Hybrid RRAM/SRAM In-Memory Computing for Robust DNN Acceleration

Gokul Krishnan*, *Student Member, IEEE,* Zhenyu Wang*, *Student Member, IEEE,* Injune Yeo*, *Member, IEEE,* Li Yang*, *Student Member, IEEE,* Jian Meng*, *Student Member, IEEE,* Maximilian Liehr†, Rajiv V. Joshi‡, *Fellow, IEEE,* Nathaniel C. Cady†, *Member, IEEE,* Deliang Fan*, *Member, IEEE,* Jae-sun Seo*, *Senior Member, IEEE,* Yu Cao*, *Fellow, IEEE*

Abstract*—RRAM-based in-memory computing (IMC) effectively accelerates deep neural networks (DNNs) and other machine learning algorithms. On the other hand, in the presence of RRAM device variations and lower precision, the mapping of DNNs to RRAM-based IMC suffers from severe accuracy loss. In this work, we propose a novel hybrid IMC architecture that integrates an RRAM-based IMC macro with a digital SRAM macro using a programmable shifter to compensate for the RRAM variations and recover the accuracy. The digital SRAM macro consists of a small SRAM memory array and an array of multiply-and-accumulate (MAC) units. The non-ideal output from the RRAM macro, due to device and circuit non-idealities, is compensated by adding the precise output from the SRAM macro. In addition, the programmable shifter allows for different scales of compensation by shifting the SRAM macro output relative to the RRAM macro output. On the algorithm side, we develop a framework for the training of DNNs to support the hybrid IMC architecture through ensemble learning. The proposed framework performs quantization (weights and activations), pruning, RRAM IMC-aware training, and employs ensemble learning through different compensation scales by utilizing the programmable shifter. Finally, we design a silicon prototype of the proposed hybrid IMC architecture in the 65nm SUNY process to demonstrate its efficacy. Experimental evaluation of the hybrid IMC architecture shows that the SRAM compensation allows for a realistic IMC architecture with multi-level RRAM cells (MLC) even though they suffer from high variations. The hybrid IMC architecture achieves up to 21.9%, 12.65%, and 6.52% improvement in post-mapping accuracy over state-of-the-art techniques, at minimal overhead, for ResNet-20 on CIFAR-10, VGG-16 on CIFAR-10, and ResNet-18 on ImageNet, respectively.

I. Introduction

Modern deep neural networks (DNNs) outperform humans for a variety of applications such as computer vision and natural language processing. Higher accuracy comes at the cost of increased computational complexity, depth, and width for the DNN. The increased DNN complexity poses great challenges to traditional accelerator architectures that separate both the memory and computation unit [1]. In contrast, RRAM-based IMC accelerators combine both memory access and computation into a single unit. RRAM is a two-terminal device with programmable resistance representing the weights of the DNN and has high density, fast read speed, high memory accessing bandwidth, and good compatibility with CMOS fabrication technology. The crossbar-based RRAM-based IMC provides a dense and parallel structure to achieve high performance and energy efficiency for DNN acceleration. Prior works with RRAM-based crossbar architectures have shown up to 1,000× improvement in energy efficiency as compared to CPUs/GPUs [2–4]. The enhanced energy-efficiency is attributed to a full-custom design, higher density, and faster memory bandwidth [2, 3].

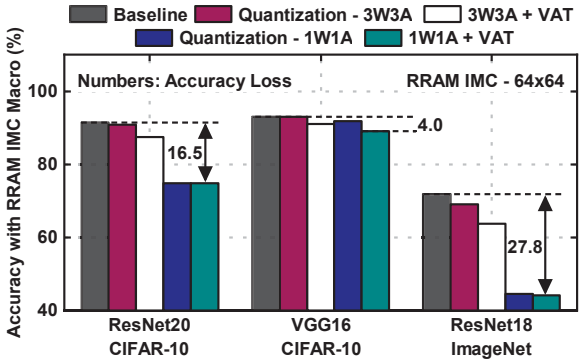

Fig. 1. Accuracy with RRAM IMC macro for three different DNNs for both CIFAR-10 and ImageNet datasets. The baseline model deals with a 32-bit floating-point model, quantization refers to fixed-point precision for activation and weights (e.g., 3W3A means 3-bit weight and 3-bit activation), and VAT refers to variation-aware training with the RRAM variations [5].

However, RRAM device suffers from several non-idealities such as limited resistance levels, device-to-device write variations, stuck-at-faults, and limited R_{off}/R_{on} ratio, posing a significant challenge to designing reliable RRAM-based IMC architectures [5–12]. The non-idealities within the RRAM device results in a deviation of the programmed weight values (resistance value), causing a significant reduction in post-mapping accuracy for DNNs. Furthermore, the crossbar structure of the IMC, with its limited array size, requires splitting of the large convolution (conv) or fully-connected (FC) layers into partial operations. Such partial operation (conv/FC) results in further error due to the limited precision of the peripheral circuits of the RRAM-based IMC crossbar.

To mitigate the post-mapping accuracy loss in DNNs, variation-aware training (VAT) and special encoding schemes are employed [5–9, 13]. VAT exploits the inherent DNN redundancy by embedding the device variations (σ), based on a log-normal or normal distribution model, into the training process to achieve a variation-tolerant model [5–9, 13]. To further understand VAT, we evaluate the post-mapping accuracy for three DNNs across CIFAR-10 and ImageNet datasets. We perform in-training quantization for both weights [14] and activations [15]. In addition, we extract RRAM device variation

*Authors are with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe 85287, AZ.
E-mail: {gkrish19, zwang586, iyeo3, lyang166, jmeng15, dfan, jseo28, Yu.Cao}@asu.edu
†Authors are with the State University of New York Polytechnic, NY, USA.
E-mail: {mliehr2, ncady}@sunypoly.edu
‡Author is with the IBM T. J. Watson Research Center, NY, USA.
E-mail: {rvjoshi}@us.ibm.com

Manuscript received April 07, 2022; revised June 11, 2022; accepted July 05, 2022. This article was presented in the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES) 2022 and appears as part of the ESWEEK-TCAD special issue.

from our 65nm SUNY 1T1R device (average variation (σ_{avg}) of 0.3) to perform VAT. Finally, we perform a hardware-aware training for the DNN by splitting the conv and FC layers into partial operations based on the IMC crossbar size (we use 64×64 [13]). Fig. 1 shows the accuracy of the quantized VAT trained RRAM IMC macro. Although VAT improves the accuracy, it does not achieve the same accuracy as the baseline 32-bit floating-point (FP-32) model, with up to 27.8% accuracy loss for ImageNet class DNNs. Hence, there is an urgent need to bridge this gap in the accuracy for RRAM-based IMC acceleration of DNNs.

To address this, in this work, we first propose a hybrid RRAM/SRAM IMC architecture for robust DNN acceleration. The proposed hybrid architecture utilizes an RRAM-based IMC macro, an SRAM-based macro, and a programmable shifter. The RRAM macro consists of the RRAM IMC crossbar, decoder, and associated peripheral circuits. The SRAM macro consists of an SRAM memory array, buffers, and an output stationary CMOS-based multiply-and-accumulate (MAC) engine. The output from the SRAM macro (clean) is added to the noisy RRAM macro output to create an ensemble model and achieve bit-level compensation. Furthermore, the degree of compensation is controlled by utilizing a programmable shifter. Depending on the DNN, different scales of the shift operation are performed on the SRAM macro output to achieve varying degrees of compensation for the RRAM macro output. To illustrate the efficacy of the architecture, we design a test-chip in the 65nm SUNY process [16] to demonstrate the proposed hybrid RRAM/SRAM IMC architecture. We utilize an RRAM macro with a 64×64 IMC and associated peripheral circuits, and a dedicated control logic. Furthermore, we design an SRAM macro with a 32×64 SRAM memory array and an output stationary MAC engine [17]. Finally, a custom control logic is designed to synchronize both RRAM and SRAM macro. The programmable shifter is implemented outside the chip for simplicity.

Next, on the algorithm side, we develop a framework for the training of the DNNs to support the hybrid IMC architecture. The proposed framework performs in-training quantization for both weights and activations following [14, 15], structured pruning [18], RRAM IMC-aware training, and support for different compensation scales through the programmable shifter. The parallel SRAM macro addition is performed in a layer-wise manner. In addition, structured pruning is performed on the SRAM macro weights to achieve minimal hardware overhead. For efficient hardware execution, the precision of the RRAM and SRAM macros, activations, and shift scale are kept constant across all layers of the DNN. To accurately model the RRAM device in the framework, RRAM data is collected from a fully integrated 1T1R structure on a 300mm wafer, using a custom RRAM module within the SUNY 65nm process. We plan to open-source the algorithm framework upon acceptance of this work.

