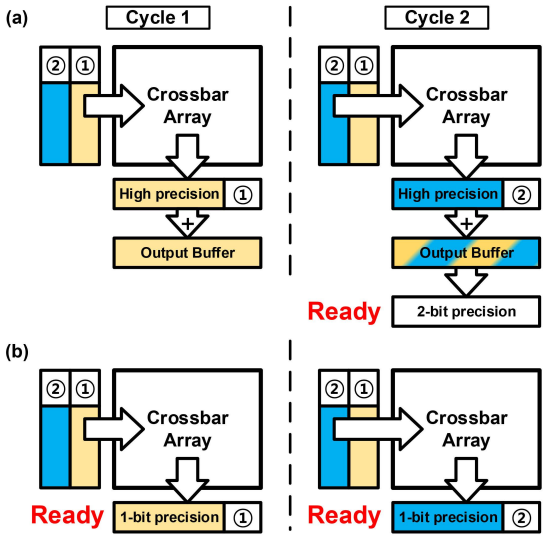

Fig. 5. Computing a multi-bit network on (a) ISAAC-like architecture and (b) MOSAIC architecture. In MOSAIC, an output from each cycle can be used by the next layer immediately.

data becomes a complete weighted sum value. For this reason, high resolution ADCs are required.

Such an ADC-related overhead is significantly reduced in the proposed MOSAIC architecture coupled with the BitSplit-Net scheme. First, different bit paths in BitSplit-Net can reuse the single memory array just like the conventional cases since different bit paths share the same weight parameters. Second, since each bit propagates independently throughout the network after the network is split at the beginning, output results from each cycle do not need to be accumulated in MOSAIC. Fig. 5(b) describes a memory array operation in the MOSAIC architecture. The output from the first input block can be directly activated and quantized to 1-bit because the network is modified to propagate binary feature map data with binary activation function. Therefore, single bit conventional sense amplifier (SA) can replace high resolution ADCs in MOSAIC. Note that using 1-bit buffer to drive word line and 1-bit SA to read out bit line is one of the simplest forms that memory array can have for peripheral circuits.

B. Pipelined bitwise propagation in MOSAIC

In MOSAIC architecture, each bit of a network propagates separately throughout the network so that different input bits can be computed in different time steps. Consequently, computations for each bit path can be pipelined (Fig. 6). Suppose that the target network has N layers and k bit paths and let us define an array operation cycle as the time which the array needs to finish a vector-matrix multiplication. In ISAAC-like architectures, it takes $N \times k$ array operation cycles to finish computing the whole network. However, since different bit paths can be pipelined independently in MOSAIC, it takes $N+k-1$ cycles only. For example, computing ResNet-34 with 4-bit activation requires 136 and 37 array operation cycles in ISAAC and MOSAIC, respectively. Furthermore, it is expected that an array operation cycle time of MOSAIC is expected to be smaller than that of ISAAC-like design due to the simpler SA read-out structure compared to high-resolution ADC. Also note that throughput of ISAAC-like design becomes similar to that of MOSAIC in batch processing cases, but for single image processing case which is common in many inference scenarios, MOSAIC achieves much smaller latency.

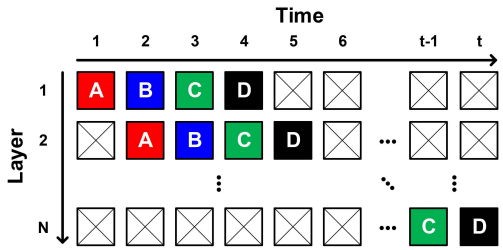

Fig. 6. Pipelined computation on MOSAIC. Different character in each block denotes different bit path of the network.

TABLE I
CLASSIFICATION ACCURACY OF BITSPLIT-NET FOR LENET-5 ON MNIST AND VGG-9 ON CIFAR-10. BINARY WEIGHT WAS USED FOR ALL CASES.

Network	LeNet-5		VGG-9	
	Accuracy	gap	Accuracy	gap
Baseline	99.32	-	90.36	-
k = 1	98.79	**0.53**	87.93	**2.43**
k = 2	99.17	**0.15**	89.50	**0.86**
k = 3	99.19	**0.13**	89.76	**0.60**
k = 4	99.22	**0.10**	89.99	**0.37**

In addition, MOSAIC has as good bit-scalability as ISAAC. Other architectures using DACs and ADCs (PRIME-like) can only compute networks that use limited bit precision for activation values. For example, if an architecture has 2-bit DACs, it is hard to compute networks which use higher than 2-bit for activation values. PRIME-like architectures can handle this problem by time-encoding strategy used in ISAAC, but it comes with additional overhead. However, networks with any bit precision can be computed on MOSAIC without burden since MOSAIC handles each bit path independently.

