Modeling and Optimization of SRAM-based In-Memory Computing Hardware Design

Jyotishman Saikia*, Shihui Yin*, Sai Kiran Cherupally*, Bo Zhang†, Jian Meng*, Mingoo Seok†, Jae-sun Seo*
*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA
†Department of Electrical Engineering, Columbia University, New York, NY, USA
Email: jsaikia@asu.edu, jaesun.seo@asu.edu

***Abstract*—In-memory computing (IMC) has been demonstrated as a promising technique to significantly improve energy-efficiency for deep neural network (DNN) hardware accelerators. However, designing one involves setting many design variables such as the number of parallel rows to assert, analog-to-digital converter (ADC) at the periphery of memory sub-array, activation/weight precisions of DNNs, etc., which affect energy-efficiency, DNN accuracy, and area. While individual IMC designs have been presented in the literature, they have not investigated this multi-dimensional design optimization. In this paper, to fill this knowledge gap, we present a SRAM-based IMC hardware modeling and optimization framework. A unified systematic study closely models IMC hardware, and investigates how a number of design variables and non-idealities (e.g. device mismatch and ADC quantization) affect the DNN accuracy of IMC design. To maintain high DNN accuracy for the IMC SRAM hardware, it is shown that the number of activated rows, ADC resolution, ADC quantization range, and different sources of variability/noise need to be carefully selected and co-optimized with an underlying DNN algorithm to implement.**

***Index Terms*—In-memory computing, deep neural networks, SRAM, modeling, optimization**

I. Introduction

Deep neural networks (DNNs) are prevalent in a range of real-world applications such as computer vision, speech recognition, natural language processing, etc. While state-of-the-art DNN algorithms continue to achieve higher accuracy with a smaller number of operations and parameters, the most compact DNN model for ImageNet dataset still requires >0.5 billion multiply-and-accumulate (MAC) operations and >2 million weights [1] for the inference of a single image.

As a means to effectively reduce the computation cost and memory storage of large DNNs, recent algorithm techniques successfully reduced the activation/weight precision to 2-bit/4-bit [2], [3], while maintaining on-par DNN inference accuracy for ImageNet compared to DNNs with floating-point precision. To efficiently implement such DNN algorithms, many digital DNN accelerators have been designed to support specialized processing dataflows that optimize data access through a memory hierarchy for MAC computations in a 2D systolic array [4]–[7]. However, the energy/power breakdown reported in [5]–[7] shows that memory access and data movement accounts for a dominant portion of the total on-chip energy/power consumption. This is because activation/weight data of DNNs need to be fetched from memory and communicated to a separate MAC array for a large number of iterations.

To address this limitation, in-memory computing (IMC) has emerged, where MAC computation is performed inside on-chip memory (e.g. SRAM) by turning on multiple/all rows of a memory array. The MAC result is represented by analog bitline voltage/current, and digitized by an analog-to-digital converter (ADC). This substantially reduces data transfer (compared to digital accelerators with separate MAC arrays) and increases parallelism (compared to conventional row-by-row access), largely improving the energy-efficiency of MAC operations.

In recent years, there have been a number of silicon demonstrations of IMC SRAM [8]–[13], where energy-efficiencies of up to hundreds of TOPS/W have been demonstrated. IMC designs can be broadly categorized into two approaches: (1) resistive IMC that implements bit-wise/multi-bit MAC operation by resistive pull-down/-up via transistors [8]–[10], and (2) capacitive IMC that implements MAC operation by capacitive coupling or charge sharing [11]–[13]. The key challenge of IMC is that analog computation exhibits less margin to variability or noise, which can degrade the DNN accuracy.

In IMC design, there are important design parameters to determine, in order to achieve high energy-efficiency as well as high DNN inference accuracy. These include the number of active rows to turn on, partial sum quantization scheme with respect to activation/weight precision of DNN model, ADC precision (i.e. the number of ADC levels), bitcell variation, and ADC offset. These design parameters govern the energy-efficiency, throughput, variability of in-memory computation, and DNN accuracy. Each of the IMC prototypes in the literature determined their own set of design parameters, but the rationale or criteria of how such design decisions were made are often not reported in detail. For IMC hardware design, there is a need to systematically study different types of variations including the resistive or capacitive pull-down/-up strength that impacts the read bitline (RBL) voltage, local mismatch, and ADC offset.