We perform extensive experiments on four DNNs across CIFAR-10 and ImageNet datasets at different precision (weights and activations). We show that at higher RRAM macro precision and SRAM macro precision, the proposed method achieves near FP-32 accuracy (at 1× and 2× RRAM variations of 65nm RRAM device). Furthermore, we show that the proposed hybrid IMC architecture with SRAM compensation opens up the opportunity for a realistic IMC architecture with multi-level RRAM cells (MLC) even though they suffer from high variations. Compared to state-of-the-art methods, the proposed hybrid IMC architecture achieves up to 21.9%, 12.65%, and 6.52% improvement in post-mapping accuracy with minimal overhead for ResNet-20 on CIFAR-10, VGG-16 on CIFAR-10, and ResNet-18 on ImageNet, respectively.

In addition, we analyze the overhead from the SRAM macro and programmable shifter. In terms of training time, the proposed method incurs up to a 25% increase in training time compared to the VAT method. We evaluate the hardware cost overhead for the SRAM macro by extracting the post-layout area and power measurements from the 65nm test-chip. The proposed hybrid IMC architecture achieves small overhead in terms of memory requirement (up to 24%), area (up to 20%), and power (up to 2.6%) across different DNNs and datasets. The major contributions of this work are as follows:

- We propose a novel hybrid RRAM/SRAM IMC architecture that utilizes an RRAM IMC macro with MLC cells, SRAM macro, and a programmable shifter for robust DNN acceleration,
- We develop a training framework to enable the hybrid IMC architecture that supports quantization, structured pruning, RRAM IMC-aware training, and different compensation scales through the programmable shifter,
- *Experimental evaluation of the hybrid IMC architecture shows that the SRAM compensation opens the opportunity for a realistic IMC architecture with multi-level RRAM cells.* Compared to state-of-the-art methods, the proposed hybrid IMC architecture achieves up to 21.9%, 12.65%, and 6.52% improvement in post-mapping accuracy with minimal overhead for ResNet-20 on CIFAR-10, VGG-16 on CIFAR-10, and ResNet-18 on ImageNet, respectively.
- Finally, we design a test-chip using the 65nm SUNY process to demonstrate the proposed hybrid IMC architecture and analyze the hardware performance.

II. Related Work

A. RRAM-based IMC Architecture

RRAM-based IMC architectures provide a promising alternative to conventional von-Neumann architectures [2–4, 8, 19–21]. The crossbar-based IMC structure efficiently combines both memory access and analog-domain computation into a single unit for DNN acceleration. In addition to the IMC crossbar array, peripheral circuits such as analog-to-digital converter (ADC), WL drivers, decoder, and column multiplexers are used. Furthermore, a digital-to-analog converter (DAC) [3] or bit-serial computing with a shift and add circuit [2] is employed for multi-bit activations. Finally, a point-to-point or network-on-chip (NoC) interconnect is utilized to perform the on-chip data movement. However, prior works focus more on the hardware performance, with little focus on the effect of non-idealities associated with an RRAM-based IMC on DNN acceleration. To address this, in this work, we propose

a hybrid RRAM/SRAM IMC architecture and design a test-chip in 65nm for robust DNN acceleration. The proposed hybrid architecture incorporates the effect of device and circuit properties into the accuracy estimation while ensuring the best hardware performance.

B. Mitigation of Post-Mapping Accuracy Loss

Several methods have been proposed in prior works to mitigate the post-mapping accuracy loss for RRAM-based in-memory acceleration of DNNs [22]. Closed-Loop-on-Device (CLD) and Open-Loop-off-Device (OLD) perform iterative read-verify-write (R-V-W) operations at the RRAM device till the resistance converges to the desired value [23]. [7, 24] utilize VAT based on known device variation (σ) characterized from RRAM devices, while [5] combines VAT with dynamic precision quantization to mitigate the post-mapping accuracy loss. [21] utilizes a post-mapping training by selecting a random subset of weights and mapping them to an on-chip memory to recover the accuracy. Meanwhile, [8] utilizes knowledge distillation and online adaptation for accuracy mitigation. The authors in [8] utilize an SRAM-based IMC as the parallel network, while [21] propose to use a register file and a randomization circuit. Neither works [8, 21] consider the activation quantization, real RRAM mapping, and the actual hardware implementation. At the same time, [6, 9] propose to use a custom unary mapping scheme by mapping the MSB and LSB of the weights to RRAM devices based on the individual cell variations and bit significance. These approaches partially recover the accuracy but fail to consider the effect of activation quantization and the hardware implementation for the DNN model. In addition, these methods assume a known RRAM device variation for each RRAM device and utilize an average scale of variation.

In contrast, in this work, we propose a hybrid RRAM/SRAM IMC architecture that incorporates weight and activation quantization, mapping of DNN to the IMC, and overall hardware performance (through a test-chip designed in 65nm SUNY process). Furthermore, the proposed method utilizes bit-wise RRAM variations extracted from our 65nm 1T1R RRAM device, thus having a one-to-one correspondence with the hardware.

III. Hybrid IMC Architecture

In this section, we detail the hybrid IMC architecture proposed in this work. Fig. 2 shows the overall block diagram of the proposed hybrid IMC architecture for one RRAM IMC and SRAM macro. The architecture consists of an RRAM IMC macro, an SRAM macro, a programmable shifter, adder, buffers, and associated control logic. The proposed hybrid IMC architecture utilizes an ensemble of the output from the RRAM IMC macro and the SRAM macro combined using a programmable shifter and an adder circuit. Each macro functions independently with the associated control logic. The control logic also handles the scale of the shifter to facilitate different compensation of the RRAM macro by the SRAM macro. Finally, each layer of the DNN utilizes a number of RRAM IMC and SRAM macros to perform the DNN acceleration.

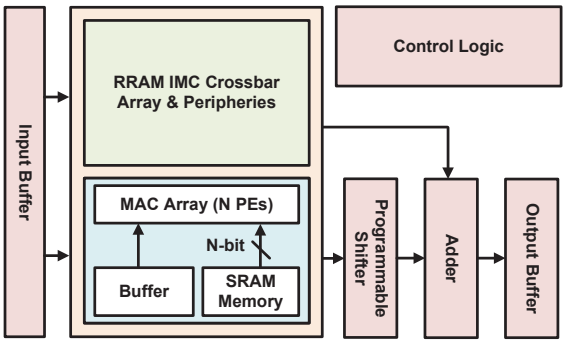

Fig. 2. Block diagram of the proposed hybrid RRAM/SRAM IMC architecture. Both the RRAM and SRAM macros compute in a parallel manner to generate the output. A programmable shifter allows for different scales of compensation using the SRAM macro. The overall output is an ensemble of the RRAM and SRAM macro outputs.

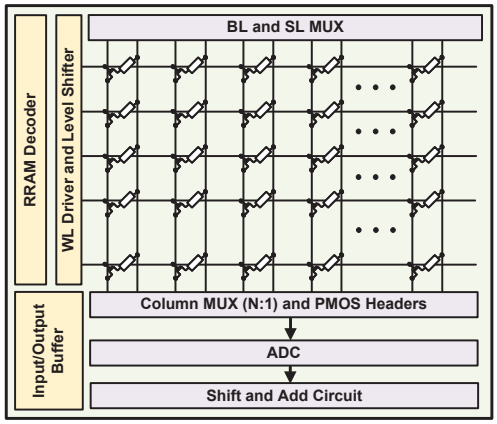

Fig. 3. Block diagram of the RRAM IMC macro within the hybrid architecture. The macro consists of a crossbar array of RRAM cells, a decoder, PMOS headers, column multiplexers (mux), BL and SL mux, shift and add circuit, and ADC.

A. RRAM IMC Macro

Fig. 3 shows the architecture of the RRAM IMC macro. The RRAM macro consists of an RRAM-based IMC crossbar structure of a specific size. Each cross-point in the array consists of a 1T1R RRAM multi-level cell. The 1T1R cell connects the transistor (gate) to the wordline (WL), while the two terminals of the RRAM are connected to the select-line (SL) and the bitline (BL). The RRAM-based IMC crossbar utilizes analog domain computation to perform the MAC operation. Each IMC crossbar has associated peripheral circuits such as analog-to-digital converter (ADC), PMOS header, column multiplexer, BL and SL mux, WL driver and level shifters, and RRAM decoder. The column multiplexer is used to share the read-out circuit (ADC and PMOS header) across multiple columns of the IMC crossbar array. The ADC performs the conversion of the analog output to the digital domain. In this work, instead of employing a digital-to-analog converter (DAC), we perform bit-serial computing over multiple cycles for multi-bit activations. The ADC output is accumulated based on the input bit significance using shifter and adder circuits to compute the MAC output. Finally, the overall result is generated by accumulating the outputs from each IMC crossbar array.

