Characterization and Mitigation of IR-Drop in RRAM-based Compute In-Memory

Brian Crafton[1], Connor Talley[1], Samuel Spetalnick[1], Jong-Hyeok Yoon[2], and Arijit Raychowdhury[1]
[1]Georgia Institute of Technology
[2]Daegu Gyeongbuk Institute of Science and Technology
Email: arijit.raychowdhury@ece.gatech.edu

2022 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-6654-8485-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISCAS48785.2022.9937307

***Abstract*—Compute in-memory (CIM) is an exciting circuit innovation that promises to increase effective memory bandwidth and perform computation on the bitlines of memory sub-arrays. Utilizing embedded non-volatile memories (eNVM) such as resistive random access memory (RRAM), various forms of neural networks can be implemented. Unfortunately, CIM faces new challenges traditional CMOS architectures have avoided. In this work, we characterize the impact of IR-drop and device variation (calibrated with measured data on foundry RRAM) and evaluate different approaches to write verify. Using various voltages and pulse widths we program cells to offset IR-drop and demonstrate a 136.4× reduction in BER during CIM.**

I. Introduction

From the edge to the cloud, nearly all modern computing systems are heavily dependent on the capacity, bandwidth, and access time of memory systems [1]. At the same time, emerging applications such as machine learning and artificial intelligence require more bandwidth and on-chip capacity to achieve target performance [2]. These demands have given rise to tremendous research effort in both hardware accelerators [3] and software frameworks, yielding significant improvement in performance and energy efficiency. Despite the strong improvements, we face limitations using memory systems comprised of just SRAM and DRAM. These limitations, along with the slow decline of Moore's law, have inspired demand for new memory technologies and techniques to enable future workloads on future computing systems.

Fortunately, new circuit techniques and memory technologies are being actively researched to help push the limits of current CMOS technology. Compute in-memory is one such research thread that reads and accumulates multiple memory cells onto the same bitline (BL). This increases memory bandwidth and performs (binary) multiplication and addition without the use of CMOS logic. At the same time, eNVM such as RRAM and phase change RAM (PCRAM) are making strides towards commercial viability [1]. These memories offer high density non-volatile storage while being both logic and process compatible. Furthermore, these technologies store information through change of resistance which can enable multi-level storage and a more natural primitive for compute in-memory.

Despite these benefits, both CIM and eNVM face several challenges not before faced by traditional CMOS designs. First, because CIM accumulates several cells on the same bitline it increases total noise and reduces sensing margin for each state. Therefore, CIM will inherently have a higher bit error rate (BER) than traditional memory arrays which read a single wordline (WL) at a time. Second, CIM faces challenges with IR-drop. IR-drop is always present in memory, because some cells are farther from the read circuit than others and thus observe more wire resistance. This variable wire resistance has more impact in CIM, where multiple cells are read at the same time yielding a lower effective resistance. This means the parasitic wire resistance accounts for a larger fraction of the voltage drop on the bitline and yields higher variation and BER. Lastly, eNVM suffer from limited endurance and high write energy compared to SRAM or DRAM. These properties necessitate careful programming to guarantee endurance specifications and limit power consumption.

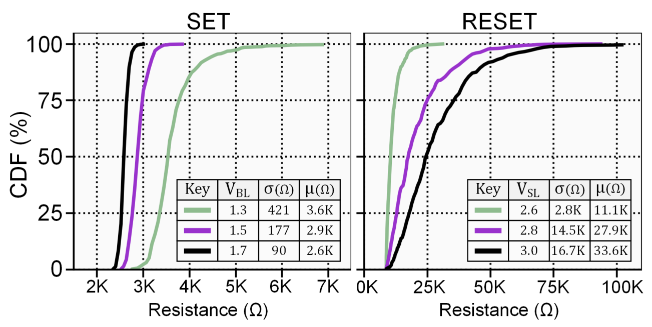
Recent work has attempted to mitigate the impact of these errors in several different ways. Various new forms of ECC have been applied to CIM [4]–[6]. Statistical methods have been used to maximize CIM performance (active wordlines) under error constraints [7]–[9]. Training a network to be robust to device variation induced error can be done both off-chip and on-chip. Off-chip training attempts to train a neural network to tolerate device variation induced errors [10]–[12], however this technique still results in accuracy degradation. On-chip training [13] can be done to minimize error for a specific chip, however this is expensive since each chip must be trained based on its own specific devices. Lastly, write-verify methods have been proposed to reduce the cell-to-cell variation [14]. However, current write-verify methods do not account for IR-drop, which has significant impact on CIM.