There have been a few prior works that partially investigated this aspect. In [13], it has been reported that activating more rows for IMC operation leads to worse signal-to-quantization-noise ratio (SQNR), but the relationship of SQNR and DNN accuracy was not investigated. In [14], the authors discussed the effect of variation on RBL voltage swing and the energy vs. accuracy tradeoff, but the analysis is limited with only four rows activated in a cycle and with simple machine learning algorithms (non-DNN) such as support vector machines.

In this paper, we present a Python-based modeling framework for IMC SRAM hardware, which evaluates the inference accuracy of an arbitrary DNN. We model and analyze the

978-3-9819263-5-4/DATE21/©2021 EDAA

effect of the core design parameters of IMC for DNN accuracy, including the number of activated rows, different partial sum quantization schemes, ADC precision, bitcell variation on RBL voltage, and ADC offset. For a given number of activated rows, ADC precision and quantization scheme, we model and add noise for IMC SRAM that represents bitcell variation and ADC offset, which results in a quantized and noisy partial sum value for each IMC SRAM array. This partial MAC result is accumulated for the final MAC value of a DNN layer and combined with digital simulations of non-MAC operations, towards evaluating the entire DNN accuracy. We present detailed analysis results for DNNs with 1-bit to 4-bit precision, for both CIFAR-10 and ImageNet datasets.

The main contributions of this work include:

- For SRAM-based IMC, we present a comprehensive simulation framework that integrates different core design parameters, including DNN weight/activation precision, the number of activated rows, partial sum quantization schemes, ADC precision, ADC offset, and RBL voltage variation due to bitcell or capacitor mismatch.
- We characterize the effect of different variations on the DNN accuracy for both CIFAR-10 and ImageNet datasets.
- We show the relationship of partial sum SQNR and network-level accuracy and report that a quantization range much wider than ± 3-σ distribution may be necessary for IMC, to maintain high DNN accuracy.

Code for this work is available at https://github.com/Jsaikia47/IMC-SRAM-Modeling. The remainder of this paper is organized as follows. Section II describes prior IMC SRAM works, and Section III presents the proposed IMC modeling framework. Section IV discusses the experimental results and Section V concludes the paper.

II. Prior In-Memory Computing Works

A. IMC Design with 6-T vs. (6+extra)-T Bitcell

Depending on the bitcell type and the degree of parallel computation, the IMC design can be largely divided into three structures, as shown in Fig. 1. Since the early IMC SRAM works, 6-T bitcell has been employed for IMC [14], [15] (Fig. 1(a)). High parallelism could be achieved by activating all/multiple rows, but one of the critical concerns is that IMC with 6-T bitcell is subject to read disturb issue, causing the wordline voltage to be driven with a low voltage (e.g. 0.4V or less). In other recent works [16], to eliminate read disturb, 6-T bitcell has been used, but a dedicated local compute engine was employed for a certain group of rows (Fig. 1(b)). However, for each group, the MAC computation is still row-by-row.

On the other hand, there are IMC works that add a couple of extra transistors on top of the 6-T bitcell (Fig. 1(c)), where all/multiple rows are turned on [8]–[13]. In this IMC structure, high parallelism is achieved without read disturb issue, by trading off the bitcell and array area. Since this IMC scheme has demonstrated the highest energy-efficiency in the literature by activating a high number of rows in a robust manner without read disturb, in this work, we focus our proposed IMC modeling framework using this (6+extra)-T bitcell scheme.

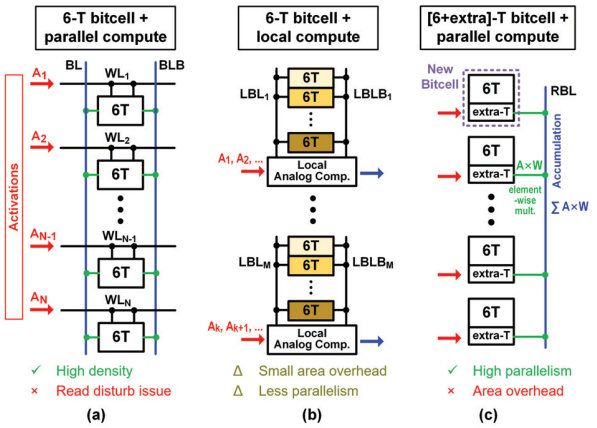

Fig. 1. Structure of different analog IMC SRAM schemes depending on bitcell design and parallel/local computation.