V. EXPERIMENTAL RESULTS

A. BitSplit-Net training results

We applied the BitSplit scheme to various image classification datasets and demonstrated that BitSplit-Net produces comparable classification accuracy to that of conventional multi-bit networks. We trained multiple networks including LeNet-5 [9] for MNIST dataset, VGG-9 [10] for CIFAR-10 dataset, and ResNet-18 [11] for ImageNet dataset. Large number of bitwise activation functions act as strong regularizer, so we lowered the value of weight decay.

Since our target is low precision multi-bit neural network which can be efficiently computed on in-memory computing architecture, we do not pursue higher than 4-bit cases in this work. We quantized weight following the methods described in [5].

Results on small datasets. We first evaluated our BitSplit scheme on small datasets such as MNIST and CIFAR-10, focusing on bit-scalability. While maintaining the weight precision to binary, we observed that the classification accuracy was dependent upon the number of the activation bits. Table I clearly shows that the classification accuracy improves as the precision of activation (k) increases. When $k = 4$, BitSplit-Net achieved an accuracy loss less than 0.5% compared to the full-precision baseline.

Results on ImageNet dataset. We also evaluated the BitSplit version of ResNet-18 on an ImageNet dataset which is a larger dataset than MNIST or CIFAR-10. We trained the BitSplit version of ResNet-18 with different bit configurations and compare the classification accuracy with other low-precision networks. Table II shows the classification accuracy and accuracy gap to the full-precision baseline for ResNet-18. We compared BitSplit-Net with other quantized networks where low-precision quantization is used. BitSplit versions of ResNet-18 were successfully trained to have similar accuracy level ($< \pm 0.5\%$) compared to other existing networks. Detailed data for AlexNet are skipped due to page limit but they showed similar trend as ResNet-18 results.

TABLE II
CLASSIFICATION ACCURACY OF RESNET-18 IN VARIOUS SCHEMES.

Network	Precision		Accuracy			
	A	W	Top-1	Top-5	Gap-1	Gap-5
Baseline	FP	FP	69.3	89.2	-	-
BNN	1	1	42.2	67.1	27.1	22.1
XNOR	1	1	51.2	73.2	18.1	16.0
HWGQ	2	1	56.1	79.7	13.2	9.5
ABC-Net	1	1	42.7	67.6	26.6	21.6
	3	3	61.0	83.2	8.3	6.0
BitSplit	2	1	56.8	79.7	12.5	9.5
	2	2	58.4	80.6	10.9	8.6
	2	3	59.0	81.3	10.3	7.9
	3	2	59.2	81.4	10.1	7.8
	2	4	60.6	82.5	8.7	6.7
	3	3	61.2	82.8	8.1	6.4

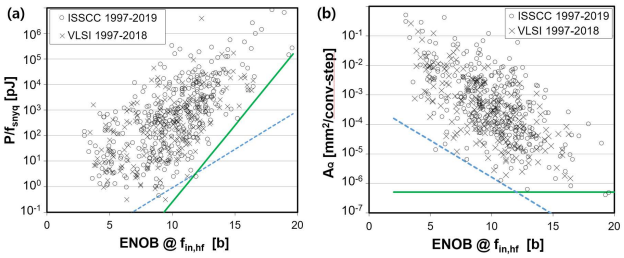

Fig. 7. Energy consumption and area per conversion-step of ADC designs from Murmann's ADC survey (as of April 2019) [12]. (Dashed line: minimum bound of ADCs with ENOB lower than 12-bit, Solid line: minimum bound of ADCs with ENOB greater than 12-bit)

B. Hardware evaluation setups

In this section, we introduce background rules and setups used for the hardware evaluation. We derived power and area of each element based on technology used, operating frequency, and bit resolution. For power and area of a memory array, we adopted the data from ISAAC [2].

