Self-Terminating Write of Multi-Level Cell ReRAM for Efficient Neuromorphic Computing

Zongwu Wang[1], Zhezhi He[1†], Rui Yang[1], Shiquan Fan[2], Jie Lin[3], Fangxin Liu[1,4], Yueyang Jia[1], Chenxi Yuan[2]
Qidong Tang[1], Li Jiang[1,4,5†]
[1]Shanghai Jiao Tong University, [2]Xi'an Jiao Tong University, [3]University of Central Florida
[4]Shanghai Qi Zhi Institute, [5]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

***Abstract*—The Resistive Random-Access-Memory (ReRAM) in crossbar structure has shown great potential in accelerating the vector-matrix multiplication, owing to the fascinating computing complexity reduction (from $O(n^2)$ to $O(1)$). Nevertheless, the ReRAM cells still encounter device programming variation and resistance drifting during computation (known as read disturbance), which significantly hamper its analog computing precision. Inspired by prior precise memory programming works, we propose a Self-Terminating Write (STW) circuit for Multi-Level Cell (MLC) ReRAM. In order to minimize the area overhead, the design heavily reuses inherent computing peripherals (e.g., Analog-to-Digital Converter and Trans-Impedance Amplifier) in conventional dot-product engine. Thanks to the fast and precise programming capability of our design, the ReRAM cell can possess 4 linear distributed conductance levels, with minimum latency used for intermediate resistance refreshing. Our comprehensive cross-layer (device/circuit/architecture) simulation indicates that the proposed MLC STW scheme can effectively obtain 2-bit precision via a single programming pulse. Besides, our design outperforms the prior write&verify schemes by $4.7\times$ and $2\times$ in programming latency and energy, respectively.**

***Index Terms*—ReRAM, Programming, Multi-Level Cell, Read Disturbance**

I. Introduction

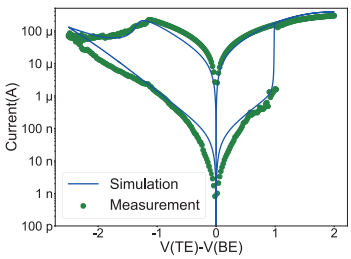
The Resistive Random-Access-Memory (ReRAM) is a two-terminal device with programmable resistance, ranging from Low-Resistance State (LRS) to High-Resistance State (HRS), for information storage [1]. It has attracted tremendous interest in the past decade from both academia and industry, owing to its outstanding device properties, including high density, non-volatility, large ON/OFF ratio, etc [2], [3]. Besides its superior characteristics as the main memory alternative, the ReRAM cells wired in the crossbar structure [4], [5] have been widely adopted as the dot-product engine since its in-situ current-weighted summation operation intrinsically conducts Vector-Matrix Multiplication (VMM). Such ReRAM crossbar design can significantly simplify the computation complexity of VMM from $O(n^2)$ to $O(1)$, thus bringing great opportunities in accelerating VMM-intensive applications, e.g., the neural network inference especially. Meanwhile, due to the process variation and the stochastic switching mechanism, the programmable resistance range and programming sensitivity differ from cycle to cycle and device to device. Generally, such write variation brings a great challenge for both the conventional memory function and VMM acceleration. When using ReRAM for data storage, the write variation can significantly lower the sense margin (between LRS and HRS), thus leading to erroneous read results. Such a sense margin is further compromised when using the ReRAM crossbar for VMM acceleration, as multiple ReRAM cells are in the sensing (i.e., computation) path simultaneously.