In this work, we characterize and evaluate various approaches to programming RRAM for CIM. We perform our experiments on a 40nm foundry RRAM test-chip array, whose circuit details appear in [15]. Next, we evaluate various iterative write-verify protocols based on BER and classification accuracy on ImageNet. We then propose a macro-level write protocol to account for IR-drop in our array. We find that by using various programming voltages or pulse widths we can program cells to offset the unique IR-drop that occurs at each cell (or WL). We demonstrate a 136.4× reduction in BER over a baseline method that does not consider IR-drop.

II. Background and Motivation

A. Compute In-Memory (CIM)

To implement VMM ($\vec{y} = W\vec{x}$), CIM systems encode the input vector $\vec{x}$ as wordline voltages and the weight matrix W as conductance states in a memory cell. The current through



Key	V_{BL}	σ(Ω)	μ(Ω)
🟩	1.3	421	3.6K
🟪	1.5	177	2.9K
⬛	1.7	90	2.6K

Key	V_{SL}	σ(Ω)	μ(Ω)
🟩	2.6	2.8K	11.1K
🟪	2.8	14.5K	27.9K
⬛	3.0	16.7K	33.6K

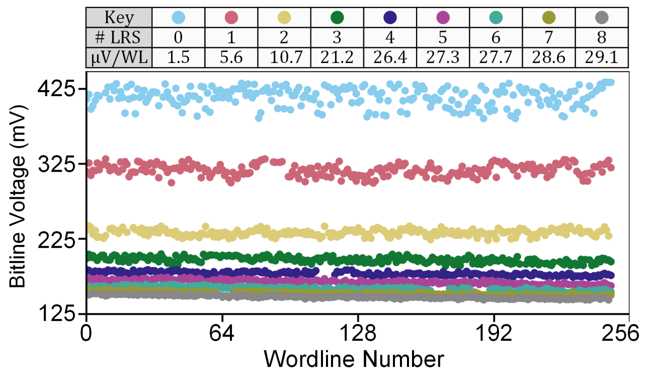
Fig. 1. CDF of measured resistance values for various write voltages for (A) set operation and (B) reset operation [15].

each cell is proportional to the product of the programmed conductance (W_{ij}) and applied voltage ($\vec{x}_i$). The resulting currents that are summed along the columns of the crossbar are proportional to the product of the matrix and vector, ($\vec{y}$).

B. Characterizing Device Variation in RRAM

CIM seeks to read and accumulate several states on a bitline at once, and therefore a key obstacle to enabling CIM is cell-to-cell variation. These variations are typically normally distributed [14], [15] and measured as the standard deviation (σ) from the mean (μ) resistance value. Cell-to-cell variation is commonly expressed as a percentage of the mean (σ/μ), and we use this notation for the rest of the paper. For binary (2-level) cells, a digital '0' is encoded as the high resistance state (HRS) and a digital '1' is encoded as the low resistance state (LRS). Ideally, current from reading a cell in the HRS could be ignored if the difference between between the HRS and LRS (on/off ratio) were several orders of magnitude. Unfortunately, this is not the case as recent RRAM [15] and PCM [14] demonstrate an on/off ratio between 10$\times$ to 100$\times$.

Recent demonstrations of RRAM [15] and PCM [14] show low LRS variation (3.5%) and high HRS (50%) variation with an on/off ratio of 10$\times$ to 100$\times$ depending on how the cells are written [14], [15]. Using higher write voltages and an iterative write verify protocol, lower variation and higher on/off ratio can be achieved. To quantify the variation, we measure the resistance values of RRAM cells from a recent RRAM test-chip prototype on a 40nm foundry RRAM array [15]. The array contains 256 $\times$ 256 RRAM cells (64Kb). The details of



Key	●	●	●	●	●	●	●	●	●
# LRS	0	1	2	3	4	5	6	7	8
μV/WL	1.5	5.6	10.7	21.2	26.4	27.3	27.7	28.6	29.1

Fig. 2. CIM (8 WL) samples for various weight encodings. IR-drop slope (μV/WL) is computed using linear regression.

the read and write circuit of the array are beyond the scope of this paper and interested readers are pointed to [15] for further discussions.