Energy consumption of ADC. We first introduce energy model we used for ADCs with different resolutions. To consider realistic variations and current state-of-the-art ADC designs, we adopted the data from Murmann's ADC survey (Fig. 7) [12]. In Fig.7(a), more than 500 ADC designs published at top circuit conferences are plotted as a function of Effective Number of Bits (ENOB) and energy consumption (power per sampling frequency). According to [12], the minimum energy consumption of ADC follows two different metrics depending on whether the ENOB of ADC is greater or less than 12-bit. Based on the data, the minimum energy consumption of ADC for each bit resolution can be modeled as

$$E_{ADC} \geq \begin{cases} 0.88 \times 2^{ENOB} \text{ fJ} & \text{if } ENOB < 12 \\ 0.5 \times 10^{0.1(6.02 \times ENOB - 33.66)} \text{ fJ} & \text{if } ENOB \geq 12 \end{cases} \quad (6)$$

Area overhead of ADC. For area estimation, we applied an empirical methodology adopted from [13] to the Murmann's ADC

TABLE III
POWER AND AREA OF EACH COMPONENT WHEN SCALED TO 32NM TECHNOLOGY.

Type	Size	Power	Area (mm^2)	Frequency (Hz)
Array [2]	128^2	0.3 mW	2.5×10^{-5}	10 M
SA	1-bit	23 nW	4.58×10^{-6}	10 M
DAC [15]	10-bit	1 mW	0.0019	30 M
Buffer	1-bit	92 nW	0.91×10^{-6}	10 M

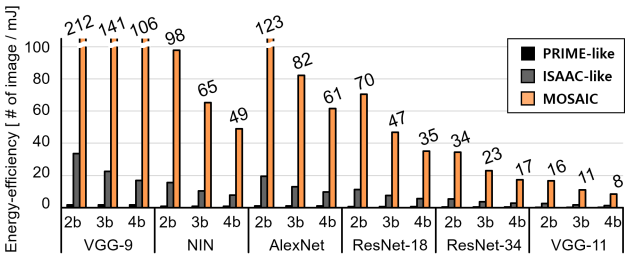

Fig. 8. Energy efficiency of different architectures on various benchmark networks. Array size is fixed to 128x128. "N"b represents the network uses N-bit activation and N-bit weight.

survey data to consider state-of-art designs. Fig. 7(b) shows minimum active area per conversion-step (A_Q) vs. ENOB trend. Then, the area of ADC (A) can be modeled as follows,

$$A(= A_Q \times 2^{ENOB}) \geq \begin{cases} 5\times10^{-7}\times 2^{ENOB}\ \text{mm}^2 & \text{if } ENOB \geq 12 \\ 10^{-0.25\times ENOB-3.3}\times 2^{ENOB}\ \text{mm}^2 & \text{if } ENOB < 12 \end{cases} \quad (7)$$

Power and area of sense amplifier. In case of 1-bit precision, we drew layout of the sense amplifier and measured total area. Power consumption was measured using SPICE simulation for the extracted netlist from the layout. Area and power consumption of 1-bit SA are summarized in table III.

Power and area of input drivers. For power and area of DACs, we used the scaling rule introduced in [14], which was also used in ISAAC [2].

$$P_{DAC} \propto \frac{2^N-1}{N+1}\times f, \quad A_{DAC} \propto \frac{2^N-1}{N+1} \quad (8)$$

We chose [15] as a reference design and scaled the power and area following (8).

Since 1-bit DAC can be replaced by 1-bit buffer, we also drew the layout of 1-bit buffer for area and power measurement. Table III summarizes the power, area and operating frequency of each component except ADC.

Benchmark Networks. To evaluate the MOSAIC architecture, we used two image classification datasets. For Cifar-10 dataset, we used a VGG-9 network. The network consists of 6 convolution layers and 3 dense (fully-connected) layers. For ImageNet dataset, we used 5 different networks; Network-In-Network (NIN) [16], AlexNet [17], ResNet-18 & ResNet-34 [11], and VGG-11 [10]. Since the parameter size and computational cost of first and last layers are relatively small, they are not quantized in general. Therefore, we left those layers out when evaluating energy and area.

C. Energy efficiency

In this section, we evaluate the energy efficiency of 3 different architectures; PRIME-like, ISAAC-like and MOSAIC. We focused on analyzing the energy consumption of memory array, input drivers, and read-out circuits. 3 different architectures were set to target up to 4-bit precision networks for fair comparison. The throughput of all the architectures was set equal and all designs were scaled to 32nm technology. Due to the high overhead of ADCs, single ADC was shared for 128 columns in PRIME-like and ISAAC-like architectures. Instead, sampling frequency of the shared ADCs was set 128 times higher than array operating frequency as suggested in [2].