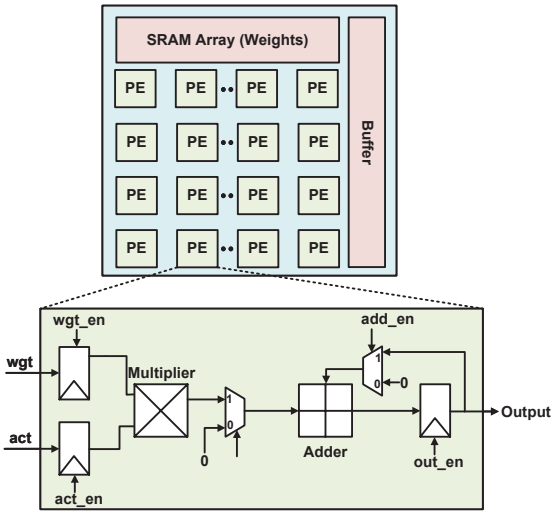

Fig. 4. Block diagram of the SRAM macros within the hybrid architecture. An SRAM array stores the weights while an array of processing elements (PE) perform the MAC operations.

B. SRAM+MAC Engine Macro

Fig. 4 shows the block diagram of the SRAM macro within the hybrid IMC architecture. The SRAM macro consists of an array of processing elements (PE) of MAC engines, SRAM memory array, buffers, and associated control logic. The SRAM memory array stores the weights while the inputs are streamed into the PEs through the buffer. Each PE utilizes an output stationary computation flow to reduce on-chip data movement. The multiplier and adder support the fixed-point MAC operations with the required precision for the SRAM macro output. The final output is obtained pixel-wise across feature maps and moved to the output buffer within the SRAM macro. Note that the read word size of the SRAM memory array is equal to the number of parallel PE to ensure maximum utilization and throughput. The SRAM macro performs the computations in parallel with the RRAM macro, thus avoiding any reduction in the overall hardware performance.

C. Programmable Shifter

The programmable shifter performs the shift operation for the output from the SRAM macro, as shown in Fig. 2. The different scales of shifting allow for the compensation of different bits of the RRAM macro output. The shift operation follows a right shift within the hybrid IMC architecture. Fig. 5 shows an example of 1-bit, 2-bit, and 3-bit SRAM macro output compensation for a 2-bit RRAM macro output. We show three shift cases across all configurations. Fig. 5(a) shows RRAM macro output at 2-bit and SRAM macro output at 1-bit. First, the blue region shows the case when the SRAM macro output compensates only for the MSB of the RRAM macro output, i.e., no shift. Such compensation provides a larger impact as the position of compensation of the RRAM macro output has higher significance. Next, the grey region shows the case when the SRAM macro output compensates only the LSB of the RRAM macro output through a 1-bit shift. Finally, the purple region shows a 2-bit shift for the SRAM macro output, thus adding a higher precision for the RRAM macro output by adding an extra bit. In this case, neither of the RRAM output bits are individually compensated, while the extra bit provides an increased precision to the overall output. Similarly, Fig. 5(b) and Fig. 5(c) show the cases for 2-bit and 3-bit SRAM macro compensation. The optimal choice of the scale of shift depends on the DNN algorithm, dataset, the extent of RRAM device variations, and the hardware performance overhead (Section V-F).

D. 65nm Hybrid IMC Test Chip

To demonstrate the efficacy of the proposed hybrid IMC architecture, we design a test-chip using a custom RRAM module within the 65nm SUNY process. Fig. 6 shows the layout of the 65nm test-chip of the hybrid IMC architecture. The test-chip consists of a RRAM macro, an SRAM macro, and associated testing structures such as scan chain. The shifter circuit is implemented outside the chip for higher testing flexibility. Finally, the test-chip is designed for an operating frequency of 100MHz.

1) RRAM IMC Macro: The RRAM macro consists of a 64×64 IMC crossbar array of 1T1R cells. In addition to the IMC crossbar array, peripheral circuits such as ADC, PMOS header, BL/SL/column multiplexers, WL driver and level shifter, and RRAM decoder are custom designed in the 65nm process. A 64-to-1 BL and SL one-hot multiplexer are utilized for the programming of RRAM cells. Furthermore, to share eight columns across each read-out circuit, an 8-to-1 column multiplexer is designed using a transmission gate and sized carefully to reduce the overall resistance of the circuit. The read-out circuit within the RRAM macro utilizes a flash ADC with a 3-bit resolution and a PMOS header. A single column (BL) combined with the PMOS header forms a resistance divider circuit, which converts the accumulated current to voltage. The voltage is then used as the input to the ADC, converting the analog output to the digital domain. Overall, eight read-out circuit instances are utilized across the 64 columns of the IMC crossbar array. Note that the PMOS header is sized appropriately to ensure that the resistance is much lower than the minimum resistance from the IMC crossbar array (with only one row turned on). Finally, input and output buffers are used to store the activations.

The RRAM decoder performs the overall control of the macro. The decoder utilizes a finite state machine (FSM) with three main states to generate the required control signals. Furthermore, the decoder also performs the operation of driving the inputs to the WL through the driver and level shifters. During the write state, the RRAM cells can be programmed one-by-one by choosing a specific WL, BL, and SL. Next, the compute state performs the MAC operations in a parallel fashion across rows and columns. The sharing of the columns requires the column multiplexer to choose columns sequentially to generate the output. Hence, 8 cycles are required to perform the MAC operations with the 64×64 IMC crossbar array. Furthermore, during the compute state the decoder enables the ADC to perform the analog to digital conversion for the MAC output. The ADC output is then moved to the buffer that holds the value until it is moved outside the chip using the scan chain. Finally, the new input

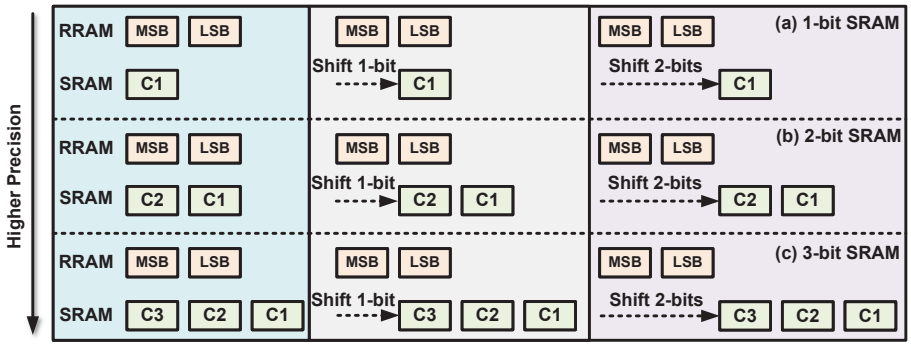

Fig. 5. Functioning of the programmable shifter within the hybrid IMC architecture. (a) 1-bit SRAM macro output compensates for the MSB in the blue region, LSB in the grey region (1-bit shift), and adds an extra bit for increased precision in the purple region. A similar operation is performed for both (b) 2-bit and (c) 3-bit SRAM macro outputs.

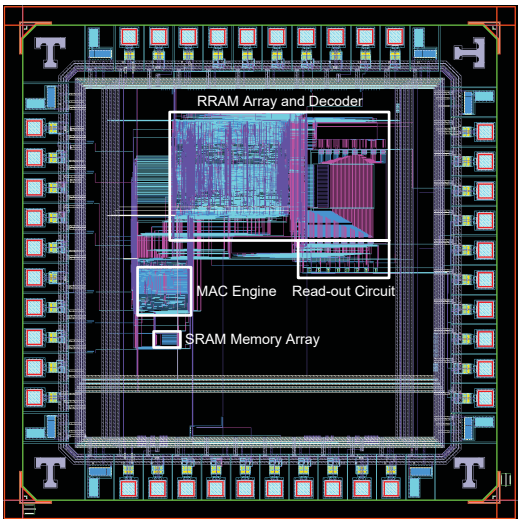

Fig. 6. Layout of the 65nm test-chip of the proposed hybrid IMC architecture. The prototype consists of a 64×64 1T1R RRAM crossbar array and associated peripheral circuits. In addition, the prototype implements an SRAM memory array of size 32x64 with a 16-word size and 16 MAC engines that utilize an output stationary dataflow.