To understand the various control parameters available to iterative write verify algorithms, we study the impact of various write configurations. First, we use three different voltages for both the set (LRS) and reset (HRS) operations. For each voltage, we use a 100ns write pulse. The CDF of these measurements as well as the corresponding μ and σ are shown in Figure 1. From this experiment, we find that a higher write voltage can reduce device variation and increase on-off ratio. Despite these observations, we must acknowledge several drawbacks. First, these techniques increase write energy and latency. Second, they greatly reduce the endurance of RRAM and most other eNVM [16].

C. Impact of Device Variation & IR-Drop on CIM

To better understand the impact of device variation (Fig. 1) and IR-drop, we program the test array with a bit pattern to generate various encodings (0-8 LRS). For this experiment, we use a write voltage of 1.7V and pulse width of 100ns. Next, we enable 8 wordlines at a time, sample the results, and plot them in Figure 2. Each encoding is marked by a different color given by the legend above the plot. For each encoding, we compute IR-drop using linear regression on the set of all samples. The resulting regression slope (or coefficient) is presented in the table as μV per WL. As a proxy for distance from the read circuit, we use the WL number of the cell that is read.

From this experiment, we observe two key properties. First, IR-drop decreases linearly with the WL number. As discussed in Section I, the linear decrease occurs because the parasitic wire resistance decreasing linearly along the WL due to distance from the read circuit. In this design, the read circuit is placed below the array itself, and thus WL 255 is closest to the read circuit. An illustration of the array level design is given in Figure 3. We target 2 different cells and approximate the wire resistance on the read path. Second, we find that the effect of IR-drop increases with the encoding (# of LRS states). This is because more LRS states yields a lower resistance on the bitline. When the resistance is lower, the parasitic wire resistance accounts for a larger fraction of the voltage drop on the bitline.

To quantify the impact of device variation and IR-drop, we evaluate the error rate of the measured data. We classify the data in Figure 2 by setting reference voltages for each encoding. We present this result as a confusion matrix in Figure 4A. Each bin shows the percent of actual ADC output codes were obtained for the expected ADC output code. From this result we observe that errors only occur (for this sample size) when the encoding is large ($\geq$7). Unfortunately, this occurs because the effects of both device variation and IR-drop are exacerbated at higher encodings. To isolate the impact of IR-drop, we can subtract the expected voltage drop due to wire resistance. To do this, we use the coefficients computed using linear regression in Figure 2. Next, we re-compute the

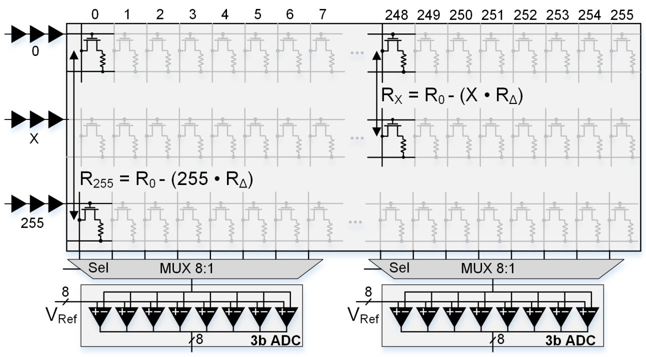
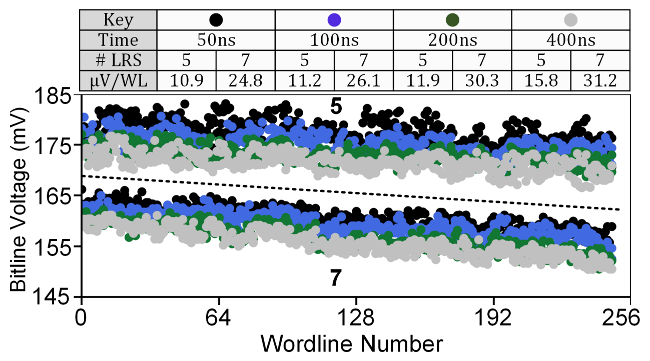


Fig. 3. 256×256 RRAM array architecture. 8 adjacent cells form an 8-bit weight and share a 3-bit ADC through a 8-to-1 multiplexer. Total read resistance is function of distance from bitline and the RRAM cell itself.



Key	●		●		●		●	
Time	50ns		100ns		200ns		400ns	
# LRS	5	7	5	7	5	7	5	7
µV/WL	10.9	24.8	11.2	26.1	11.9	30.3	15.8	31.2

Fig. 5. IR-drop characterization over several programming pulse times. Bitline voltage versus wordline number reveals linear IR-drop relationship.

confusion matrix after removing IR-drop in Figure 4B. We find that error is greatly reduced after removing IR-drop.