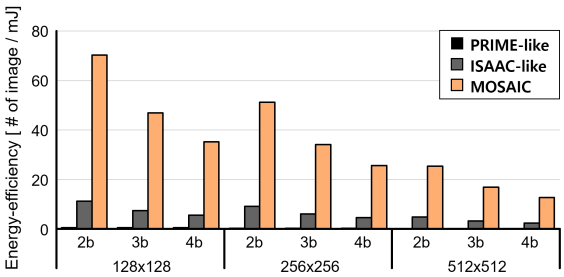

Fig. 9. Dependence of energy efficiency on array size. Note that energy efficiency of the PRIME-like architecture is much smaller than that of other designs so that the corresponding data value is not clearly seen in the graph.

Fig. 8 shows energy efficiency of 3 different architectures for 6 different benchmark networks. It is clearly shown that MOSAIC architecture achieves higher energy efficiency than other architectures for all benchmarks. Most of the differences come from the ADC overhead. For example, in ISAAC, 59.1% of total energy is consumed by ADCs when processing single image with ResNet-18, while MOSAIC uses SA that consumes 0.9% of total energy. In case of PRIME-like architecture, the energy efficiency is not improved with lower bit precisions since they use ADCs and DACs with the highest precision which they need to handle as discussed in section IV-B.

We also analyze the effect of array size on energy efficiency. Fig. 9 shows energy efficiency of 3 different architectures with different bit precision and different array size on ResNet-18 benchmark. First of all, MOSAIC achieved better energy efficiency than PRIME-like and ISAAC-like architectures for all configurations. Another observation from Fig. 9 is that energy efficiency of MOSAIC is reduced as the size of memory array increases. This is mainly due to the under-utilization of memory arrays. The number of output channels in neural networks for image classification varies from 64 to 1024. To guarantee best throughput of each architecture, we assumed that each array was solely dedicated for a single layer. Therefore, when the number of columns in an array is larger than the number of output channels that are assigned to the array, the memory cells in the unused columns are wasted, thereby resulting in loss of utilization ratio. On the other hand, the energy efficiency of other architectures are less affected by the under-utilization of memory array than MOSAIC. In case of MOSAIC architecture, memory array dominates energy consumption, and hence array under-utilization determines energy efficiency trends. However, peripheral circuits dominate the energy consumption in other architectures, thus effect of the array under-utilization is relatively small.

D. Area efficiency

For evaluation of area efficiency, we compared the area of memory arrays and the peripheral circuits required to compute ResNet-18. While previous architectures chose to share an ADC for multiple columns to reduce area overhead, MOSAIC uses a dedicated SA for each column since area of SA is much smaller than area of ADC. Despite having a dedicated SA for each column, MOSAIC

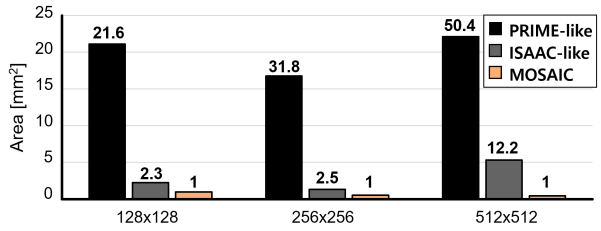

Fig. 10. Area of each architecture with different unit array sizes. Number on top of each bar represents normalized area to MOSAIC.

can achieve at least 21.6x and 2.3x smaller area than PRIME-like and ISAAC-like architectures, respectively, thanks to the absence of ADCs (Fig. 10). Regarding the dependence of the area on the array size, there are two conflicting factors. First, total number of peripheral circuits becomes smaller as the size of array increases, which tends to make total area smaller with the increased array size. On the other hand, under-utilization of the memory array results in the lower area efficiency when array size increases. When the trade-offs are considered together, it was observed that total area of MOSAIC structure tends to decrease as the array size increases (Fig. 10).

VI. Discussion

A. Addressing limited array size

In many cases, the number of rows in a memory array is smaller than the number of input neurons in a convolution/fully-connected layer. As a result, multiple memory arrays are typically used to map a large neural network layer on in-memory neural network accelerators [1], [2]. In such a case, results from each memory array are not a complete weighted sum, and hence high-precision ADCs may still be needed to compute partial sums even in BNN or BitSplit-Net computing hardware [18]. To address the limited array size issue, techniques such as partial sum quantization method using low-precision ADCs [18] or input-splitting method [19] have been proposed. Among them, input-split scheme enables 1-bit activation for each memory array in an in-memory BNN hardware by simply partitioning large layers to small partial layers so that the number of inputs of each partial layer becomes less than the number of rows of single memory array [19]. Hence, we applied the input-split method when there were limited array size cases in our analysis based on MOSAIC and BitSplit-Net.