B. Resistive vs. Capacitive IMC Design

IMC works with the (6+extra)-T bitcell structure can be categorized into either resistive IMC scheme or capacitive IMC scheme. The resistive IMC scheme implements bit-wise/multi-bit MAC operation by resistive pull-down/-up via transistors [8]–[10], and the capacitive IMC scheme implements MAC operations by capacitive coupling or charge sharing [11]–[13]. While the resistive IMC scheme does not need additional capacitors, the pull-down/-up characteristics of the transistors across a wide voltage range cause the RBL transfer curve to exhibit a non-linear behavior for MAC value [9], [10].

On the other hand, the capacitive IMC scheme employs an additional capacitor per bitcell, to perform MAC computation by means of capacitive coupling or charge sharing. Due to the nature of capacitive charge transfer, the RBL transfer curve of capacitive IMC works shows a linear behavior, and the variability of capacitors is typically less than that of transistors in the advanced CMOS technology nodes. By employing a metal-oxide-metal (MOM) capacitor on top of the bitcell, additional front-end area need not be consumed. Aided by these advantages, in this work, we focus on the capacitive IMC scheme for the IMC modeling framework. Still, by properly modeling the non-linear behavior of CMOS transistors at a given voltage, our IMC modeling framework could be further extended to resistive IMC scheme as well.

Fig. 2 illustrates the two representative capacitive IMC works [11], [12]. The IMC SRAM bitcell in [11] performs XNOR operation between a binary input activation and a binary weight. In the first phase of operation, depending on the bitwise XNOR computation result, each bitcell charges or discharges the capacitor. In the second phase, all of the capacitors in a neuron tile will be connected and charge sharing of all capacitors effectively performs the binary MAC operation.

In C3SRAM [12], the bitwise multiplication is performed by the row periphery driving MWL (MWLB) from V_{RST} to VDD while MWLB (MWL) remains at V_{RST}. Depending on the stored weight, the voltage ramping via either of the two transistors connected to MWL/MWLB induces a displacement current through the series capacitor C_C. Since the other side of

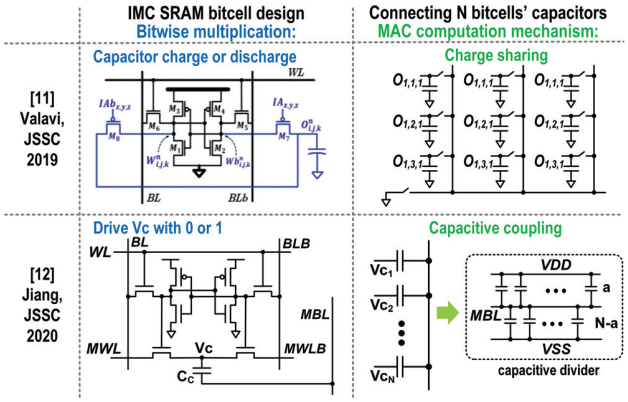

Fig. 2. Capacitive IMC design and operation. In [11], each bitcell's computation result is stored on a capacitor and subsequently charge shared. In [12], each bitcell's computation result forms a capacitive voltage divider and MBL voltage represents the MAC value. Adapted from [11] and [12].

C_C from all bitcells in the same column are connected to the same MBL signal, driving MWL/MWLB and the C_C results in capacitive coupling, which effectively performs the binary MAC computation as a capacitive divider (Fig. 2).

C. ADC Design and Quantization in Previous IMC Chips

Most of the IMC chip designs use either successive approximation register (SAR) ADC [8], [13] or flash ADC [9], [10], [12]. Flash ADC performs digitization at a high speed but requires many comparators, the number of which grow exponentially with the ADC precision. While SAR ADCs only use a single comparator iteratively leading to a small area, the speed is slower and additional capacitive banks are typically required for digital-to-analog conversion inside the ADC.

Some IMC chips seemingly use full range linear quantization [13], while some IMC designs experimented using non-linear quantization where fine-grain references are placed where there exists more data [9]. Other IMC designs employed linear quantization within a clipped range [9], [12], where the quantization is performed in a restricted range where most of the data exists. Exploiting the DNN data distribution, this clipped linear quantization can result in better DNN accuracy than full-range linear quantization for identical ADC precision, while also avoiding the complexity of non-linear quantization.

III. IMC Hardware Modeling Framework

In this section, we present the IMC modeling framework and how we incorporated different variation/noise sources such as ADC quantization, ADC offset, and capacitor mismatch.