III. Mitigating IR-Drop

In the previous section, we identified IR-drop as an issue with significant impact on CIM BER. To reduce the impact of IR-drop at the circuit level, we have a few options. First, we can modify our read logic to account for the expected IR-drop. An example of this is changing the reference voltages used by our ADC to account for IR-drop computed as a function of the WL. While this method works, it requires significant overhead because many on-chip references are required and for each read operation the references must be computed. Instead, we program the RRAM cells such that their resistance offsets the impact of IR-drop. For example, we expect that the cumulative resistance when reading wordline x to be lower than reading wordline 0 given by the following equation:

$$R_x = R_0 - (x \cdot R_\Delta)$$

where R_Δ is the average wire resistance between 2 adjacent wordlines. For this example, we can attempt to set cells on WL x with a lower voltage or shorter pulse width to offset the parasitic wire resistance.

A. Experimental Calibration

To offset the impact of IR-drop during write, we must have various write configurations that result in different target resistances. For this, we collect CIM (8 WL) measurements on our test chip after writing cells with various write voltages, pulse widths, and weight encodings. For each weight encoding, we generate a bit pattern such that all CIM results read that number of LRS cells. For example, if the the target encoding is '4' we could write the repeating pattern: '11001100' to the BL under test. The measurement conditions are as follows:

1) *Set Voltage*: 1.7V, 1.8V, 1.9V, 2.0V
2) *Set Pulse*: 50ns, 100ns, 200ns, 400ns
3) *Encoding*: 0, 1, 2, 3, 4, 5, 6, 7, 8

We perform all combinations of the measurement conditions for a total of 128 experiments. We choose small increments in voltage and large increments in pulse width because prior work [17] has established that voltage typically has larger impact on the final resistance.

In Figures 5 and 6, we plot a subset of this data. In Figure 5, we plot the bitline voltage versus wordline number for the various pulse widths at a fixed write voltage of 1.8V. We choose to show data for both 5 and 7 LRS cells to illustrate how IR-drop varies based on encoding. We observe that by increasing the encoding or the pulse width, IR-drop increases due to decreased RRAM resistance and thus increased voltage drop due to parasitic wire resistance. We show the same experiment in Figure 6 for the various voltages at a fixed pulse width of 200ns. For both these experiments we observe roughly $30\mu V$ change per WL in the worst case. While this may seem negligible, over 256 WLs it becomes a $7.7mV$ increase and accounts for the majority of error we observe at higher encodings.

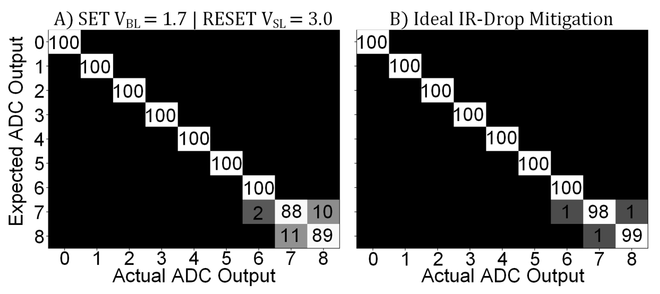
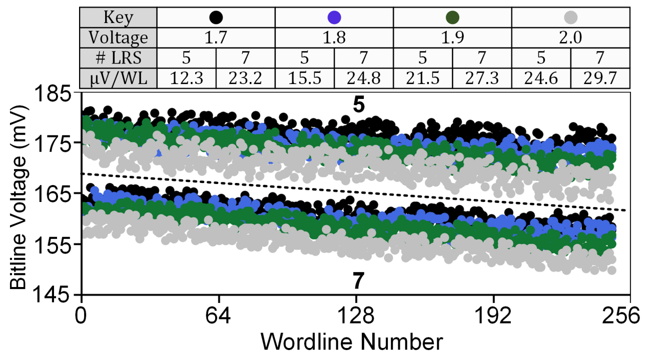


Fig. 4. Confusion matrix between actual and expected ADC outputs during CIM for (A) measured data using 1.7V and (B) without IR-drop.



