PRIVE: Efficient RRAM Programming with Chip Verification for RRAM-based In-Memory Computing Acceleration

Wangxin He[1], Jian Meng[1], Sujan Kumar Gonugondla[2], Shimeng Yu[3], Naresh R. Shanbhag[4] and Jae-sun Seo[1]
[1]*Arizona State Univeristy, Tempe, AZ,* [2]*Amazon, New York, NY,* [3]*Georgia Institute of Technology, Atlanta, GA,*
[4]*University of Illinois Urbana-Champaign, Urbana, IL*

***Abstract*—As deep neural networks (DNNs) have been successfully developed in many applications with continuously increasing complexity, the number of weights in DNNs surges, leading to consistent demands for denser memories than SRAMs. RRAM-based in-memory computing (IMC) achieves high density and energy-efficiency for DNN inference, but RRAM programming remains to be a bottleneck due to high write latency and energy consumption. In this work, we present the Progressive-wRite In-memory program-VErify (PRIVE) scheme, which we verify with an RRAM testchip for IMC-based hardware acceleration for DNNs. We optimize the progressive write operations on different bit positions of RRAM weights to enable error compensation and reduce programming latency/energy, while achieving high DNN accuracy. For 5-bit precision DNNs, PRIVE reduces the RRAM programming energy by 1.82×, while maintaining high accuracy of 91.91% (VGG-7) and 71.47% (ResNet-18) on CIFAR-10 and CIFAR-100 datasets, respectively.**

***Index Terms*—Deep neural network, resistive RAM (RRAM), in-memory computing, RRAM programming, write-verify.**

I. Introduction

Deep neural networks (DNNs) have been successfully developed in many applications including computer vision, speech recognition, and others. As the complexity of DNN tasks increases, the number of weights or parameters in DNNs surges as well, leading to consistent demands for denser memories than SRAMs. Conventional DNN accelerator systems have used DRAM to store a large number of DNN weights, but DRAM requires cumbersome refresh operations and off-chip memory access consumes very high energy consumption [1]. Instead of using off-chip memory, several recent accelerators employed embedded non-volatile memory (NVM) such as resistive RAM (RRAM) [2], [3] and magnetic RAM (MRAM) [4], to store a large amount of weights fully on-chip and reduce the energy consumption for overall memory access.

While these works [2]–[4] demonstrated on-chip integration of embedded NVMs, the NVMs only served the purpose of storage, and physically-separate processing engines (PEs) performed the computation. In this case, the DNN weights are accessed row-by-row from the NVM array and communicated to the PEs. To further improve this bottleneck, in-memory computing (IMC) [5] has emerged as a promising scheme to embed computation inside the memory, thereby largely reducing the data transfer. Several different memory technologies, such as RRAM [6], [7], SRAM [8], and phase change memory (PCM) [9] have been investigated for IMC. Non-volatile resistive devices such as RRAM can naturally support IMC operations with multiple rows turned on, where the weighted sum current between the wordline voltage (representing DNN activations) and RRAM conductance (representing DNN weights) represents the dot-product result.

Most RRAM-based IMC works employ a weight stationary scheme, but still the RRAM devices need to be programmed often, to execute inference of different DNN models over time, or to run workloads that require frequent weight updates (e.g. DNN training). Efficient programming is one of the critical bottlenecks for RRAM, since RRAM write operation requires high voltage and latency. More importantly, a single write operation is insufficient to program a target conductance value to the RRAM device due to device and voltage variation.

A popular method to address this challenge is to iterate the process of writing the targeted value to an RRAM cell and reading the RRAM cell conductance for verification, which is referred to as write-verify or program-verify method [10], [11]. Besides RRAM, the conventional write-verify (CWV) method has been also employed for other devices to compute analog matrix-vector multiplications (MVMs) such as PCM [9] and MRAM [12]. The CWV method can minimize the write uncertainty and RRAM conductance variation, but multiple iterations are required for each cell, which incurs large latency and energy overheads. Furthermore, frequent writing operations can hurt the cell endurance and possibly limit the lifetime of the memory device [13].

Several prior works investigated reducing the RRAM programming energy for IMC macros/systems, by reducing the number of write iterations and compensating the induced error (due to the smaller number of write iterations) by different methods. SWIPE [14] proposed to perform programming from MSB to LSB, and compensate for the write error from MSBs by adjusting the programming of LSBs. Probabilistic early termination was proposed in [15], and the programming process was partitioned into the coarse predictive phase (with fewer write iterations) and the fine-tuning phase in [16]. However, [14]–[16] all reported only simulation results with behavioral RRAM models, and included unrealistic assumptions such as single-pulse programming for RRAM devices [14], [16].