As the countermeasure to the ReRAM write-variation, numerous approaches have been proposed from the device- [6], circuit- [7], or algorithm-level [4], which are summarized as follows: **Device-level**, although the device experts are devoted to fundamentally address the aforementioned ReRAM programming variation from the device physics perspective, due to the immature of ReRAM technology and the ReRAM switching mechanism remains uncertain [6], such objective can hardly be achieved within a short time. **Circuit-level**, prior works have brought up various circuit-level countermeasures [7]–[11]. However, all those prior designs partially fulfill the ReRAM crossbar of VMM acceleration requirements in terms of the ReRAM set (i.e., LRS) precision, and area efficiency, but the *fast programming of MLC (Multi-Level Cell) ReRAM is still absent*. **Algorithm-level**, the most widely adopted algorithm-level technique is to incorporate the variation within the model training process [12]. It leverages the fault tolerance capability of the neural network. However, the time-consuming training and accompanied accuracy degradation make such a solution infeasible in practice.

To summarize the discussion above, as a detour of mitigating the ReRAM write variation, the circuit-level is the most promising and practical strategy. *In this work, we propose a Self-Terminating Write (STW) scheme to enable MLC ReRAM crossbar programming with high precision and low latency, for variation efficient VMM acceleration.* Our contributions in this work are enumerated as:

- To our best knowledge, we are the first to propose a valid self-terminating write circuit for MLC ReRAM, perform ReRAM crossbar set and reset in a fast and precise fashion, and the compact model in [13] is calibrated for better characterizing the non-ideal ReRAM variation.
- Our proposed STW circuit heavily reuses the original ReRAM peripheral and shares the circuit for set and reset

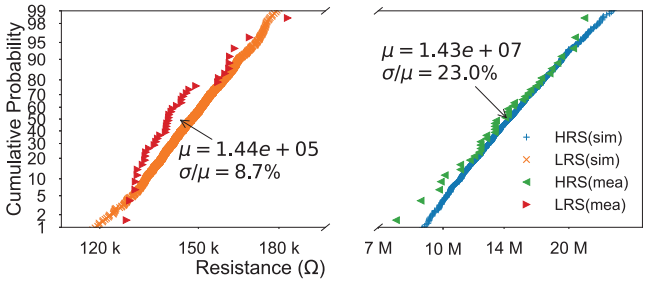
Li Jiang and Zhezhi He are the corresponding authors.



(a) Parameters fitting of ReRAM compact model w.r.t DC programming experiment results.

Parameter	Unit	Value SET	Value RESET
g_0	nm	0.23	
V_0	V	0.364	
ν_0	m/s	15.17	
I_0	mA	0.34	
α	1	1.124	
β	1	3.208	5.396
γ_0	1	20.792	21.075
E_A	1	0.918	
δ_g^0	-	1.2e-2	
g_{min}*	nm	0.6	
g_{max}**	nm	2	

(b) Fitting Coefficients for ReRAM compact model.



(c) Normal probability paper: variation-related parameters fitting of ReRAM compact model with pulse programming experiment results.

Fig. 1: Compact ReRAM modeling w.r.t our experimental measurement. (a) DC programming curve fitting, (b) Fitting coefficients. (c) Pulse programming curve fitting.

to minimize area overhead. In comparison to prior STW designs, our proposed method achieves 2-bit precision with similar area utilization, while the prior designs [7], [10], [11] can only perform STW for SLC (Single-Level Cell) ReRAM. Compared to the write&verify design [14], [15] on MLC ReRAM, our design reduces the programming latency and energy by $4.7\times$ and $2\times$, respectively.

- We conduct cross-layer simulation (device/circuit/system) to validate our design. With neural networks of VGG and ResNet inference on ReRAM-based accelerator, the impact of the read disturbance on system performance is evaluated to enlighten further studies.

II. Preliminary

A. ReRAM and ReRAM Crossbar

1) ReRAM Device Modeling and Variation: By far, in terms of the mechanism of ReRAM resistance switching, the formation and rupture of the conductive filament that consists of oxygen vacancies are the most well-accepted theory by the community [13], [16]. To consider the intrinsic switching variability of ReRAM (i.e., programming variation) and to fit the I-V characteristics of our fabricated ReRAM cell, we use the popular filament model introduced in [13] for analysis. Through the curve fitting process w.r.t the experimental data (I-V curve as shown in Fig. 1a), the acquired fitting coefficients are listed in Fig. 1b. The notations are identical as used in [13], which will not be specified here due to the space limit. The setting of other unlisted parameters is identical as in [13].