A. Overall Python Framework Workflow

The proposed Python-based model focuses primarily on the implementation of the bit-by-bit computation aspect of IMC for convolution and fully-connected (FC) layers of DNNs. In this work, we use ResNet-18 for CIFAR-10 and ImageNet datasets. Other non-IMC operations, such as batch normalization, pooling, etc. are performed digitally with high precision, so that they will not affect the DNN accuracy.

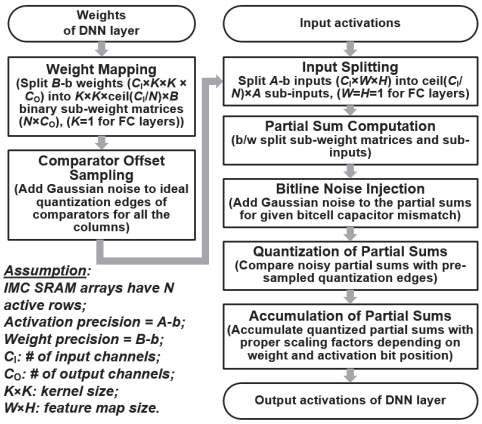

Fig. 3. High-level flow of the Python-based IMC modeling framework.

Fig. 3 shows the flow chart of the operations executed within a convolution or FC layer. First, we map the weights onto multiple IMC SRAM arrays by splitting the multi-bit weights into N-row binary sub-weight matrices. Second, we randomly sample the input-referred offsets according to a Gaussian distribution for the comparators in each column. Note that this ADC offset noise can generally model the non-ideality of both flash ADC and SAR ADC, where our ADC offset models the local mismatch/variation of the comparators (in both flash ADC and SAR ADC) or the capacitor DAC bank (in the SAR ADC).

We also split the multi-bit input into binary N-channel sub-inputs. Bit-by-bit convolution/FC computation between split sub-inputs and sub-weights are performed to obtain the partial sums. Gaussian noise is added to these ideal partial sums to model the RBL voltage noise caused by the bitcell capacitor mismatch and other thermal noise. Then, the noisy partial sum of each bit precision is quantized to obtain the quantized noisy output. The partial sums are added with proper binary-weighted coefficients depending on the corresponding bit positions of the sub-input and the sub-weight. Subsequently, the partial sums are accumulated towards the final sum, and finally, we obtain the output activation of the DNN layer.

B. Noise Modeling and Quantization Range Determination

The proposed IMC modeling framework includes implementations and models of ADC quantization, ADC offset, and bitcell/capacitance variation for RBL noise. Our IMC framework mainly models two types of variations: (1) RBL voltage variation, arising from the capacitance mismatch, and (2) ADC output variation, due to the quantization error and the imperfect ADC quantization edges. For a capacitive IMC SRAM array with N rows, the RBL voltage is formulated as:

$$V_{RBL} = \frac{\sum_{i=1}^{N} c_i \times y_i}{\sum_{i=1}^{N} c_i}, \tag{1}$$

where $y_1, \cdots, y_N$ represents the bitwise binary MAC computation results from N rows, which will be either VDD or 0. $c_1, \cdots, c_N$ represent the capacitor values of IMC bitcells in different rows, which are sampled from a Gaussian distribution with a certain standard variation over mean (σ/μ) value. Random

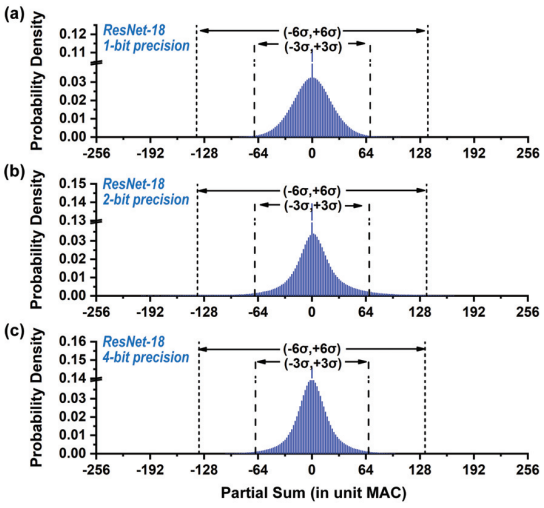

Fig. 4. Column-wise partial sum distributions for ResNet-18 for CIFAR-10 with different activation/weight precision of (a) 1-bit, (b) 2-bit, and (c) 4-bit.

combinations of c_i and y_i that result in the same ideal partial sum are generated. We then measure the standard deviation of V_{RBL} and use it to model the effect of bitcell capacitor mismatch on V_{RBL}. This noise is then converted into unit MAC value and applied to the ideal partial sums.