This work is partially supported by NSF grant 1652866, JUMP C-BRIC and ASCENT programs (SRC programs sponsored by DARPA). This work was done outside of Amazon. (E-mail: jaesun.seo@asu.edu)

978-3-9819263-7-8/DATE23/© 2023 EDAA

Therefore, the claimed programming energy reduction of ∼10× likely will not be possible for actual RRAM hardware.

[17] is one of the few works that reported programming energy reduction with measurements of multi-level RRAM devices, but only row-by-row RRAM device readout was reported without turning on multiple rows as done for IMC and did not report accuracy results for DNNs.

In this work, we propose the Progressive-wRite In-memory program-VErify (PRIVE) scheme that is compatible with RRAM-based IMC hardware that has been prototyped in a commercial RRAM process. We performed RRAM programming with the PRIVE scheme and obtained measurements of the RRAM prototype chip, which was evaluated for VGG-7 and ResNet-18 DNNs on CIFAR-10 and CIFAR-100 datasets. To the best of our knowledge, this is the only work to date that reports RRAM programming energy reduction with an actual RRAM chip for IMC operation targeting DNN workloads. The main contributions of this work are:

- A new progressive write-verify scheme (PRIVE) is proposed for multi-bit weight programming using 1-bit-per-cell RRAM hardware. While inspired by prior work on RRAM programming energy reduction [14], we identify impractical aspects of [14] such as single-pulse programming and present a practical progressive programming scheme from MSB to LSB that is verified by RRAM chips.
- PRIVE implementation does not exhibit any overhead, e.g. DNN model re-training on the algorithm side [15] or any additional circuits on the hardware side, as only the RRAM programming method has changed compared to the CWV scheme.
- We determine optimal configurations to apply PRIVE by investigating two different low resistance state (LRS) values for RRAM programming and evaluate the trade-offs and corresponding DNN accuracy values.
- 1.82× energy reduction is achieved for the overall RRAM programming compared to the CWV scheme, while maintaining high DNN accuracy for VGG-7/ResNet-18 models on CIFAR-10 and CIFAR-100 datasets, based on RRAM chip measurements.

II. Background and Related Works

A. RRAM based in-memory computing

Recently, non-volatile memory (NVM) based IMC prototype chips have been reported in [6], [7] and SRAM-based IMC prototype chips have been reported in [8]. Although the NVM technology is less mature than CMOS, they have advantages such as non-volatility, low power consumption, CMOS compatibility, and exhibit higher density compared to SRAMs at the same technology node. To that end, we focus on RRAM-based IMC design in this work.

Fig. 1 shows the high-level overview of the RRAM prototype chip reported in [18], [19]. Each 1T1R cell can be programmed to a low resistance state (LRS) or a high resistance state (HRS). The RRAM conductance represents DNN weights and the wordline (WL) voltage represents the activations. When multiple WLs are activated, the number of RRAM cells driven by LRS versus HRS determines the analog bitline (BL) voltage, which is digitized by the analog-to-digital converter (ADC).

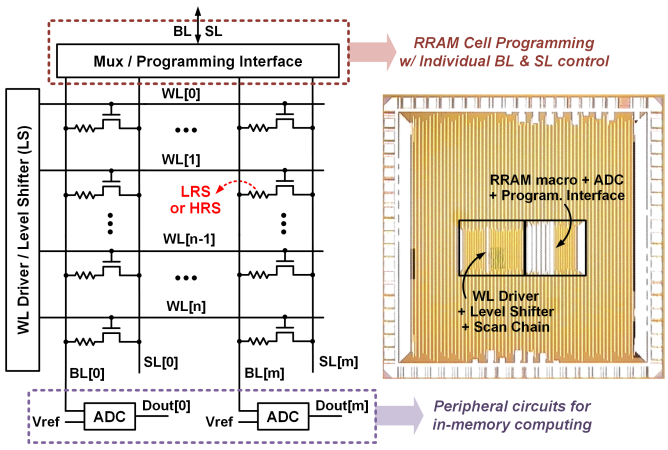


Fig. 1: High-level overview of the RRAM prototype chip [18].

B. Prior works on efficient RRAM programming

A number of prior works pursued efficient and accurate programming with the write-verify schemes on RRAM devices and arrays. Early research [10], [11] proposed write-verify method for RRAM programming. [13], [20] proposed write-verify method for multi-level RRAM devices, by tuning multiple parameters (e.g. bitline voltage, gate voltage, etc.) for set/reset processes. Building upon [13], [20], a two-step 2-bit-per-cell write-verify scheme was presented in [21]. Similar programming method such as POST [22] was proposed, which splits the write pluses into several small pulses for single-cell programming in the RRAM array. However, these works only focused on the accurate programming of a single RRAM cell, while multi-bit weight programming involving multiple RRAM cells has not been investigated.