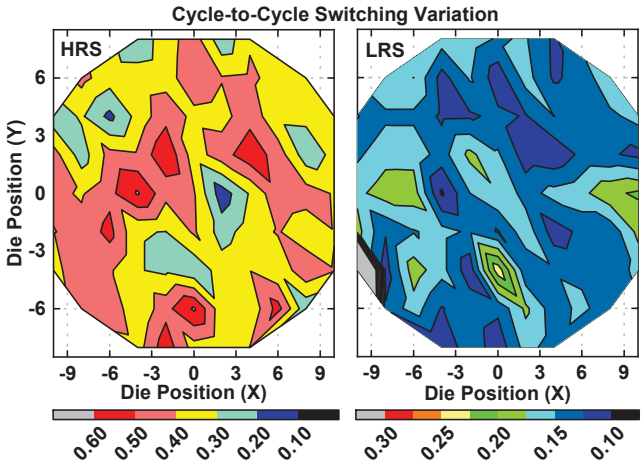

Fig. 7. HRS (left) and LRS (right) cycle-to-cycle switching variation across the 300mm wafer at 65nm [16]. HRS state has a higher variation than LRS.

state is utilized within the decoder to accept the next stream of input activations.

2) SRAM Macro: The SRAM macro consists of an SRAM memory array, MAC engine array (PE array), and associated control logic. The SRAM memory consists of a 32×64 SRAM cells with a read-out word size of 16 bits. To match the SRAM memory read-out size, the MAC engine consists of an array of 16 PEs that implement an output stationary dataflow. The memory read out, and the number of computation units is matched to achieve the best performance and utilization. Next, a custom control logic is implemented utilizing an FSM. The FSM consists of 4 main states, namely, idle, load, compute, and finish. The idle state is the default state the SRAM macro assumes upon power-up of the chip. The load state is utilized to perform the loading of the weights and activations to the SRAM memory and local input buffer. The load state triggers a counter that automatically moves the data from the scan chain to the corresponding memory/buffer. The SRAM memory with 64 columns is divided into 16 sets, with each column multiplexer servicing four columns of the array. Next, the compute state is utilized to perform the MAC operations. A counter is triggered such that it reads the weights from the SRAM memory and the input buffer to perform the MAC operation within the PEs. All PEs operate in a parallel fashion on different data, thus providing high throughput. Each PE performs the computations for a pixel within an output feature map. Through this, one pixel across 16 different output feature maps is generated. In addition, PE control is generated to enable the D flip-flops (DFF) at the input and the subsequent datapath to follow the output stationary dataflow. Finally, the finish state is utilized to denote the completion of the MAC operations and the transfer of the output data to the scan chain. We note that, for the increased flexibility of testing, we perform the computation of multi-bit inputs with bit-serial processing. The output is then post-processed outside the chip to obtain the final result.

IV. Hybrid IMC Training Framework

In this section, we detail the algorithm framework developed for the hybrid RRAM/SRAM IMC architecture. The overall framework is developed using the Python programming language and the PyTorch deep learning framework.

A. Statistical RRAM Device Models

To accurately model the RRAM device, data is collected from a fully integrated 1T1R RRAM structure on a 300mm

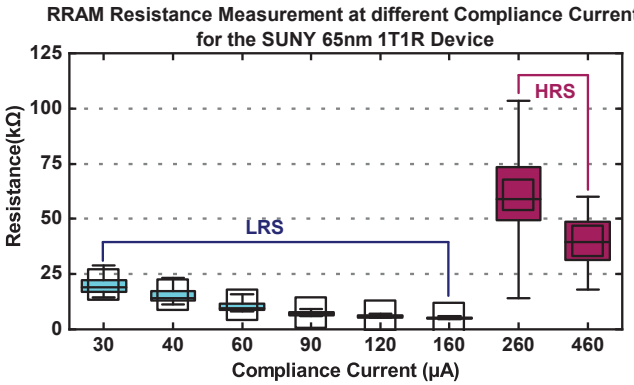

Fig. 8. Box and whisker plot showing the eight distinct resistance levels (6 LRS and 2 HRS) within our 65nm 1T1R RRAM device. Different compliance currents lead to different resistance levels.

TABLE I
RRAM DEVICE VARIATION FOR DIFFERENT BIT

Level		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
RRAM State		LRS						HRS	
Average Variation (σ)	**1-bit**	0.1035	NA						
	2-bit	0.1035	0.2760	NA					
	3-bit	0.1035	0.2760	0.2259	0.3549				

wafer (at room temperature), using a custom RRAM module within the SUNY 65nm process [10]. In this work, we focus on a multi-level RRAM device. Each RRAM device has a size of 100nm×100nm and utilizes a device stack comprised of a 6nm HfO_2 mem-resistive switching layer, a 6nm PVD Ti oxygen exchange layer (OEL), and TiN electrodes (top and bottom). Pulses with a magnitude of 1V – 1.2V and width of 10μs are used for the set/reset operation of RRAM devices.

Fig. 7 shows the wafer-level cycle-to-cycle switching variations for the 65nm RRAM device measured using the pulse-based switching technique. The high-resistance state (HRS) has a higher variation up to 0.6 σ, while the low-resistance state (LRS) has a lower variation up to 0.2 σ. The average variation (σ_{avg}) for the entire range of HRS and LRS across the wafer amounts to 0.3. To further understand the RRAM variations, we analyze the distinct resistance levels that can be achieved for the 65nm 1T1R RRAM device. Fig. 8 shows the box and whisker plot of the measured resistance at different compliance currents for the multi-level 65nm RRAM device. The 65nm RRAM device achieves up to 8 distinct levels with 6 LRS and 2 HRS states. Furthermore, at the single device level, the LRS achieves a lower variation compared to HRS. Hence, the 65nm RRAM device can support up to 3-bit data.

Next, we analyze the variations for each bit when mapped to the 65nm 1T1R RRAM device. Table I summarizes the level-wise and bit-wise RRAM device variation up to 3-bits for the 65nm 1T1R RRAM device. For a 1-bit value, only two levels are needed to map the data to the RRAM device. Hence, the lowest two resistance levels (LRS) can be utilized, thus reducing the overall RRAM device variations. Simultaneously, for a 2-bit value, four levels are required to map the data to the RRAM device. The first two levels utilize the lowest two resistance levels from the 1-bit case, while the third and fourth levels utilize higher resistance levels (LRS) with higher variation, as shown in Fig. 8. Finally, for a 3-bit data, eight resistance levels are required to map to the RRAM device. The first four levels utilize the same resistance levels as that for the 2-bit case. Meanwhile, the third bit further utilizes four resistance states of which two are LRS and two are HRS for the 65nm RRAM device. Hence, for accurate RRAM variation modeling, we utilize the bit-wise variation models within the hybrid IMC training framework.

Algorithm 1: Training Methodology for the hybrid RRAM/SRAM IMC architecture

Input: DNN, RRAM weight precision (W_{RRAM}), RRAM output precision (A_{RRAM}), SRAM weight precision (W_{SRAM}), SRAM output precision (A_{SRAM}), overall activation precision (A), shift scale (b_{shift}), SRAM pruning group size (G_{prune}), Training epochs M and N, and bit-wise RRAM variations (σ_{bit})
Output: Trained Hybrid RRAM/SRAM IMC model
for *RRAM Model* **do**
  Initialize DNN model randomly
  Perform in-training quantization for weights (W_{RRAM}) and activations (A_{RRAM})
  /* **Model A** */
  Layer-wise split conv and FC layer into partial conv/FC based on RRAM crossbar size
  Load trained quantized weights from **Model A**
  Add partial conv/FC outputs to generate final layer-wise output
  for *Epoch* $\leq M$ **do**
    Add RRAM variations (bit-wise) to weights
    Train DNN model
  end
end
/* **Model B** */
for *SRAM Model* **do**
  Create parallel SRAM model layer-wise with size 100% of RRAM model and randomly initialize weights
  /* **Model C** */
  Add **Model B** output layer-wise to the **Model C** output with b_{shift} shift
  /* **Model D** */
  for *Epoch* $\leq N$ **do**
    Perform in-training quantization and group-wise pruning (G_{prune}) for **Model C** weights (W_{SRAM}) at activation precision A_{SRAM}
    Perform in-training quantization for overall layer-wise activation output to A
    Backpropagation: Freeze **Model B** weights with no update. Only update **Model C** weights
  end
end
Save final trained **Model D** and perform inference

B. Training for Hybrid IMC Architecture

Algorithm 1 details the methodology utilized to train the DNN for the hybrid RRAM/SRAM IMC architecture.