To model the ADC offset, the offset value is converted into unit MAC value. This is then applied to the quantization levels in the form of a Gaussian noise with a standard deviation of the chosen offset value (in unit MAC) in the modeling framework. To determine the quantization levels, the distribution of activation values over the entire DNN is observed, as shown in Fig. 4. The standard deviation of this distribution is represented as σ. We experiment with various integer multiple values of σ as the quantization range for the IMC partial sum, and we perform linear quantization within this clipped range [9].

C. Evaluation of Signal-to-Quantization-Noise Ratio

The DNN accuracy strongly depends on the signal-to-quantization-noise ratio (SQNR) of the partial sums generated by the IMC SRAM arrays. To evaluate the SQNR, we use the ideal output activation of a layer as the signal. The ideal output activation is the output obtained without adding any form of noise. Here, 'signal' is the ideal output activation of a DNN layer. Then we also gather the noisy output activation for the DNN layer, denoted as X. This gives us the noise of the DNN layer, where 'noise' is $(X-signal)$. With the power of the signal and the noise, we can obtain the SQNR as:

$$SQNR = E[signal^2]/E[noise^2]. \quad (2)$$

SQNR related experiments from our modeling framework will be discussed in Section IV.

IV. Experimental Results

A. Quantization Range

We first analyze the partial sum quantization range of IMC. Using the linear quantization scheme in a clipped range [9], we

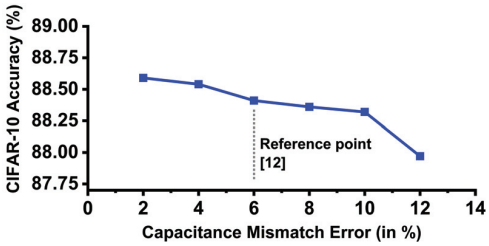

Fig. 5. The capacitance mismatch error is swept for binary ResNet-18 for CIFAR-10 at full quantization range. 6% capacitance mismatch error was reported for [12], and used as a reference point for other experiments.

experimented with various quantization range values. Fig. 4 shows partial sum distributions of ResNet-18 for CIFAR-10 with activation/weight precisions of 1-bit/1-bit, 2-bit/2-bit, and 4-bit/4-bit. ± 3-σ covers 99.73% of the entire data, and as seen in Fig. 4, this might seem sufficient to perform clipped linear quantization within this ± 3-σ range. However, as we will report in ensuing experiments, interestingly ± 3-σ is found to be an insufficient range and the optimal DNN accuracy is found at much wider quantization ranges for certain experiments, such as ± 8-σ range. The reason behind this is that,s while there is scarce data near the minimum and maximum point of the partial sum, if an error occurs, the magnitude of the error is large due to the partial sum magnitude, and this can contribute to noticeable DNN accuracy degradation. This is especially true for DNN and IMC structures with a high dynamic range, such as multi-bit precision and a high number of activated rows.

B. Capacitor Mismatch and the Number of Activated Rows

In the capacitive IMC scheme, the bitcell variation mostly comes from the capacitance mismatch, which essentially affects the RBL voltage. The capacitance mismatch is modeled as standard Gaussian noise, as described in Section III. When averaged over several iterations of Gaussian noise sampling, as the capacitance mismatch error increases, the DNN accuracy degrades in general (Fig. 5).

We experimented with how the number of activated rows in IMC SRAM array affects SQNR and network-level DNN accuracy. In Fig. 6, we consider a full range quantization sweep, 6% capacitance mismatch (measured value from [12]), and 5mV ADC offset (estimated as the upper range). As the number of SRAM rows increases, SQNR decreases, resulting in worse DNN accuracy. Energy consumption will be linearly dependent on the number of activated rows, e.g. IMC with 256 activated

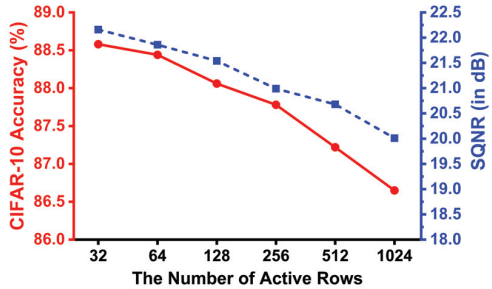

Fig. 6. SQNR and DNN accuracy for different numbers of activated rows.