For multi-bit weight programming, instead of iterating the programming of single RRAM cells excessively, SWIPE [14] proposed to compensate for the programming errors of more significant bits by adjusting the programming values of less significant bits. RRAM write energy reduction of up to 10× was reported, but only simulation results with behavioral device models were reported and incorrect assumptions such as single-pulse programming were made. [15] presented probabilistic early termination on programming single devices and optimized re-programming a subset of multiple RRAMs that constitute the multi-bit weight, reporting >3× programming cost reduction based on simulation results. [16] partitioned the programming process into predictive phase and the fine-tuning phase, reducing the RRAM programming energy by ∼90% based on simulation. Simultaneously programming multiple RRAM cells was assumed in the simulation, while this has not been verified with actual RRAM hardware. RADAR [17] performed multi-level RRAM programming with a coarse resistance control phase and a fine-tuning control phase and reported 2.4× programming energy reduction with actual RRAM hardware, but it did not perform IMC or evaluate the accuracy of DNN workloads.

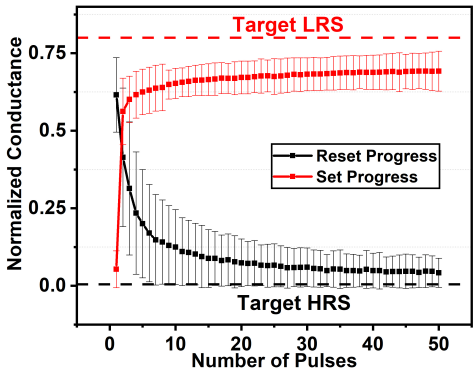


Fig. 2: RRAM chip measurements on the RRAM conductance with the number of pulses for set and reset processes.

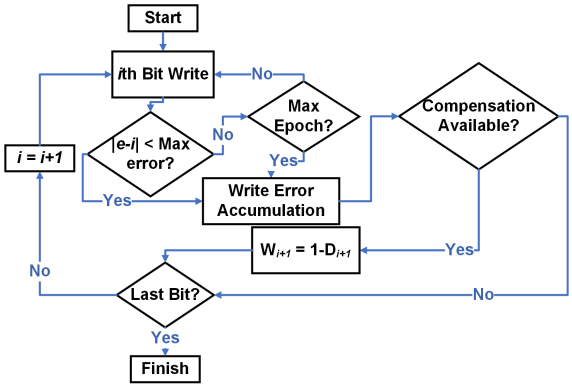


Fig. 3: The algorithm and RRAM programming flow of the proposed progressive write-verify algorithm (PRIVE). The less significant bits are employed to compensate for the RRAM programming error in more significant bits.

III. Proposed PRIVE Scheme

A. Limitations of prior works for RRAM hardware

Several important discrepancies exist between actual RRAM hardware and the assumptions used in simulation-based prior works to reduce RRAM programming energy [14]–[16]. The main discrepancy is in the number of pulses needed for the RRAM crossbar programming. In [14], [16], it is assumed that a single write pulse could be sufficient to program the RRAM devices for a target conductance. However, in practice, programming each RRAM device with an acceptable error rate requires multiple programming pulses, as illustrated from the RRAM chip measurement results in Fig. 2. The RRAM programming results in Fig. 2 are based on multiple pulses repeatedly writing RRAM cells to LRS and HRS on chips from [18]. Programming with multiple pulses is necessary for real RRAM hardware for three reasons:

1) The conductance error after the single-write pulse is too high, and this can be largely reduced and fine-tuned by programming using multiple pulses.
2) Due to the cell-to-cell variation, the single-write programming can result in a high variation of programming value. With adaptive multiple pulses of programming, cell-to-cell conductance variation can be reduced to a smaller value during multiple write-verify pulses, improving the stability against device non-ideality.
3) Aggressive high-voltage or wide pulses for single-write operations can potentially hurt the cell endurance, and increase the chances of breaking down the RRAM cell.

In addition, SWIPE assumes that the RRAM conductance programmed with single write pulses could either overshoot or undershoot the target LRS value. The existence of both positive and negative conductance errors at more significant bits could enable more error compensation capabilities at lower significant bits. However, especially with target LRS values that are configured to achieve a high on/off ratio, Fig. 2 shows that the intermediate conductance values during programming for LRS do not overshoot the target LRS value, which can limit the error compensation capabilities of SWIPE for certain weight values of DNNs.