1) RRAM Macro Training: The training of the RRAM macro is performed in two stages. The first stage performs in-training quantization for the DNN model, while the second stage performs the RRAM IMC-aware training in the presence of quantization. First, the DNN model weights are randomly initialized. Next, we perform in-training quantization for the weights with precision W_{RRAM} and RRAM macro output (A_{RRAM}) and train the DNN to generate Model A. In this work, we keep the weight and activation precision the same for the RRAM macro model (W_{RRAM} equal to A_{RRAM}). We utilize the quantization method in [14] for the weights and [15] for the activations. Next, we split each conv and FC layer into partial MAC operations based on the IMC crossbar size. For our test-chip, we utilize a 64×64 IMC crossbar array. Finally, the output from each partial MAC operation is accumulated to generate layer-wise output activations. The quantized weights from Model A are then loaded to the partial conv/FC operations, and RRAM variations (σ_{bit}) are added using the log-normal distribution [8] (at each epoch) in a bit-wise manner following Table I. Finally, the model is trained for M epochs to generate the RRAM IMC-aware trained model (Model B). Lines 3–13 of Algorithm 1 show the RRAM macro model training.

2) Training SRAM Macro and Programmable Shifter: Once the RRAM macro is trained (Model B), we perform the training for the SRAM macro with the programmable shifter. First, we add a parallel model (Model C) for the SRAM macro with the same size as that of RRAM macro model and randomly initialize weights in a layer-wise manner (100% size as that of the RRAM macro model). Next, we add the output from Model C to that of Model B in a layer-wise manner. The Model C output is shifted based on the scale of shifting (b_{shift}) utilized in the programmable shifter. We note that the scale of shift is kept constant across all layers of the DNN. Through this, we generate the DNN structure that follows the hybrid IMC architecture (Model D). Thereafter, Model D is trained such that the weights of the SRAM Macro model (Model C) are quantized to W_{SRAM} and the activations to A_{SRAM} following the methodology in [14] and [15], respectively. Furthermore, the output from each layer within Model D (after adding the Model B and shifted Model C outputs) is quantized to the overall layer-wise activation precision A. During the training of Model D, the weights of Model B are frozen without any update during backpropagation while the Model C weights are updated. Hence, the RRAM macro model serves as the backbone model while the SRAM macro model assists it.

Next, to reduce the overhead from the SRAM macro, we utilize group-wise pruning [18] with a group-size of G_{prune} for the Model C weights. The pruning utilizes weight-penalty clipping with a self-adapting threshold [18], as shown below:

$$\hat{\mathcal{L}} = \mathcal{L}(f(\mathbf{x}; \{\mathbf{W}_l\}_{l=1}^{L}), \mathbf{t}) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{G_i} \min(\|W_{l,i}\|_2; \delta_l) \quad (1)$$

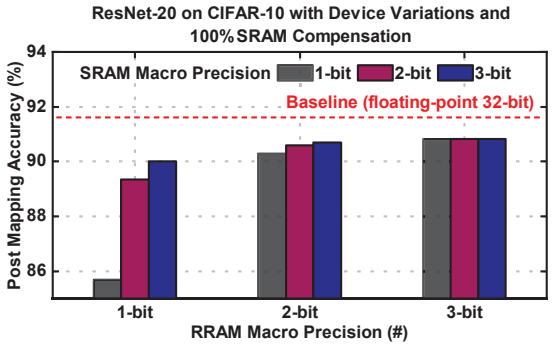

Fig. 9. Post-mapping accuracy with the proposed hybrid IMC architecture for ResNet-20 on CIFAR-10. A higher SRAM macro precision leads to better compensation across different RRAM precision.

$$\delta_l = \alpha . \frac{1}{G_l} \sum_{i=1}^{G_i} \|\mathbf{W}_{l,i}\|_2 \quad (2)$$

where δ_l denotes the layer-wise self-adapting clipping threshold, L is the number of layers, G_l is the number of groups in the SRAM macro model for the l-th layer, λ is the hyper-parameter to be tuned based on the dataset, and α is the scaling coefficient. The pruning is performed group-wise along the output channel dimension: for layer l with SRAM macro weight matrix $W_l \in \mathbb{R}_{\mathbb{Q}}^{N_{of} \times N_{if} \times K_x \times K_y}$, we choose a group of size G_{prune} along the N_{if} dimension, where the maximum value of G_{prune} is determined by the number of PEs in the SRAM macro ($\mathbb{R}_{\mathbb{Q}}$ denotes quantized SRAM macro weights). The training for Model D is performed for N epochs, where N is less than M. Overall, the trained model consists of an RRAM macro model with weight precision W_{RRAM} and output activation precision A_{RRAM}, a structured sparse SRAM macro model with weight precision W_{SRAM} output activation precision A_{SRAM}, overall layer-wise activation precision A, and a shift scale of b_{shift}. Lines 14–22 of Algorithm 1 show the SRAM macro with programmable shifter model training.

V. Experiments and Results

We perform extensive experiments to evaluate the proposed hybrid RRAM/SRAM IMC architecture from both an algorithm and a hardware standpoint. The algorithm experiments are performed on a Nvidia Quadro RTX 8000 GPU by utilizing the algorithm framework developed in this work (Section IV). We evaluate four different DNNs across two datasets: ResNet-20 on CIFAR-10 (0.27M), VGG-16 on CIFAR-10 (15M), ResNet-18 for ImageNet (11.5M), and MobileNet-v2 for ImageNet (3.4M). All experiments performed utilize the device models extracted (at room temperature) for the 65nm 1T1R RRAM device with up to 8 levels (Section IV-A). The weight and output activation precision values are kept the same within each macro (W_{RRAM}=A_{RRAM} and W_{SRAM}=A_{SRAM}) throughout the DNN. The experiments utilize a RRAM crossbar size of 64×64 for consistency with the 65nm test-chip. Furthermore, we evaluate up to a 3-bit RRAM macro weight and activation precision, a 3-bit SRAM macro weight and activation precision, a 6-bit overall layer-wise activation precision, and a 3-bit shift scale. Finally, for the hardware performance, we utilize the post-layout performance of the 65nm test-chip at 100MHz. The results obtained through VAT refer to the variation-aware training method in [5].

TABLE II
COMPREHENSIVE EVALUATION OF THE CHOICE OF PROGRAMMABLE SHIFTER ACROSS DIFFERENT DNNS FOR CIFAR-10 DATASET. (*AT SAME RRAM AND OVERALL ACTIVATION PRECISION). VAT – VARIATION-AWARE TRAINING [5].

Network	RRAM Macro Precision	Accuracy with VAT Only* (%)	SRAM Macro Precision	Activation Precision	Shifter Scale for SRAM Macro	Post-Mapping Accuracy - Ours (%)	Improvement in Accuracy over VAT only (%)
ResNet-20	2-bit	87.40	1-bit	3-bit	0-bit	90.28	2.88
					1-bit	**90.33**	2.93
					2-bit	89.68	2.28
	3-bit	87.45	3-bit	6-bit	0-bit	**90.92**	3.47
					1-bit	90.73	3.28
					2-bit	90.87	3.42
					3-bit	90.80	3.35
VGG-16	1-bit	89.02	2-bit	3-bit	0-bit	**92.56**	3.54
					1-bit	92.54	3.52
	3-bit	91.06	3-bit	6-bit	0-bit	**92.97**	1.91
					1-bit	92.96	1.90
					2-bit	92.92	1.86
					3-bit	92.90	1.84

A. *Effect of different Scales of SRAM Compensation*

We analyse the effect of different scales of SRAM compensation by varying the SRAM macro precision. Fig. 9 shows the post-mapping accuracy for ResNet-20 on CIFAR-10 across different RRAM macro and SRAM macro precisions. No pruning is performed for the SRAM macro weights, and no shift is applied. The baseline FP-32 ResNet-20 model for CIFAR-10 dataset achieves 91.34% accuracy. Consider an RRAM macro precision of 1-bit. At an SRAM macro precision of 1-bit, the SRAM compensates entirely for the variations within the RRAM macro. Meanwhile, at a higher SRAM macro precision, in addition to the variation compensation the SRAM adds additional bits to increase the dynamic range and the precision of the RRAM macro output for the DNN layer. Hence, at higher SRAM macro precisions, higher accuracy is achieved with up to 90.2%.

Simultaneously, consider an RRAM macro precision of 3-bits. A 3-bit RRAM macro precision provides higher density and better hardware performance, but suffers from higher RRAM device variations. At an SRAM macro precision of 1-bit, the MSB of the RRAM macro is compensated, thus providing a high degree of compensation. Hence, the higher RRAM macro precision and the MSB compensation result in a higher post-mapping accuracy of 90.01%. At an SRAM macro precision of 2-bit, both the MSB and the second bit are compensated, thus providing a higher accuracy of 90.7%. Finally, at a 3-bit SRAM macro precision, all the three bits of the RRAM macro output are compensated, thus providing maximum accuracy of 90.92%. Hence, the SRAM macro compensates for the RRAM macro variations, thus achieving higher accuracy. Through this, the SRAM macro compensation, the hybrid IMC architecture helps exploit the advantage within MLC RRAM cells.