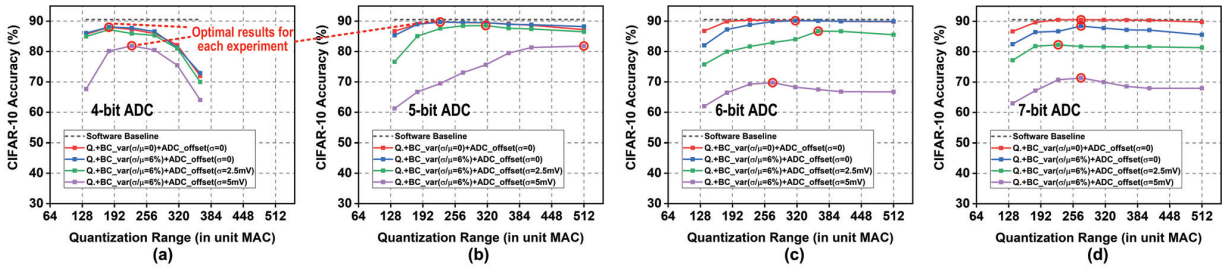

Fig. 7. Detailed simulation results with the proposed IMC modeling framework for ResNet-18 for CIFAR-10 with 2-bit/2-bit activation/weight precision. For ADC precision of (a) 4-bit, (b) 5-bit, (c) 6-bit, and (d) 7-bit, results for various quantization ranges, bitcell variation (BC_var), and ADC offset values are shown.

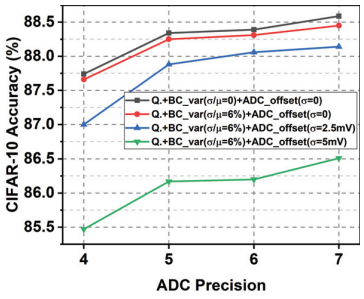

Fig. 8. The optimal results of 1-bit ResNet-18 for CIFAR-10 across different ADC precision, bitcell variation (BC_var), and ADC offset, where the optimal quantization range is swept and chosen for each experiment.

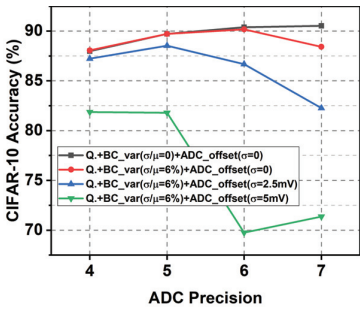

Fig. 9. The optimal results of 2-bit ResNet-18 for CIFAR-10 across different ADC precision, bitcell variation (BC_var), and ADC offset, where the optimal quantization range is swept and chosen for each experiment.

rows will be 4× more energy-efficient than IMC with 64 activated rows. In Fig. 6, activating more than 256 rows worsens DNN accuracy more steeply, hence we will focus on having 256 activated rows for the IMC SRAM in subsequent experiments. For more error-prone IMC schemes or DNN models, this can be a useful knob to regain the DNN accuracy.

C. Detailed Experiments of ResNet-18 for CIFAR-10

Putting these altogether in our IMC modeling framework, Fig. 7 shows the detailed simulation results for ResNet-18 for CIFAR-10 dataset with 2-bit/2-bit activation/weight precision. Across various quantization ranges, CIFAR-10 accuracy results for different ADC precision, bitcell variation (BC_var), and ADC offset values are shown.

Since ±3-σ quantization range represents ∼132 in unit MAC (Fig. 4(b)), the results of many experiments in Fig. 7 indicate that employing quantization ranges much higher than ±3-σ

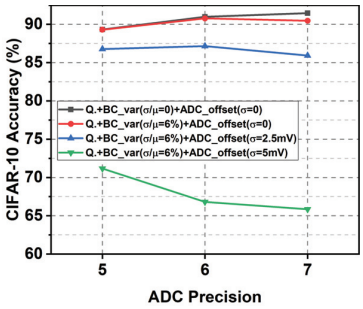

Fig. 10. The optimal results of 4-bit ResNet-18 for CIFAR-10 across different ADC precision, bitcell variation (BC_var), and ADC offset, where the optimal quantization range is swept and chosen for each experiment.