One of the most critical challenges of MLC ReRAM that can deteriorate the sensing margin between the two adjacent

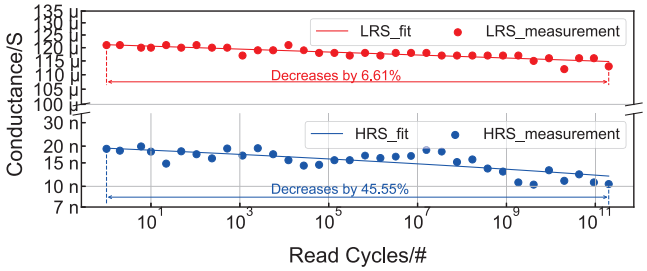


Fig. 2: Read disturbance effect is investigated by fitting the relationship between conductance shifting and read cycles, the measurement data origins from [18].

resistance levels is the Cycle-to-Cycle (C2C) switching variability. The C2C variation, known as temporal non-uniformity, can result in random switching speed during each ReRAM programming cycle. For modeling the temporal non-uniformity, the C2C fluctuation is characterized by the parameter δ_g^0 in Fig. 1b, which correlates to the amplitude of the additive random disturbance on filament formation and rupture speed. Setting $\delta_g^0 = 1.2e{-}2$, the 10^4 trials Monte Carlo simulation is shown in Fig. 1c indicates both the LRS and HRS resistance distribution fit well w.r.t our measured data reported in [17].

2) ReRAM Device Read Disturbance Modeling: In addition to the variations mentioned above, we consider the read disturbance issue. Read disturbance refers to the memory states change after a high number of the read operations, thus the information stored is unintentionally tampered [19]. Read disturbance effect is a key contributor to the non-volatile memory reliability issues in ReRAM [20]. According to [20], even read verification during the write procedure can disturb the ReRAM conductance, thus triples the programming time. As shown in Fig. 2, we modeled the effect of read disturbance by fitting the measurement data in [18] via the least-squares method, and the result indicates that the HRS is more vulnerable to read disturbance which coincides with [21].

B. Related Works and Motivation

A countermeasure against the ReRAM programming variation summarized in Section I, due to the scope of this work, we will focus on circuit-level solutions. Prior circuit-level solutions can be categorized into twofold: write&verify (w&v) and self-terminating write, which are specified hereafter.

1) Write&Verify Method: The w&v method [8], [9] iteratively performs the ReRAM write with a high-frequency voltage pulse followed by verifying the programmed ReRAM resistance, until the target ReRAM cell reaches the target value. The drawbacks of such w&v method can be enumerated as: 1) multiple cycles are required, which incurs high programming latency (e.g., [9] shows programming a VGG-16 network with 138M weights takes 46mins); 2) high hardware cost for conducting parallel ReRAM write.

2) Self-Terminating Write: Unlike the w&v methods utilizing multiple cycles with programming pulses, the STW method only applies a single wide programming pulse. Four STW circuits are chosen here as the competing designs, which are developed by [10], [7] and [11] respectively, where [10] is