B. *Optimal Shifter Configuration*

In this section, we analyze the effect of the scale of shift utilized in the hybrid IMC architecture on the post-mapping accuracy. We note that no pruning is performed on the SRAM macro in this experiment. Table II shows the post-mapping accuracy for different DNNs on CIFAR-10 dataset across varying RRAM and SRAM macro precisions at different shifting scales. Consider ResNet-20 on CIFAR-10 at 2-bit RRAM macro precision and 1-bit SRAM macro precision. We utilize a 3-bit overall activation precision across layers. At a 0-bit shift scale, the SRAM output compensates for the MSB of the RRAM macro output, achieving a post-mapping accuracy of 90.28%. At the same time, a 1-bit shift for the SRAM macro compensates the LSB of the RRAM macro output, achieving an accuracy of 90.33% (within statistical error range). Finally, a 3-bit shift for the SRAM macro output results in the extension of the precision of the output with no compensation for the MSB and LSB of the RRAM macro output. Such a scale of shift results in reduced post-mapping accuracy (89.68%) due to the effect of the RRAM variations on the output. Similarly, we evaluate a 3-bit RRAM macro precision and a 3-bit SRAM macro precision. A 0-bit shift provides the best post-mapping accuracy as the SRAM macro compensates for all the bits of the RRAM macro output.

We repeat the same experiment with VGG-16 for CIFAR-10. For a 1-bit RRAM macro precision and a 2-bit SRAM macro precision, a 0-bit shift provides the highest accuracy. Similarly, a 0-bit shift provides the highest accuracy of 92.97% for a 3-bit RRAM and SRAM macro precision. We conclude that the optimal scale of shift depends on the DNN structure, RRAM and SRAM macro precisions, and the overall activation precision. To provide further context, we compare the post-mapping accuracy of the two DNNs to the FP-32 baseline accuracy. For ResNet-20 on CIFAR-10, the FP-32 model achieves 91.34% accuracy, while the best configuration hybrid IMC model achieves 90.92% accuracy. For VGG-16 on CIFAR-10, the hybrid IMC model achieves 92.97% accuracy compared to 93.04% for the FP-32 model. Finally, we compare the post-mapping accuracy with the hybrid IMC architecture to that with VAT only. The proposed method

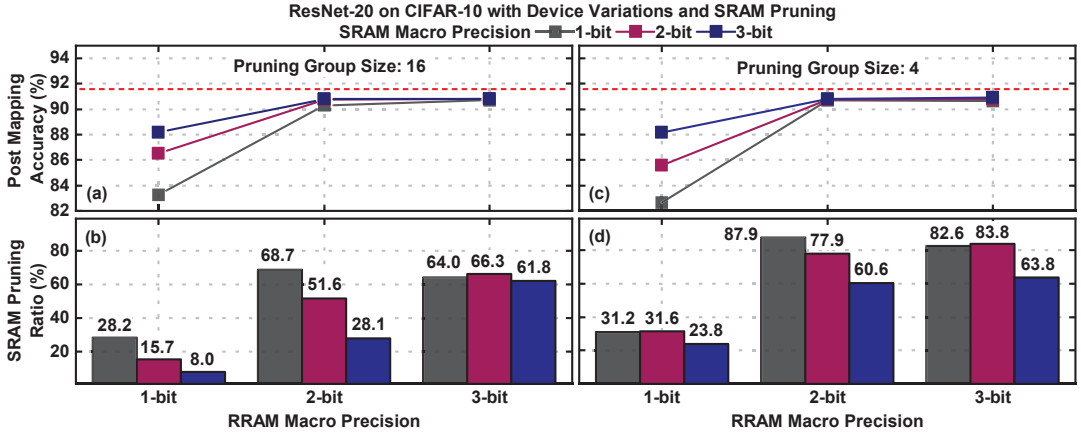

Fig. 10. Post-mapping accuracy for ResNet-20 on CIFAR-10 at different RRAM macro and SRAM macro precision. (a) Post-mapping accuracy with SRAM macro at G_{prune} of 16 and (b) Pruning ratio of the SRAM macro weights across different RRAM and SRAM precision at G_{prune} of 16, (c) Post-mapping accuracy for a G_{prune} of 4 for the SRAM macro weights, and (d) Pruning ratio of the SRAM macro weights at G_{prune} of 4. A lower group size leads to higher pruning at the same accuracy.

TABLE III
COMPREHENSIVE EVALUATION OF THE POST-MAPPING ACCURACY ACROSS DIFFERENT DNNS AND DATASETS WITH THE PROPOSED HYBRID IMC ARCHITECTURE. SRAM IS PRUNED WITH A GROUP SIZE OF 4. (*TOP-1 ACCURACY, **AT SAME RRAM PRECISION).

Network	Dataset	RRAM Precision	VAT Only Accuracy (%)**	SRAM Precision	Activation Precision	Shifter Scale	SRAM Pruning Ratio (%)	Post-Mapping Accuracy - Ours (%)	Accuracy Improvement over VAT only (%)
ResNet-20	CIFAR-10	2-bit	87.40	1-bit	3-bit	1-bit	87.8	**90.73**	**3.3**
VGG-16	CIFAR-10	3-bit	91.06	3-bit	6-bit	0-bit	98.9	**92.75**	**1.7**
ResNet-18	ImageNet	3-bit	63.79	2-bit	5-bit	2-bit	99.7	**69.21***	**5.4**
MobileNet-v2	ImageNet	3-bit	36.60	2-bit	5-bit	2-bit	36.6	**61.6***	**25.0**

consistently outperforms the VAT method and achieves near baseline (Floating-point 32-bit) accuracy.

C. Pruning Analysis of SRAM

The addition of the SRAM macro and programmable shifter results in an overhead for the hardware architecture. Hence, to reduce the overhead, we utilize group-wise pruning of the weights within the SRAM macro, as detailed in Section IV-B. Fig. 10 shows the post-mapping accuracy and the SRAM macro pruning ratio for two different pruning group sizes for ResNet-20 on CIFAR-10. The pruning ratio gives the amount of sparsity achieved through pruning. We note that we perform the pruning for the optimal configurations of the shift scale. We explore 1-bit to 3-bit precisions for both the RRAM and SRAM macros. Fig. 10(a) shows the post-mapping accuracy across different configurations for a pruning group size of 16. A higher SRAM macro precision allows for higher compensation for the RRAM macro output in the presence of sparsity. Next, we analyze the pruning ratio achieved with pruning across the different configurations, as shown in Fig. 10(b). For the 1-bit RRAM macro precision, a lower pruning ratio (sparsity) is obtained for best post-mapping accuracy since the SRAM macro output provides both compensation and increased precision and range. Meanwhile, a higher precision for the RRAM macro weights results in a higher accuracy compared to a binary model. Therefore, for 2-bit and 3-bit RRAM macro precision, the SRAM macro compensation is easier, allowing the pruning method to generate a higher pruning ratio for the SRAM macro. We repeat the same experiment for a smaller group size of 4, as shown in Fig. 10(c) and (d). The hybrid IMC architecture achieves similar post-mapping accuracy as that for the 16 group size. In addition, a higher pruning ratio is achieved for the SRAM in all cases due to the ability of the pruning algorithm to remove smaller groups of weights compared to a group size of 16. Hence, we conclude that a smaller group size and a higher RRAM precision is preferred as it provides higher sparsity (lower overhead), higher RRAM density, and similar post-mapping accuracy with the hybrid IMC architecture.

D. Overall Accuracy Results

Table III shows the overall comprehensive results for the hybrid IMC architecture. The baseline (floating-point 32-bit) accuracy for the DNNs are as follows; ResNet-20 - 91.32%, VGG-16 - 93.04%, ResNet-18 - 69.57%, and MobileNet-v2 - 71.87%. The choice of the optimal configuration is determined based on the post-mapping accuracy and the SRAM macro pruning ratio. For example, for VGG-16 on CIFAR-10 with 3-bit RRAM macro precision, an SRAM macro precision of 2-bit with a 2-bit shift achieves 92.76% post-mapping accuracy with a 95% SRAM pruning ratio. At the same time, a 3-bit SRAM macro precision with a 0-bit shift results in 92.75% accuracy at a 98.9% SRAM pruning ratio. Hence, we choose the 3-bit RRAM 3-bit SRAM configuration considering both the accuracy and SRAM macro pruning ratio.