B. Proposed progressive write-verify algorithm

To address the limitations described in Section III-A, we propose a progressive write-verify algorithm called PRIVE, as shown in Fig. 3. The PRIVE scheme is based on 1-bit-per-cell RRAM devices, where a low resistance state (LRS) represents "1", and a high resistance state (HRS) represents "0" as binary storage. For a positive N-bit weight, we employ N RRAM cells and program each RRAM cell with LRS or HRS.

In every write epoch, the error of each RRAM cell can be presented as $E_i = G_i - G_{iW}$, where G_i is the programmed conductance for the i-th cell. Suppose $D_i \epsilon \{0, 1\}$ is the desired state of i-th bit, and $W_i \epsilon \{0, 1\}$ is the actual state that RRAM is programmed with. By default, W_i is identical to D_i. Depending on the value of W_i, G_{iW} would be G_{HRS} or G_{LRS}, the typical conductance value of HRS or LRS value, respectively. The write iteration will stop when the $|E_i|$ is smaller than the maximum error threshold, or when the epoch loop number exceeds the maximum epoch limit. Then, for a n-bit weight programming, the error of the i-th bit is accumulated as:

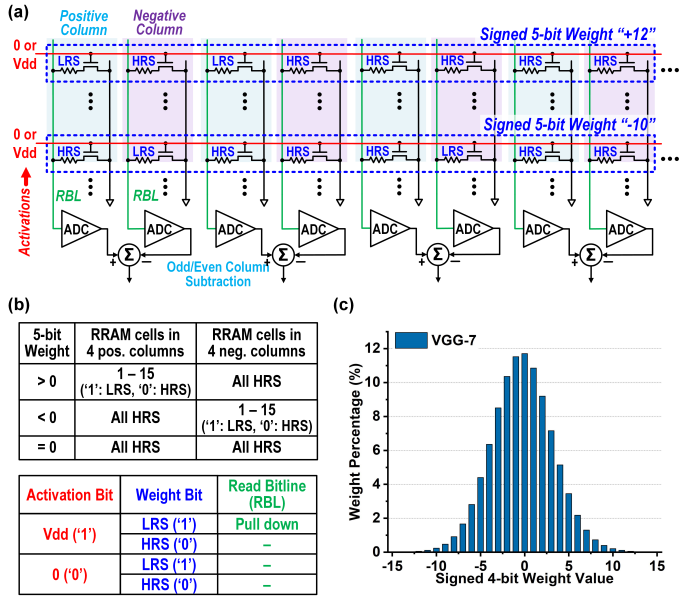
$$E \leftarrow E + E_i \times 2^{(n-i)} + (G_{LRS} - G_{HRS}) \times (W_i - D_i)^{(n-i)} \quad (1)$$

The accumulated error E is used to determine if the error in the current bit requires compensation in the next bit programming. If the error is larger than $(G_{LRS} - G_{HRS})$ conductance gap, the binary value for the next bit position is available for a bit-flipping compensation. This means that W_{i+1} will be effectively programmed as $W_{i+1} = 1 - D_{i+1}$, and this change will be reflected in the error accumulation for the $(i+1)$-th bit in Eq. (1).

In this work, we mapped the 4-bit weights with four 1-bit-per-cell RRAM devices. To analyze the effectiveness of the PRIVE scheme, we quantify the equivalent weight (W_{eq}) for each 4-bit weight as formulated in Eq. (2), and compare it with the ideal value.

$$W_{eq} = [\sum_{i=0}^{3} (G_i \times 2^i) - G_{HRS} \times 15] / G_{LRS} \quad (2)$$

For an ideal RRAM device with $G_{HRS} = 0$, the equivalent weight will be identical to the 4-bit weight value.



5-bit Weight	RRAM cells in 4 pos. columns	RRAM cells in 4 neg. columns
> 0	1 – 15 ('1': LRS, '0': HRS)	All HRS
< 0	All HRS	1 – 15 ('1': LRS, '0': HRS)
= 0	All HRS	All HRS

Activation Bit	Weight Bit	Read Bitline (RBL)
Vdd ('1')	LRS ('1')	Pull down
	HRS ('0')	–
0 ('0')	LRS ('1')	–
	HRS ('0')	–

Fig. 4: (a) Hardware implementation of PRIVE for the 5-bit weights. (b) Signed programming method truth table applied in this work with PRIVE. (c) Software quantization model mapping of PRIVE.

With the PRIVE scheme, the number of pulses required for write-verify iterations on each RRAM cell can be progressively reduced to an appropriate number, in order to balance the trade-off between precise RRAM programming and the energy reduction of RRAM programming. Note that PRIVE still follows the traditional write-verify at each pulse, and the proposed changes are made on the decision levels during each programming stage. Therefore, compared to the conventional write-verify progress, there does not exist any additional overhead or extra energy consumption to implement the PRIVE scheme on RRAM chips.