Key	●		●		●		●	
Voltage	1.7		1.8		1.9		2.0	
# LRS	5	7	5	7	5	7	5	7
µV/WL	12.3	23.2	15.5	24.8	21.5	27.3	24.6	29.7

Fig. 6. IR-drop characterization over several programming voltages. Bitline voltage versus wordline number reveals linear IR-drop relationship.

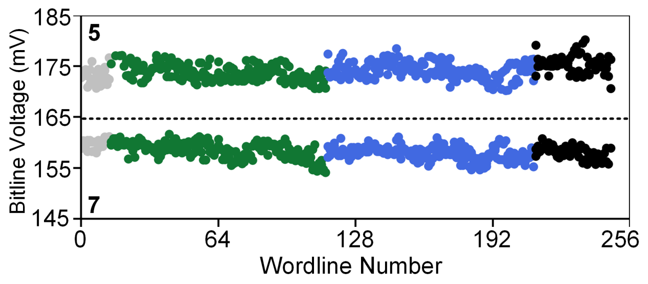


Fig. 7. Optimal write configuration using the 4 configurations in Figure 5.

B. IR-Drop Model and Optimization

Ideally, we could program each WL with a unique write configuration to eliminate IR-drop to implement our experiment from section II. However, this is impractical because it would require far too many voltage or pulse width combinations to implement. Furthermore, it is unlikely that 256 such protocols could be implemented such that the average programmed resistance could offset $10\mu V$ to $30\mu V$ at each WL. Instead, we consider subsets of our 16 calibrated write configurations and assign each WL a write configuration that minimizes error due to IR-drop.

To identify the optimal write configuration for each WL, we first build models for each write configuration and establish an optimization objective. For each write configuration (1.7V / 50ns, 1.8V / 100ns, etc.) at each encoding (0 LRS, 1 LRS, etc.), we use the regression slope to model error due to IR-drop (as in Figures 2, 5, and 6. Our optimization objective is minimizing the mean squared error (MSE) between each sample and the average BL voltage for the encoding it belongs to. We choose to use MSE because it penalizes large deviations from the mean which are likely to result in CIM errors.

After establishing the models and optimization objective, we iterate through the subset of configurations for each WL and compute the MSE. For each write configuration and WL pair, we must also consider the 9 (0-8) possible encodings that can occur. Thus, at each pair we compute the MSE due to IR-drop as:

$$MSE = \sum_{N=0}^{8} (\alpha_N + \beta_N \cdot x - \mu_N)^2$$

where α is the regression intercept (R_0), β is the regression slope (R_Δ), x is the WL, and μ is the mean bitline voltage. After computing the MSE for each write configuration, we simply select the smallest error for the WL and repeat for all 256 WL. In Figure 7, we demonstrate our algorithm on the 4 measurements provided in Figure 5. Each WL is assigned the write configuration that minimizes MSE.

IV. Results

To evaluate the effectiveness of our algorithm for IR-drop mitigation, we compare against a baseline. We evaluate both BER and classification accuracy of ResNet18 on ImageNet. To compute BER we use measured data from the different write configurations we collected. And to compute classification accuracy, we simulate ResNet18 with the given BER. This is done by creating a custom VMM kernel in TensorFlow that randomly inserts errors at the given BER. For this evaluation we use a 8-bit quantized ResNet18 model that achieves 68.3% Top-1 accuracy on ImageNet.

Configuration		Performance	
Voltage	Time	BER	Accuracy Loss (%)
1.7V	ALL	$3.7 \cdot 10^{-4}$	0.4%
1.8V	ALL	$1.2 \cdot 10^{-4}$	0.2%
1.9V	ALL	$8.8 \cdot 10^{-6}$	0.0%
2.0V	ALL	$2.3 \cdot 10^{-5}$	0.0%
ALL	50ns	$1.5 \cdot 10^{-3}$	5.3%
ALL	100ns	$2.3 \cdot 10^{-4}$	0.4%
ALL	200ns	$9.3 \cdot 10^{-5}$	0.2%
ALL	400ns	$7.9 \cdot 10^{-5}$	0.1%
Baseline (1.8V, 200ns)		$1.2 \cdot 10^{-3}$	4.8%

Table 1. BER and accuracy results from various write configurations.

Because our algorithm creates a combined write configuration by assigning one of several write configurations to each WL, we must choose subsets of our 16 measured write configurations as in Figure 7. To evaluate our algorithm, we choose 8 sets of 4 write configurations that are shown in Table 1. For 4 of these configurations we consider all pulse times for a given write voltage. And for the other 4 we consider all write voltages for the various pulse times. We consider write time and write voltage separately because they feature different trade-offs.