We compare the post-mapping accuracy of the hybrid IMC architecture with that of the conventional VAT technique. For fair comparison, we utilize the same RRAM macro precision (weights and activation) for both approaches. For ResNet-20 on CIFAR-10, the hybrid IMC architecture achieves 3.3%

TABLE IV
POST-MAPPING ACCURACY AND SRAM MACRO PRUNING RATIO FOR 1X AND 2X RRAM VARIATION. WE USE A PRUNING GROUP SIZE OF 4.

RRAM Precision	SRAM Precision	Post-Mapping Accuracy (%)		SRAM Macro Pruning Ratio (%)	
		1x Var	2x Var	1x Var	2x Var
1-bit	1-bit	82.68	83.50	31.20	29.10
	2-bit	85.59	85.37	31.63	31.10
	3-bit	88.18	88.00	27.78	27.30
2-bit	1-bit	90.73	90.11	87.88	67.11
	2-bit	90.78	90.41	77.90	73.70
	3-bit	90.84	90.39	60.52	50.60
3-bit	1-bit	90.65	90.59	82.62	54.80
	2-bit	90.83	90.75	83.79	74.11
	3-bit	90.90	90.90	63.81	44.90

higher accuracy, while for VGG-16 on CIFAR-10 the hybrid architecture achieves 1.7% improvement in post-mapping accuracy. At the same time, for ImageNet models ResNet-18 and MobileNet-v2, the proposed hybrid IMC architecture achieves 5.4% and 25% improvement, respectively. For all the networks except MobileNet-v2, the proposed hybrid IMC architecture achieves greater than 87% sparsity (pruning ratio) for the SRAM macro. MobileNet-v2 being a small model for the ImageNet dataset requires highly accurate weights in the RRAM macro, i.e., a higher degree of compensation to ensure high post-mapping accuracy.

E. Evaluation with 2× RRAM Variation

In this section, we evaluate the post-mapping accuracy for the hybrid IMC architecture with 2× the bit-wise variations (σ) in Table I. Note that we perform shifting and pruning for the SRAM macro in this experiment. Table IV shows the comparison of the post-mapping accuracy and SRAM pruning ratio for ResNet-20 on CIFAR-10 at 1× and 2× RRAM variations. For each RRAM macro precision, the proposed hybrid IMC architecture achieves similar post-mapping accuracy for both 1× and 2× RRAM variations. At the same time, a 1× RRAM variations results in a higher SRAM pruning ratio and a lower SRAM macro overhead. The increased overhead for 2× RRAM variations is attributed to the higher SRAM compensation needed for better accuracy. Furthermore, for a given RRAM macro precision, the optimal SRAM macro precision results in higher accuracy for both 1× and 2× RRAM variations. Hence, the hybrid IMC architecture provides a scalable solution that opens the opportunity for multi-level RRAM devices for the acceleration of a wide range of DNNs across different datasets.

F. SRAM Macro and Shifter Overhead Analysis

We analyze the overhead for the SRAM and programmable shifter from an algorithm and hardware standpoint in the proposed hybrid IMC architecture. Note that we utilize the optimal hybrid IMC architecture model for each of the DNNs. Fig. 11(a) shows the training time overhead for the SRAM macro as a percentage of the training time for the RRAM

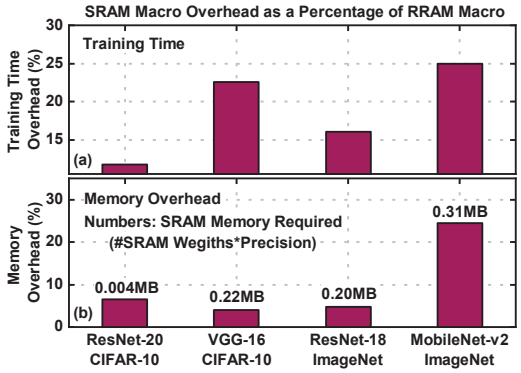

Fig. 11. (a) Training time overhead for the SRAM macro and the shifter. The proposed hybrid IMC architecture incurs at most 25% (compared to time of RRAM macro) overhead in training time. (b) Memory overhead for the SRAM macro compared to the RRAM macro. The hybrid IMC architecture incurs at most 24% overhead (compared to the memory of RRAM macro).

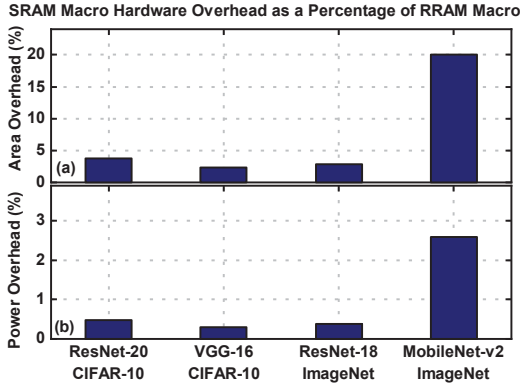

Fig. 12. (a) Area overhead and (b) power overhead of the SRAM macro as a percentage of the total RRAM macro area and power. The proposed hybrid IMC architecture results in a very low overhead with up to 20% area and 2.6% power overhead for state-of-the-art accuracy across different DNNs.

macro. For ResNet-20 and VGG-16 on CIFAR-10, the hybrid IMC architecture results in 12% and 23% overhead in training time. The increased overhead for VGG-16 is attributed to the larger model size compared to ResNet-20 (15M for VGG-16 compared to 0.27M for ResNet-20). At the same time, for ResNet-18 and MobileNet-v2, a 16% and 25% overhead in training time are incurred. Hence, the hybrid IMC architecture at most incurs a training time overhead of 25% of the RRAM macro training time. Next, we compare the overhead for the hybrid IMC training to the traditional read-verify-write (R-V-W) method utilized to achieve accurate RRAM device resistance levels. The R-V-W method requires time of the order of days to verify and write 100% of the RRAM cells [8]. At the same time, the proposed hybrid IMC architecture requires time of the order of a couple of hours to achieve near baseline (FP-32) accuracy, thus, providing a scalable solution.

Next, we evaluate the memory overhead as a percentage of the total RRAM memory requirement. Based on the SRAM pruning ratio, we evaluate the total number of non-zero bits that need to be stored within the SRAM macro. The structured pruning method employed allows the skipping of the zero weights in the SRAM macro. Fig. 11(b) shows the memory overhead for the hybrid IMC architecture. For ResNet-20 and VGG-16 on CIFAR-10, the hybrid IMC architecture incurs 6.5% (0.004MB SRAM to 0.067MB RRAM) and 4%

TABLE V
COMPARISON OF POST-MAPPING ACCURACY WITH STATE-OF-THE-ART METHODS. OURS-A SRAM MACRO WEIGHTS PRUNED; OURS-B$^{\#}$: SRAM MACRO WEIGHTS NOT PRUNED. (**PRECISION NOT REPORTED IN THE MANUSCRIPT)

Method	Weight Precision	Activation Precision	Post-Mapping Accuracy (%)	Accuracy Improvement by Our Work (%)
ResNet-20 on CIFAR-10				
Model Stability [10]	8-bit	8-bit	68.83	**21.9**
ReSNA [12]	8-bit	8-bit	88.43	**2.3**
Ours-A	**2-bit RRAM & 1-bit SRAM**	**3-bit**	**90.73**	**-**
VGG-16 on CIFAR-10				
DFP+DVA [5]	8-bit	**	80.1	**12.65**
Go Unary [6]	8-bit	**	87.94	**4.84**
KD+OSA [8]	4-bit	32-bit	92.57	**0.18**
Unary Opt [9]	8-bit	**	92.77	**0.20 (Compared to Ours-B)**
Ours-A	**3-bit RRAM & SRAM**	**6-bit**	**92.75**	-
Ours-B$^{\#}$	**3-bit RRAM & SRAM**	**6-bit**	**92.97**	**-**
ResNet-18 on ImageNet (Top-1)				
Unary Opt [9]	8-bit	**	62.69	**6.52**
Ours-A	**3-bit RRAM & 2-bit SRAM**	**5-bit**	**69.21**	**-**

(0.22MB SRAM to 5.63MB RRAM) overhead in the memory requirement. At the same time, for the ImageNet dataset, a 4.7% (0.2MB SRAM to 4.32MB RRAM) and 24% (0.31MB SRAM to 1.3MB RRAM) overhead in memory is incurred for ResNet-18 and MobileNet-v2, respectively. The increased overhead for MobileNet-v2 is attributed to the lower SRAM pruning ratio due to the need for higher compensation.