In this work, we set the default number of write-verify pulses to be 25 for the programming of each RRAM cell, since the error saturates after ∼25 iterations of CWV (Fig. 2) for the RRAM chips [18]. For multi-bit programming, MSB contributes the highest amount of write error, so we keep the number of programming pulses for MSB at 25. Then, we progressively reduce the number of pulses from MSB to LSB for the 4-bit weight programming in the PRIVE scheme, to balance the programming energy reduction and the programming accuracy. In particular, we use 25 pulses for bit 3 (MSB), 15 pulses for bit 2, 10 pulses for bit 1, and 5 pulses for bit 0 (LSB). These number of pulses (25-15-10-5) were chosen as they provided a good energy/accuracy balance, based on a coarse sweep of different pulse configurations (e.g. 25-10-5-2). Compared to the CWV scheme of using 25-25-25-25 pulses for 4-bit weights, the PRIVE scheme with 25-15-10-5 pulses reduces the total number of programming pulses by 45%, and this translates into proportional write latency and energy savings.

The main trade-off of PRIVE is the correctness of the single RRAM cell. As an example in Fig. 6(b), the errors on less significant bits can increase due to the reduced programming iterations, which may not get properly compensated (e.g. no LSBs available). However, this will only occur to a small portion of the weight levels, such as "7" and "15", and the overall DNN-level effect will be further discussed in Section IV.

IV. Experiment Results and Analysis

The PRIVE scheme is evaluated with RRAM prototype chips [18], [19] (Fig. 1), which were fabricated in a commercial CMOS process that monolithically integrates HfO_2 RRAM. For RRAM programming, we use SET voltage of 2.1V, SET pulse width of 100ns, SET gate voltage of 2.6V/2.3V for $6k\Omega$/$9k\Omega$ LRS, RESET voltage of 3.8V, RESET pulse width of 250ns, and RESET gate voltage of 4.0V.

We programmed 5-bit weights onto eight 1T1R RRAM cells with positive and negative columns [6] as shown in Fig. 4(a), throughout the entire RRAM array for both conventional and PRIVE schemes. The truth table of each 1-bit RRAM is shown in Fig. 4(b). The programmed value on even or odd columns will determine the sign of the weight, and all RRAM cells in the other columns will be programmed with HRS for analog subtraction according to Eq. (2). As a result, the RRAM-based IMC inference is performed with the signed weight values without introducing auxiliary offset columns [23]. Furthermore, the weights of the IMC inference are directly represented by the measured conductance values, which fully incorporate the non-idealities of the actual RRAM devices. The column-wise partial sum will be scaled by the difference between the typical HRS and LRS values, so arithmetic correctness will be preserved in the algorithm-level simulation.

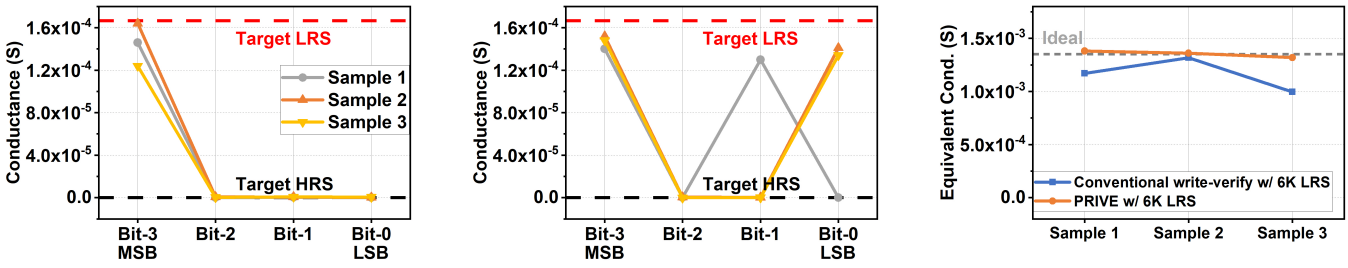
Based on the RRAM chip measurement data, we evaluate the DNN inference accuracy of RRAM IMC hardware with the pre-trained quantized DNN models on VGG-7 and ResNet-18 frames with CIFAR-10 and CIFAR-100 datasets. We program the quantized weights on the prototype chip and the measured conductance weight values are used to evaluate the overall DNN accuracy. We performed the overall experiments in the following steps:

1) Map the DNN weights onto existing RRAM chips with a given LRS target for both CWV and PRIVE schemes
2) Retrieve programmed conductance data from RRAM chips
3) Validate inference accuracy on pre-trained quantization models of VGG-7 and ResNet-18 DNNs for CIFAR-10 and CIFAR-100 datasets.