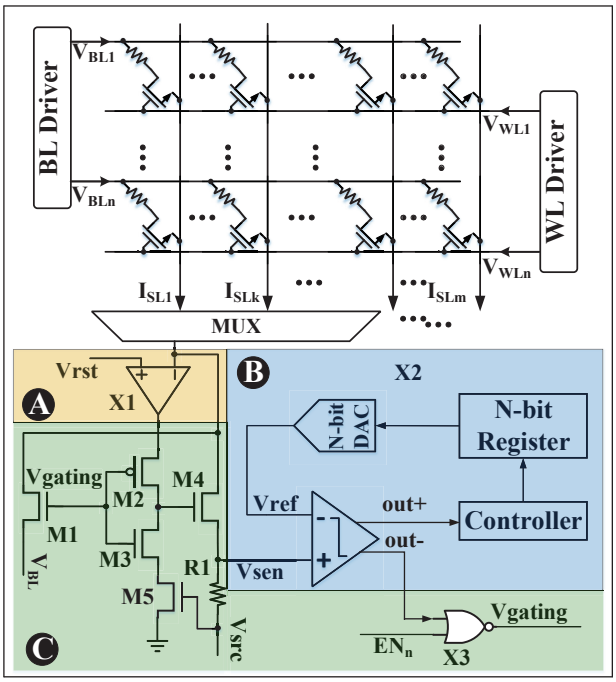


Fig. 3: ReRAM-based dot product accelerator embedded with STW scheme. The overall architecture consists of three parts: (1) The ReRAM cell array with white background. (2) The peripheral modules for dot product acceleration with khaki and blue background. (3) The additive module for STW implementation with green background.

the first work that develops the STW on SLC ReRAM. It uses the current mirror for programming current monitoring, a verdict module receiving the feedback signal, then cut-off the bias voltage once target resistance is reached. This design owns two main drawbacks: 1) the current mirror failed to copy the programming current for sensing accurately; 2) the design introduces a large area overhead due to two additional comparators and a verdict module. Other designs proposed in [11] replace the comparator with an inverter for area efficiency. However, these are ineffective designs due to : 1) the Process/Voltage/Temperature (PVT) variation can easily affect the inverter switching threshold, making it imprecise for the application. 2) the D Flip-Flop is used in these designs, which is an area-hungry unit. 3) the set and reset termination can not share the same circuit in these designs. In contrast to the current-mode schemes in [10], the voltage mode design proposed in [7], [22] significantly simply the sensing and termination circuitry for set operation, with only utilizing several transistors and an inverter. For the reset operation, [7] uses a comparator together with another swing detector. Overall, [7] is still not robust to PVT when performing the set operation within one cycle, and the reset operation can not share the simple circuit of the set operation.

III. Parallel Self-Terminating Memory Write

A. Overview of Self-terminating Write Architecture

In this section, a novel parallel STW scheme embedded into the ReRAM-based dot product accelerator is introduced.

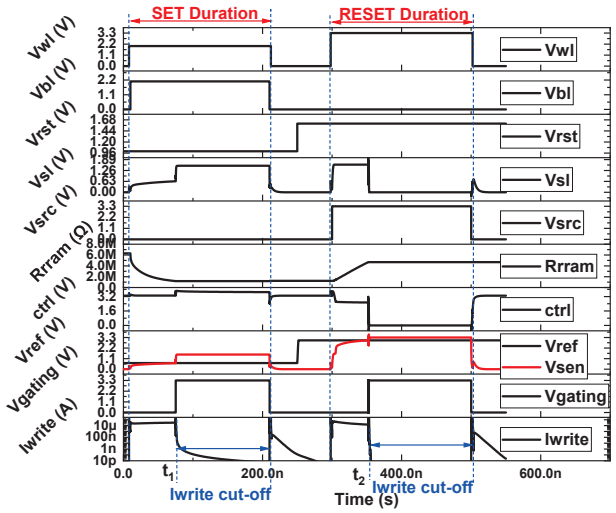


Fig. 4: Transient waveform with self-terminating function for both set and reset operation.

It turns off the single writing pulse automatically once the targeting resistance value is programmed, thus the power consumption and the write stress issues can be effectively suppressed. The overall architecture of the proposed STW scheme consists of three parts, and is shown in Fig. 3. **Part one:** the ReRAM cell array is shown with white background, and the aggregate current on each SL is elected with a multiplexer. **Part two:** the peripheral modules for dot product acceleration, consist of a TIA and a SAR-ADC. The TIA shown in Fig. 3Ⓐ with khaki background converts the SL current to voltage linearly, and the SAR-ADC in Fig. 3Ⓑ with blue background quantifies the corresponding voltage. **Part three:** the additive module for embedding the STW scheme in the dot product accelerator, shown in Fig. 3Ⓒ with green background.