(e.g. ±6-σ, ±8-σ) results in high DNN accuracy. Also, Fig. 7(a) shows that 4-bit ADC is not sufficient, but 6-bit and 7-bit ADCs (Fig. 7(c) and Fig. 7(d)) do not necessarily maintain high accuracy either, especially when bitcell variation and ADC offset are both present. This is because such variation/noise interferes with the fine-grain quantization that high-precision ADCs require. For ADC offset of 2.5mV, 5-bit ADC shows the best result close to the software baseline, but when the variation is large, some amount of DNN accuracy loss is inevitable.

For each experiment in Fig. 7, the optimal quantization range that results in the highest DNN accuracy is marked with a red circle. Only compiling those optimal datapoints for each experiment results in Fig. 9 for ResNet-18 for CIFAR-10 with 2-bit precision. We have done the same experiments for ResNet-18 for CIFAR-10 with activation/weight precision values of 1-bit and 4-bit. The optimal results of 1-bit ResNet-18 and 4-bit ResNet-18 are shown in Fig. 8 and Fig. 10, respectively.

D. Detailed Experiments of ResNet-18 for ImageNet

In Fig. 11, the detailed simulation results are shown for ResNet-18 for ImageNet dataset with 2-bit/2-bit activation/weight precision. Across various quantization ranges, ImageNet accuracy results for different ADC precision, bitcell variation (BC_var), and ADC offset values are shown. Two aspects are notable for these ImageNet results, compared to CIFAR-10 results. First, higher ADC precision is required (e.g. 7-bit ADC) to maintain high DNN accuracy. Second, the DNN accuracy based on IMC hardware is much more sensitive to capacitor variation and ADC offset.

In Fig. 11, the highest DNN accuracy datapoints for each experiment is marked with a red circle. Fig. 12 shows only

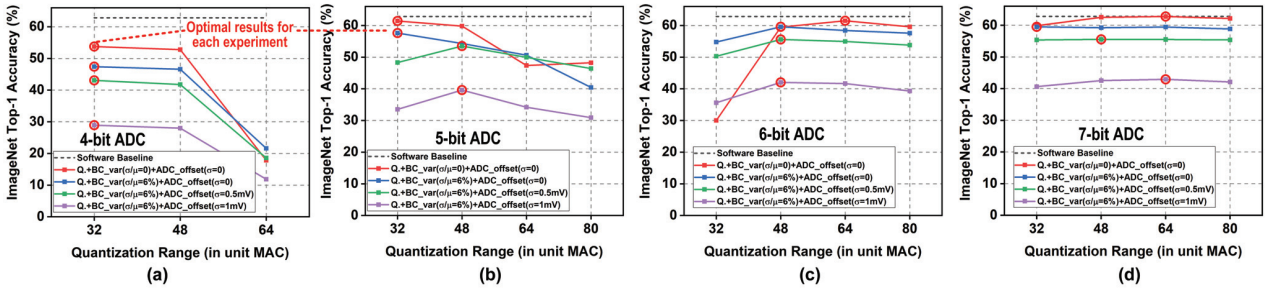

Fig. 11. Detailed simulation results with the proposed IMC modeling framework for ResNet-18 for ImageNet with 2-bit/2-bit activation/weight precision. For ADC precision of (a) 4-bit, (b) 5-bit, (c) 6-bit, and (d) 7-bit, results for various quantization ranges, bitcell variation (BC_var), and ADC offset values are shown.

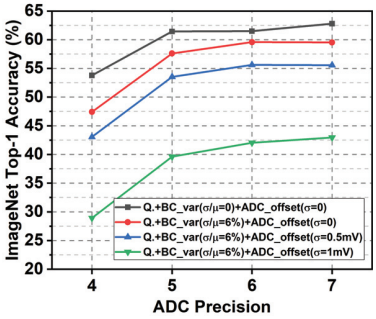

Fig. 12. The optimal results of 2-bit ResNet-18 for ImageNet across different ADC precision, bitcell variation (BC_var), and ADC offset, where the optimal quantization range is swept and chosen for each experiment.