After establishing our configurations, we apply our algorithm detailed in Section III, and then compute the BER using the total samples and the total number of errors that occur. Because a large number of samples is required to accurately compute BER and we only have 1024 per configuration (128K total), we construct a distribution for each configuration based on our measured samples. Next, we re-sample the distribution using Monte Carlo simulation to account for outliers ($>3\sigma$) that did not occur in measured data.

In Table 1 we show BER and accuracy results after the process is complete. For the measured data, we find that using various pulse widths yields better results than using various voltages (on average). However, this is likely due to the write voltages that were used and could be improved with a different set of configurations (not 100mV increments). We find that the lowest BER ($8.8{\cdot}10^{-6}$) is achieved for the 1.9V write voltage (using all pulse widths). For reference, the best BER achieved without our technique occurred for the 1.8V/200ns configuration which yielded a BER of $1.2{\cdot}10^{-3}$. Thus in the best case our technique achieves a 136.4$\times$ improvement in BER over the baseline method.

V. Acknowledgement

This work was funded by the U.S. Department of Defense's Multidisciplinary University Research Initiatives (MURI) Program under grant number FOA: N00014-16-R-FO05 and the Semiconductor Research Corporation under the Center for Brain Inspired Computing (C-BRIC).

References

[1] B. Crafton *et al.*, “Merged logic and memory fabrics for accelerating machine learning workloads,” *IEEE Design & Test*, 2020.

[2] V. Sze *et al.*, “Efficient processing of deep neural networks: A tutorial and survey,” *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[3] N. P. Jouppi *et al.*, “In-datacenter performance analysis of a tensor processing unit,” in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, IEEE, 2017.

[4] B. Crafton, S. Spetalnick, J.-H. Yoon, W. Wu, C. Tokunaga, V. De, and A. Raychowdhury, “Cim-secded: A 40nm 64kb compute in-memory rram macro with ecc enabling reliable operation,” in *2021 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 1–3, IEEE, 2021.

[5] B. Feinberg *et al.*, “Making memristive neural network accelerators reliable,” in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 52–65, IEEE, 2018.

[6] R. M. Roth, “Fault-tolerant dot-product engines,” *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2046–2057, 2018.

[7] B. Crafton, S. Spetalnick, J.-H. Yoon, and A. Raychowdhury, “Statistical optimization of compute in-memory performance under device variation,” in *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, IEEE, 2021.

[8] Y. Park *et al.*, “Unlocking wordline-level parallelism for fast inference on rram-based dnn accelerator,” in *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

[9] A. S. Rekhi *et al.*, “Analog/mixed-signal hardware error modeling for deep learning inference,” in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[10] Y. Long *et al.*, “Design of reliable dnn accelerator with un-reliable reram,” in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1769–1774, IEEE, 2019.

[11] S. Lee, G. Jung, M. E. Fouda, J. Lee, A. Eltawil, and F. Kurdahi, “Learning to predict ir drop with effective training for reram-based neural network hardware,” in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[12] M. E. Fouda, S. Lee, J. Lee, G. H. Kim, F. Kurdahi, and A. M. Eltawi, “Ir-qnn framework: An ir drop-aware offline training of quantized crossbar arrays,” *IEEE Access*, vol. 8, pp. 228392–228408, 2020.

[13] X. Sun and S. Yu, “Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 3, pp. 570–579, 2019.

[14] J. Wu *et al.*, “A 40nm low-power logic compatible phase change memory technology,” in *2018 IEEE International Electron Devices Meeting (IEDM)*, pp. 27–6, IEEE, 2018.

[15] J.-H. Yoon *et al.*, “29.1 a 40nm 64kb 56.67 tops/w read-disturb-tolerant compute-in-memory/digital rram macro with active-feedback-based read and in-situ write verification,” in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, pp. 404–406, IEEE, 2021.

[16] C. Nail *et al.*, “Understanding rram endurance, retention and window margin trade-off using experimental results and simulations,” in *2016 IEEE International Electron Devices Meeting (IEDM)*, pp. 4–5, IEEE, 2016.

[17] L. Gao, P.-Y. Chen, and S. Yu, “Programming protocol optimization for analog weight tuning in resistive memories,” *IEEE Electron Device Letters*, vol. 36, no. 11, pp. 1157–1159, 2015.