A. *Effectiveness of PRIVE programming*

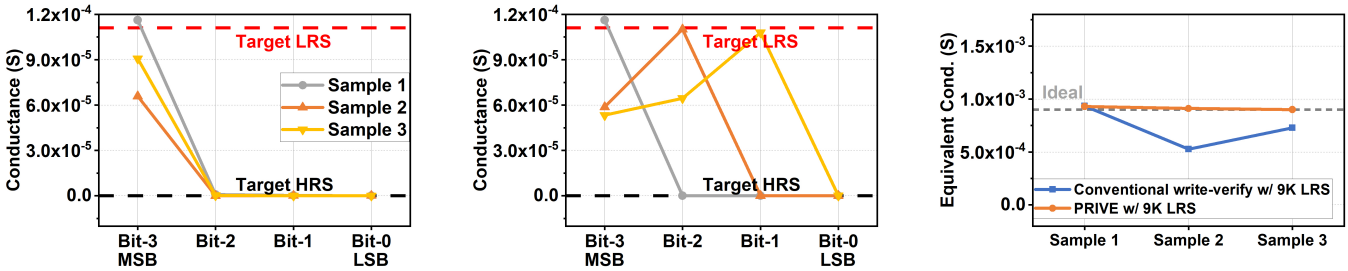
We experimented with two LRS values for RRAM programming: (1) the default $6k\Omega$ ($1.67\times10^{-4}S$) that achieves the highest on/off ratio of ∼100, and (2) a higher LRS value of $9k\Omega$ ($1.11\times10^{-4}S$) that achieves a lower on/off ratio but consumes a lower current and could exhibit better error compensation capability by PRIVE. $9k\Omega$ LRS can have better error compensation with PRIVE, because both overshoot and undershoot can occur during target conductance programming. As shown in Fig. 2, programming for $6k\Omega$ LRS will only exhibit undershoot with the write pulses.

The effectiveness of the PRIVE scheme is depicted in Fig. 5 (with $6k\Omega$ LRS) and Fig. 6 (with $9k\Omega$ LRS), where we selected



(a) (b) (c)

Fig. 5: RRAM programming for three samples of 4 RRAM cells for weight value of 8 ("1000") with $6k\Omega$ LRS: (a) conventional write-verify scheme, (b) the PRIVE scheme, and (c) effective 4-bit weight conductance comparison.



(a) (b) (c)

Fig. 6: RRAM programming for three samples of 4 RRAM cells for weight value of 8 ("1000") with $9k\Omega$ LRS: (a) conventional write-verify scheme, (b) the PRIVE scheme, and (c) effective 4-bit weight conductance comparison.

three samples of four RRAM pairs that are programmed to represent the 5-bit weight value of "+8" ("1000").

The conventional write-verify scheme will always program LRS for a '1' binary bit and HRS for a '0' binary bit with many iterations, and could achieve relatively more accurate programming on the RRAM devices. However, even if the programmed LRS conductance after write-verify iterations cannot reach the target LRS conductance as shown in Fig. 5(a) and Fig. 6(a), no further improvement or compensation can be made.

In contrast, the PRIVE scheme is aware of the errors during the programming from MSB to LSB, and is able to compensate for the programming error or conductance deviation in more significant bits by rearranging the programming states of less significant bits, as shown in Fig. 5(b) and Fig. 6(b). Fig. 5(c) and Fig. 6(c) show that the equivalent conductance of the PRIVE scheme is maintained very close to the ideal value for both $6k\Omega$ and $9k\Omega$ LRS cases, while noticeable deviations exist for the conventional write-verify scheme.

In Fig. 7, the differences between the equivalent weight W_{eq} (Eq. (2)) measured from the RRAM chip and the ideal positive 4-bit weight are compared for (1) CWV scheme with the default 25-25-25-25 programming, (2) CWV scheme with progressive 25-15-10-5 programming, and (3) PRIVE scheme with 25-15-10-5 programming, with target LRS of $6k\Omega$ and $9k\Omega$. The result of each datapoint is averaged from >100 measurements of 4 RRAM pairs. Larger W_{eq} errors are shown when we perform CWV with 25-15-10-5 epochs, but such errors are largely reduced in the PRIVE scheme owing to error compensation. While Fig. 6 shows that overshoot possibilities can provide better error compensation for $9k\Omega$ LRS in certain cases, Fig. 7