Finally, we evaluate the overhead in terms of the hardware performance (area and power) for the hybrid IMC architecture. We utilize the post-layout area and power measurements at 100MHz from the designed 65nm test-chip. The area and power of the RRAM macro consists of the RRAM IMC crossbar array, WL driver and shifter, RRAM decoder, flash ADCs (3-bits), PMOS headers, and BL/SL/column multiplexers. The power is measured by evaluating the total current drawn by each supply voltage (1.2V and 3.3V) and taking the product of the voltage and current. The PMOS header and ADC account for 90% of the power of the RRAM macro. The SRAM macro consists of the SRAM memory array, PE array, and buffers. A similar area and power estimation as that of the RRAM macro is performed for the SRAM macro.

Fig. 12(a) and Fig. 12(b) show the area and power overhead for the SRAM macro as a percentage of the RRAM macro. For ResNet-20 on CIFAR-10, an area and power overhead of 3.7% and 0.5% are incurred for the SRAM macro, respectively. At the same time, for VGG-16 on CIFAR-10, an area and power overhead of 2.3% and 0.3% are incurred for the SRAM macro, respectively. Meanwhile, MobileNet-v2 incurs the highest area and power overhead of 20% and 2.6%, respectively. The higher overhead is attributed to the higher compensation requirement resulting in a lower SRAM macro pruning ratio.

G. Comparison with Other Work

We compare the post-mapping accuracy for the proposed hybrid RRAM/SRAM IMC architecture with state-of-the-art methods. Table V shows the comparison for ResNet-20 and VGG-16 on CIFAR-10, and ResNet-18 on the ImageNet dataset. For ResNet-20 on CIFAR-10, compared to the method in [10] and [12], the proposed hybrid IMC architecture achieves 21.9% and 2.3% improvement in post-mapping accuracy at lower weight and activation precision, respectively.

Next, for VGG-16 on CIFAR-10, the proposed hybrid IMC architecture achieves 92.75% accuracy at 3-bit RRAM and SRAM macro precision with a 6-bit activation precision. Compared to [5, 6, 8], the hybrid IMC architecture achieves 12.65%, 4.84%, and 0.18% improvement in post-mapping accuracy, respectively. Furthermore, if the SRAM macro is not pruned, the proposed method achieves 92.97% accuracy, a 0.2% improvement in post-mapping accuracy compared to [9]. For ResNet-18 on ImageNet, compared to [9], the proposed hybrid IMC architecture achieves 6.52% higher post-mapping accuracy at 3-bit RRAM macro precision, 2-bit SRAM macro precision, and a 5-bit activation precision. Authors in [5, 6, 8, 9] do not discuss the quantization activation precision. The hybrid IMC architecture with a lower activation precision provides higher hardware performance and accuracy. Furthermore, [8] utilizes the larger ImageNet VGG-16 model with 3 FC layers for the CIFAR-10 dataset (134M parameters). In this work, we utilize the smaller CIFAR-10 model for VGG-16 with 15M parameters. Finally, both [6] and [9] utilize a unary mapping scheme, thus requiring exact variation measurements for each cell in the RRAM IMC crossbar. A complete R-V-W (takes up to many days) needs to be performed to quantify the variations at the cell level. Hence, the proposed hybrid IMC architecture provides a more scalable solution for robust DNN acceleration. Overall, the improved accuracy is attributed to the hybrid IMC architecture with the optimal scale of SRAM compensation achieved through the programmable shifter. We carefully tune the precision (weights

and activations), the sparsity for the SRAM MAC engine macro, and the scale of shit for the SRAM macro output to obtain the best accuracy.

VI. Conclusion

In this work, we propose a novel hybrid RRAM/SRAM IMC architecture for robust DNN acceleration. The hybrid IMC architecture utilizes an RRAM IMC macro with MLC cells, an SRAM macro, and a programmable shifter. The output from the RRAM macro is compensated by the SRAM macro output to create an ensemble model and achieve bit-level compensation. The scale of compensation is controlled by using a programmable shifter for the SRAM macro output. Next, we develop a training framework to enable the hybrid IMC architecture that supports quantization, structured pruning, RRAM IMC-aware training, and different compensation scales through the programmable shifter. Finally, we design a test-chip using the 65nm SUNY process to demonstrate the efficacy of the proposed hybrid IMC architecture.

We perform detailed experiments across different DNNs and datasets to demonstrate the performance of the proposed hybrid IMC architecture. Compared to the conventional VAT method, the proposed hybrid IMC architecture achieves up to 25% improvement in post-mapping accuracy. In addition, compared to state-of-the-art methods, the proposed hybrid IMC architecture provides a scalable solution that achieves up to 21.9%, 12.65%, and 6.52% improvement in post-mapping accuracy with minimal overhead for ResNet-20 on CIFAR-10, VGG-16 on CIFAR-10, and ResNet-18 on ImageNet, respectively. Finally, through the experimental evaluation of the hybrid IMC architecture, we show that the SRAM compensation opens the opportunity for a realistic IMC architecture with multi-level RRAM cells.

VII. Acknowledgements

This work was supported by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, National Science Foundation under Grant No. 2144751, and SUNY Polytechnic Institute authors acknowledge AFRL award FA8750-19-1-0014.

References

[1] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *IEEE JETCAS*, 2019.

[2] G. Krishnan *et al.*, "Interconnect-aware Area and Energy Optimization for In-Memory Acceleration of DNNs," *IEEE Design & Test*, 2020.

[3] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*. ACM/IEEE, 2016.

[4] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "Floatpim: In-memory Acceleration of Deep Neural Network Training with High Precision," in *ISCA*. IEEE, 2019.

[5] Y. Long, X. She, and S. Mukhopadhyay, "Design of Reliable DNN Accelerator with Un-Reliable ReRAM," in *DATE*. IEEE, 2019.

[6] C. Ma *et al.*, "Go Unary: A Novel Synapse Coding and Mapping Scheme for Reliable Reram-based Neuromorphic Computing," in *DATE*. IEEE, 2020.

[7] I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "GENIEx: A Generalized Approach to Emulating Non-Ideality in Memristive Xbars using Neural Networks," in *DAC*. IEEE, 2020.

[8] G. Charan *et al.*, "Accurate Inference with Inaccurate RRAM Devices: Statistical Data, Model Transfer, and On-line Adaptation," in *DAC*. IEEE, 2020.

[9] Y. Sun *et al.*, "Unary Coding and Variation-Aware Optimal Mapping Scheme for Reliable ReRAM-based Neuromorphic Computing," *IEEE TCAD*, 2021.

[10] G. Krishnan *et al.*, "Robust RRAM-based In-Memory Computing in Light of Model Stability," in *IRPS*, 2021.

[11] V. Joshi *et al.*, "Accurate Deep Neural Network Inference using Computational Phase-Change Memory," *Nature communications*, 2020.

[12] X. Yang *et al.*, "Multi-Objective Optimization of ReRAM Crossbars for Robust DNN Inferencing under Stochastic Noise," in *ICCAD*. IEEE/ACM, 2021.

[13] Z. He *et al.*, "Noise Injection Adaption: End-to-end Reram Crossbar Non-Ideal Effect Adaption for Neural Network Mapping," in *DAC*. IEEE/ACM, 2019.

[14] S. Zhou *et al.*, "Dorefa-net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[15] J. Choi *et al.*, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[16] M. Liehr, J. Hazra, K. Beckmann, S. Rafiq, and N. Cady, "Impact of Switching Variability of 65nm CMOS Integrated Hafnium Dioxide-based ReRAM Devices on Distinct Level Operations," in *IIRW*. IEEE, 2020.

[17] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, "Optimizing Loop Operation and Dataflow in FPGA Acceleration of Deep Convolutional Neural Networks," in *ISFPGA*, 2017.

[18] L. Yang, Z. He, and D. Fan, "Harmonious Coexistence of Structured Weight Pruning and Ternarization for Deep Neural Networks," in *AAAI*, 2020.

[19] B. K. Joardar, A. Deshwal, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "High-Throughput Training of Deep CNNs on ReRAM-based Heterogeneous Architectures via Optimized Normalization Layers," *IEEE TCAD*, 2021.

[20] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A Pipelined Reram-Based Accelerator for Deep Learning," in *HPCA*. IEEE, 2017.

[21] A. Mohanty *et al.*, "Random sparse adaptation for accurate inference with inaccurate multi-level RRAM arrays," in *IEDM*. IEEE, 2017.

[22] W. Zhang *et al.*, "A circuit-algorithm codesign method to reduce the accuracy drop of rram based computing-in-memory chip," in *ICTA*. IEEE, 2020, pp. 108–109.

[23] B. Liu *et al.*, "Reduction and IR-drop Compensations Techniques for Reliable Neuromorphic Computing Systems," in *ICCAD*. IEEE, 2014.

[24] L. Chen *et al.*, "Accelerator-Friendly Neural-Network Training: Learning Variations and Defects in RRAM Crossbar," in *DATE*. IEEE, 2017.