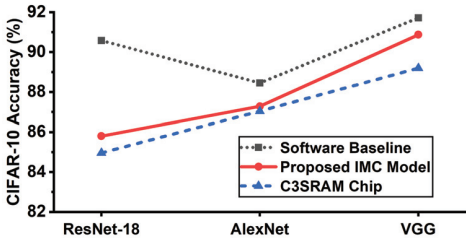

Fig. 13. Comparison of DNN accuracy based on proposed IMC model and C3SRAM [12] measurements.

those optimal results for 2-bit ResNet-18 for ImageNet dataset. With higher sensitivity to variation/noise, DNNs for ImageNet based on IMC will require much more rigorous capacitor/bitcell matching and ADC offset cancellation.

E. Validation of Model vs. IMC Chip Accuracy

To validate our proposed model, we compared the DNN accuracy projected by the proposed IMC model with the DNN accuracy based on C3SRAM chip measurements [12]. The C3SRAM chip based accuracy is evaluated based on measured error probability using ADC output distributions from 10,000 measured samples, as described in [12]. Across three different binary DNN models, Fig. 13 shows the DNN accuracies obtained by the software baseline, proposed model, and the C3SRAM chip measurements, and the proposed model closely predicts the accuracies by the C3SRAM chip.

V. Conclusion

This paper presented SRAM based IMC modeling framework and optimization. By properly modeling the bitcell variation and ADC offset, DNNs for CIFAR-10 as well as ImageNet datasets are characterized, showing the sensitivity to different variations, and core IMC design parameters such as the number of activated rows, ADC precision, and quantization range. The proposed IMC modeling framework has been open-sourced, and our model predicted DNN accuracy shows a close match to IMC chip measurement based accuracy. The evaluations and optimizations shown in this work will be useful towards efficiently implementing arbitrary DNN algorithms onto IMC hardware while maintaining high DNN accuracy.

Acknowledgment

This work was in part supported by Samsung Advanced Institute of Technology, NSF grant 1652866, and C-BRIC, one of six centers in JUMP, a SRC program sponsored by DARPA.

References

[1] L. Deng *et al.*, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proc. IEEE*, vol. 108, 2020.
[2] J. Choi *et al.*, "Accurate and Efficient 2-bit Quantized Neural Networks," in *Conference on Machine Learning and Systems (MLSys)*, 2019.
[3] S. K. Esser *et al.*, "Learned Step Size Quantization," in *ICLR*, 2020.
[4] B. Moons *et al.*, "ENVISION: A 0.26-to-10 TOPS/W Subword-Parallel Dynamic-Voltage-Accuracy-Frequency-Scalable Convolutional Neural Network Processor in 28nm FDSOI," in *IEEE ISSCC*, 2017.
[5] Y. Chen *et al.*, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE JSSC*, vol. 52, 2017.
[6] J. Sim *et al.*, "An Energy-Efficient Deep Convolutional Neural Network Inference Processor With Enhanced Output Stationary Dataflow in 65-nm CMOS," *IEEE TVLSI*, vol. 28, 2020.
[7] B. Zimmer *et al.*, "A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm," *IEEE JSSC*, vol. 55, 2020.
[8] X. Si *et al.*, "A Twin-8T SRAM Computation-In-Memory Macro for Multiple-Bit CNN-Based Machine Learning," in *IEEE ISSCC*, 2019.
[9] S. Yin *et al.*, "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks," *IEEE JSSC*, vol. 55, 2020.
[10] Q. Dong *et al.*, "A 351 TOPS/W and 372.4 GOPS Compute-in-Memory SRAM Macro in 7nm FinFET CMOS for Machine-Learning Applications," in *IEEE ISSCC*, 2020.
[11] H. Valavi *et al.*, "A 64-tile 2.4-Mb In-Memory-Computing CNN Accelerator Employing Charge-Domain Compute," *IEEE JSSC*, vol. 54, 2019.
[12] Z. Jiang *et al.*, "C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism," *IEEE JSSC*, vol. 55, 2020.
[13] H. Jia *et al.*, "A Programmable Heterogeneous Microprocessor Based on Bit-Scalable In-Memory Computing," *IEEE JSSC*, vol. 55, 2020.
[14] M. Kang *et al.*, "Deep In-Memory Architectures for Machine Learning–Accuracy Versus Efficiency Trade-Offs," *IEEE TCAS-I*, vol. 67, 2020.
[15] J. Zhang *et al.*, "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array," *IEEE JSSC*, vol. 52, 2017.
[16] J. Su *et al.*, "A 28nm 64Kb Inference-Training Two-Way Transpose Multibit 6T SRAM Compute-in-Memory Macro for AI Edge Chips," in *IEEE ISSCC*, 2020.