B. Operating Mode

The proposed STW scheme is embedded in conventional accelerate architecture seamlessly, thus it can switch between three modes with a friendly interface protocol.

1) Conventional inference mode: The STW scheme of this work is embedded into the conventional dot product accelerator seamlessly. The EN_{n} is an enable signal for STW mode, which is active low. Thus, EN_{n} keeps high in inference mode, forcing V_{gating} low to cut off the M1 and M3, and the M2 and M4 connect the TIA to SAR-ADC.

2) Reset termination implementation: During STW mode, the EN_{n} is low. The ReRAM under reset is activated with $V_{\mathrm{BL}} = 0V$, and V_{rst} is applied on the non-inverting terminal of X1. In the beginning, the ReRAM is in LRS, and the peripheral devices dominate the voltage drop. It means V_{SL} is lower than V_{rst}, thus the amplifier X1 outputs a high voltage. Meanwhile, V_{sen} is lower than V_{ref} , which causes V_{gating} to hold low for switching on M2 and switching off M1 and M3, thus the SL voltage can be clamped at V_{rst}. As the ReRAM resistance increasing, V_{sen} increases simultaneously. Once V_{sen} reaches the level of V_{ref}, V_{gating} switches to high to turn off M2 but turn on M1 and M3. The gate of M4 discharges through M3 and M5 to floating SL, and terminates the reset procedure.

TABLE I: Comparison between Proposed STW Methods (Amp: Amplifier)

	Structure	Area	Terminate	Precision
This work	**2Amp+5T+NOR**	**Medium**	**both**	**2 bits**
JSSC-2013 [10]	2Amp+R+30T +DelayUnit+others	Large	both	1 bit
ISSCC-2014 [22]	4T	Small	set	1 bit
IEDM-2017 [7]	RESET: Amp+4SW+6T SET: 5T	Medium	both	1 bit
ISSCC-2021 [23]	2Amp+R+5T+3INV +AND+Delay Unit	Large	set	1 bit

3) Set termination implementation: The set termination is implemented similar to the reset operation with the $EN_{\rm n}$ low. The ReRAM under set is activated with a high $V_{\rm BL}$ and $V_{\rm src} = 0V$, a proper $V_{\rm rst}$ is applied on the non-inverting terminal of X1. In the beginning, the ReRAM cells under set in HRS dominate the voltage drop. The $V_{\rm gating}$ is low to turn off M1 and M3, and the M2 is turned on. As the resistance of ReRAM cells decreasing, the $V_{\rm gating}$ flips eventually, and turns on the M1 to pull up the SL to terminate the set procedure. Fig. 4 shows the transient curve of self-terminating for both set and reset. Although the set- or reset-voltage is sustained for 200ns, once the $V_{\rm sen}$ reaches the reference voltage $V_{\rm ref}$, $V_{\rm gating}$ is triggered to terminate the programming procedure, and the write current is cut off simultaneously at t_1 for the set operation and t_2 for the reset operation.