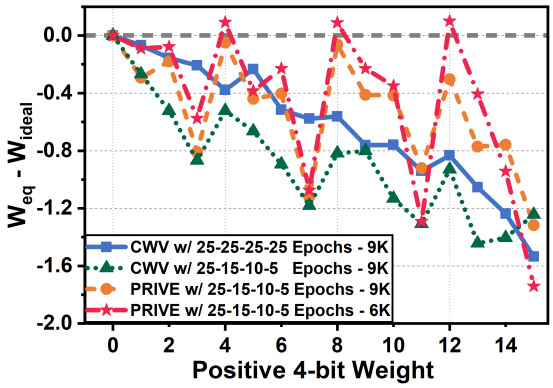


Fig. 7: The deviation of measured W_{eq} from ideal positive 4-bit weight for CWV and PRIVE schemes with different programming epochs.

shows that the average programming error of many RRAM devices is still lower for the $6k\Omega$ LRS.

For a few weight values, the PRIVE scheme cannot improve the W_{eq} error of the 25-15-10-5 conventional scheme, and notably, these weight values, such as "7" ("0111"), have more '1' bits in the binary representation. Although $9k\Omega$ LRS allows some amount of conductance overshoot during programming, as shown in Fig. 6(b), conductance undershoot is still more dominant during programming. Conductance undershoot needs to get compensated by flipping less significant bits from '0' to '1', but if the less significant bits are already filled with '1' (e.g. "0111", "0011"), the compensation capability of the PRIVE scheme is limited within the 5-bit weight programming.

TABLE I: The inference accuracy of the PRIVE scheme across different DNN models and datasets.

Model	Dataset	LRS Value	Baseline Accuracy	CWV Accuracy	**PRIVE Accuracy**
VGG-7	CIFAR-10	$6k\Omega$	92.53%	90.78%	**91.91%**
		$9k\Omega$		87.20%	**91.44%**
VGG-7	CIFAR-100	$6k\Omega$	69.90%	66.92%	**69.17%**
		$9k\Omega$		56.02%	**67.65%**
ResNet-18	CIFAR-10	$6k\Omega$	93.16%	91.56%	**92.61%**
		$9k\Omega$		86.00%	**91.97%**
ResNet-18	CIFAR-100	$6k\Omega$	72.56%	67.26%	**71.47%**
		$9k\Omega$		56.60%	**70.17%**

TABLE II: Comparison to prior works.

	[14]	[16]	[15]	[17]	**This work**
Hardware Verfication?	No	No	No	Yes	Yes
Weight Precision	2-to-9 bit	N/A	8-bit	3-bit-per-cell	5-bit
DNN Dataset	MNIST, CIFAR-10	MNIST, CIFAR-10	CIFAR-10, CIFAR-100	N/A	CIFAR-10, CIFAR-100
Network Type	LeNet-300-10, 8-Layer CNN	FC-NN, LeNet5, ResNet-20	ResNet-18, ResNet-34	N/A	VGG-7, ResNet-18
Accuracy Change	$-$ <1%	$-$2-4%	$+$0.23%	N/A	**$-$1-2%**
Write Energy Reduction	5-10×	up to 19×	∼3×	2.4×	1.82×

On the other hand, for weight values that have more '0' bits, e.g. "8" ("1000"), since fewer errors are produced at HRS, the PRIVE algorithm will have more capability to compensate the conductance undershoot error of programming "1" or LRS at more significant bits, by flipping the '0' bits at less significant bits to '1'. This leads to the result that, the PRIVE compensation works better in levels with more '0' bits (e.g. "8") than the levels with more '1' bits (e.g. "7"), as shown in Fig. 7.

B. DNN accuracy evaluation and comparison to prior works

Based on the RRAM chip measurements, we tested the inference accuracy of VGG-7 and ResNet-18 models for CIFAR-10 and CIFAR-100 datasets, and the results are shown in Table I. PRIVE provides 45% (1.82×) improvement in programming energy and latency reduction. Compared to the software baseline accuracy with the same precision, 1-2% accuracy degradation exists for CIFAR-10/100 datasets but the accuracy values are still higher than those of the CWV scheme for identical models/datasets.

We also compared the proposed PRIVE scheme and other works on RRAM programming energy reduction in Table II. Due to the reasons mentioned in Section III-A, the claimed energy reduction in [14], [15] likely will not be possible with real RRAM hardware. PRIVE employs RRAM chip measurement data to evaluate the effectiveness of realistic programming techniques for IMC designs. The universal optimization results shown in Table I and the flexibility of the PRIVE algorithm during the programming stage indicate the potential of PRIVE on more practical and efficient programming for the multi-bit RRAM array, which can be used on top of other multi-bit-per-cell approaches such as [17].