B. Further accuracy improvement

Recently, several works proposed different techniques to improve accuracy of low-precision networks [6], [7]. Although we did not utilize such techniques in our results in section V-A, it is possible to apply the techniques to BitSplit-Net for better accuracy results. For example, we could achieve accuracy which was closer to full-precision results by doubling the number of channels of each layer in a network as proposed in [6]; the top-1 accuracy of BitSplit version of AlexNet and ResNet-18 was improved to 56.3% and 67.9% which have 0.8% and 1.4% gap to full-precision results, respectively. As low-precision network training techniques are being actively studied by many researchers, we plan to keep adopting latest techniques to further improve accuracy result of BitSplit-Net.

VII. Conclusion

We propose an algorithm (BitSplit-Net) / hardware (MOSAIC) co-design approach to enable bitwise computations for approximate multi-bit neural network in an in-memory computing style. Thanks to the bitwise computing characteristics, the proposed architecture can be regarded as an extended form of BNN hardware and inherits the in-memory computing friendly features of BNN while computing multi-bit neural network. We evaluate the energy efficiency and area of 3 different schemes including the proposed MOSAIC architecture. MOSAIC can achieve 90.4x and 6.3x higher energy efficiency compared to PRIME-like and ISAAC-like architectures on average, respectively. MOSAIC can also achieve 34.6x and 5.7x area saving compared to the same architectures.

Acknowledgment

This work was in part supported by the Technology Innovation Program (10067764) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) and Samsung Electronics Co., Ltd. This work was also in part supported by Ministry of Science and ICT (MSIT, Korea), under the ICT Consilience Creative program (IITP-2019-2011-1-99783) supervised by the Institute for Information & communications Technology Planning & Evaluation (IITP) and the Nano-Material Technology Development Program through the National Research Foundation of Korea (NRF) funded by the MSIT (NRF2016M3A7B4910249).

References

[1] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *International Symposium on Computer Architecture (ISCA)*, pp. 27–39, 2016.

[2] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *International Symposium on Computer Architecture (ISCA)*, pp. 14–26, 2016.

[3] B. Li, L. Xia, P. Gu, Y. Wang, and H. Yang, "Merging the interface: Power, area and accuracy co-optimization for rram crossbar-based mixed-signal computing system," in *Design Automation Conference (DAC)*, p. 13, ACM, 2015.

[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv:1602.02830*, 2016.

[5] S. Zhou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv:1606.06160*, 2016.

[6] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: wide reduced-precision networks," *ICLR*, 2018.

[7] J. Choi *et al.*, "Accurate and efficient 2-bit quantized neural networks," in *Proceedings of the 2nd SysML Conference*, 2019.

[8] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *CVPR*, pp. 248–255, IEEE, 2009.

[9] Y. LeCun *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, p. 276, 1995.

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

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, pp. 770–778, 2016.

[12] B. Murmann, "Adc performance survey 1997-2019." http://web.stanford.edu/~murmann/adcsurvey.html. Accessed: 2019-07-30.

[13] B. E. Jonsson, "Area efficiency of adc architectures," in *European Conference on Circuit Theory and Design (ECCTD)*, pp. 560–563, 2011.

[14] M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn, "Analysis of power consumption and linearity in capacitive digital-to-analog converters used in successive approximation adcs," *Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 8, pp. 1736–1748, 2011.

[15] M. Borremans, A. Van den Bosch, M. Steynaert, and W. Sansen, "A low power, 10-bit cmos d/a converter for high speed applications," in *Custom Integrated Circuits Conference*, pp. 157–160, IEEE, 2001.

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv:1312.4400*, 2013.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[18] X. Sun *et al.*, "Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1423–1428, IEEE, 2018.

[19] Y. Kim, H. Kim, D. Ahn, and J.-J. Kim, "Input-splitting of large neural networks for power-efficient accelerator with resistive crossbar memory array," in *Proceedings of the International Symposium on Low Power Electronics and Design*, p. 41, ACM, 2018.