V. Conclusion

RRAM-based IMC accelerators can achieve high density and high energy-efficiency for DNN workloads, but programming RRAM devices consume high energy. To effectively reduce the RRAM programming energy for IMC accelerators, this work demonstrated a progressive write-verify algorithm called PRIVE, which was verified with RRAM chip measurements. We measured two different LRS values during the RRAM programming to verify the PRIVE error compensation effectiveness. We tested PRIVE hardware measurement data on several different inference models and achieved 1.82× write energy and latency improvement with minimal accuracy degradation.

References

[1] V. Sze et al. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 2017.

[2] Z. Li et al. RRAM-DNN: An RRAM and Model-Compression Empowered All-Weights-On-Chip DNN Accelerator. *IEEE JSSC*, 2021.

[3] M. Giordano et al. CHIMERA: A 0.92 TOPS, 2.2 TOPS/W Edge AI Accelerator with 2 MByte On-Chip Foundry Resistive RAM for Efficient Training and Inference. In *Symp. VLSI*, 2021.

[4] D. Rossi et al. A 1.3TOPS/W @ 32GOPS Fully Integrated 10-Core SoC for IoT End-Nodes with 1.7μW Cognitive Wake-Up From MRAM-Based State-Retentive Sleep Mode. In *IEEE ISSCC*, 2021.

[5] M. Kang et al. An Energy-Efficient VLSI Architecture for Pattern Recognition via Deep Embedding of Computation in SRAM. In *IEEE ICASSP*, 2014.

[6] Q. Liu et al. A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory Chip with Fully Parallel MAC Computing. In *IEEE ISSCC*, 2020.

[7] J.-H. Yoon et al. A 40-nm 118.44-TOPS/W Voltage-Sensing Compute-in-Memory RRAM Macro With Write Verification and Multi-Bit Encoding. *IEEE JSSC*, 2022.

[8] S. Yin et al. PIMCA: A 3.4-Mb Programmable In-Memory Computing Accelerator in 28nm for On-Chip DNN Inference. In *Symp. VLSI*, 2021.

[9] R. Khaddam-Aljameh et al. HERMES-Core—A 1.59-TOPS/mm^2 PCM on 14-nm CMOS In-Memory Compute Core Using 300-ps/LSB Linearized CCO-Based ADCs. *IEEE JSSC*, 2022.

[10] M. Cheng et al. TIME: A Training-In-Memory Architecture for RRAM-based Deep Neural Networks. *IEEE TCAD*, 2018.

[11] L. Gao et al. Programming protocol optimization for analog weight tuning in resistive memories. *IEEE EDL*, 2015.

[12] H. Noguchi et al. 4Mb STT-MRAM-based Cache with Memory-Access-Aware Power Optimization and Write-Verify-Write / Read-Modify-Write Scheme. In *IEEE ISSCC*, 2016.

[13] M. Zhao et al. Characterizing Endurance Degradation of Incremental Switching in Analog RRAM for Neuromorphic Systems. In *IEDM*, 2018.

[14] S. K Gonugondla et al. SWIPE: Enhancing Robustness of ReRAM Crossbars for In-Memory Computing. In *ACM/IEEE ICCAD*, 2020.

[15] Z. Meng et al. Write or not: Programming scheme optimization for rram-based neuromorphic computing. *ACM/IEEE DAC*, 2022.

[16] G. L. Zhang et al. An Efficient Programming Framework for Memristor-based Neuromorphic Computing. In *DATE*, 2021.

[17] B. Q. Le et al. RADAR: A Fast and Energy-Efficient Programming Technique for Multiple Bits-Per-Cell RRAM Arrays. *IEEE TED*, 2021.

[18] S. Yin et al. High-Throughput In-Memory Computing for Binary Deep Neural Networks With Monolithically Integrated RRAM and 90-nm CMOS. *IEEE TED*, 2020.

[19] W. He et al. 2-bit-per-cell RRAM-based in-memory computing for area-/energy-efficient deep learning. *IEEE SSC-L*, 2020.

[20] J. Chen et al. A parallel multibit programing scheme with high precision for rram-based neuromorphic systems. *IEEE TED*, 2020.

[21] W. Shim et al. Two-Step Write–Verify Scheme and Impact of the Read Noise in Multilevel RRAM-based Inference Engine. *Semiconductor Science and Technology*, 2020.

[22] C. Wang et al. Improving Multilevel Writes on Vertical 3-D Cross-point Resistive Memory. *IEEE TCAD*, 2020.

[23] X. Peng et al. DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies. In *IEEE IEDM*, 2019.