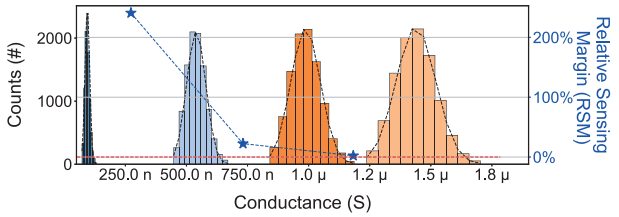
C. STW Performance Analysis

In this part, the self-terminating precision is analyzed, and an in-depth comparison of this work and prior STW schemes [7], [10], [22], [23] is conducted in Table I. The first work for STW implementation is proposed in [10], and it suffers from large area cost and imprecise programming. The feedback loop of this work is complicate, which not only induces area cost, but also contributes to imprecise writing. In our scheme, the area is optimized up to $5\times$ reduction. Furthermore, the simpler implementation boosts the feedback speed by $2\times$, which is very helpful to improve programming accuracy. Besides that, the current mirror in [10] also deteriorates the feedback precision. The scheme in [22] dominates the area cost over the designs in Table I, but only the set operation is under rough control. The design in [7] is similar to [22], whose area cost is comparable with our work, but the set and reset self-terminating function can not be achieved via sharing the same circuit, and the current mirror also deteriorates the performance. The STW scheme proposed in [23] is similar to the design in [10], and the large area cost and imprecise termination also exist. In summary, the self-terminating write scheme proposed in this work conducts much precise programming control with the similar hardware utilization of the state-of-the-art STW schemes. Moreover, the heavy circuit reuse further saves the hardware cost (9T), thus higher precision can be achieved with lower cost.

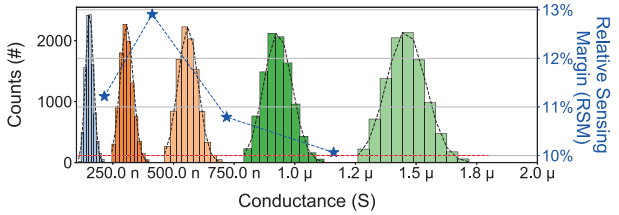
IV. EVALUATION

A. Experimental Setup

We utilize multiple EDA tools to conduct the device/circuit/architecture cross-layer simulation in this work. At the device level, we utilize and modify the compact model from [13] with coefficient fitted w.r.t our experimentally measured ReRAM data, as discussed in Section II-A. Besides, a commercial 65nm CMOS library is used. The circuit-level simulations are conducted by Spice. For architecture-level simulation and analysis, we leverage an in-house simulation framework with partial data extracted from NeuroSim [24].



(a) Histogram of 4 programmed conductance levels of $\Delta g = 460nS$ in the setting of "no margin with linear ΔG_i".



(b) Histogram of 5 programmed conductance levels in the setting of "10% margin with non-linear ΔG_i".

Fig. 5: Maximum number of conductance levels can be achieved through search process with different search setting, for MLC ReRAM using our proposed STW design.

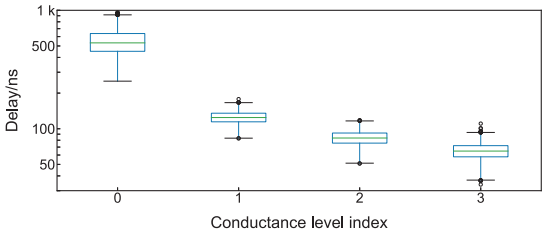
B. Programming Precision Evaluation

To examine the programming precision of the proposed circuit design, we adopt the Relative Sensing Margin (RSM) as the evaluation metric. The RSM statistically measure the margin between two adjacent programmed conductance levels (e.g., level i and $i+1$), which can be expressed as:

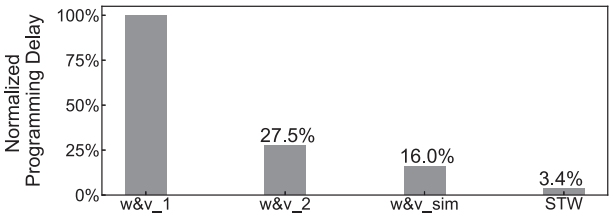
$$\text{RSM} = \frac{(\mu_{i+1} - 3\sigma_{i+1}) - (\mu_i + 3\sigma_i)}{(\mu_i + 3\sigma_i)} \times 100\% \qquad (1)$$

where the μ_i and σ_i represent the conductance mean and variance of level i, respectively. Suppose the ReRAM programming resolution is described by the conductance difference between the adjacent levels ($\Delta G_i = |\mu_{i+1} - \mu_i|$), choosing different programming resolution leads to the different utilization of entire ReRAM conductance range ($G_{\min} = 100nS$ to $G_{\max} = 1.45\mu S$), thus providing different number of conductance levels ReRAM can maximally represent.

To identify the maximum number of non-overlap conductance levels, we perform a searching process to set $\{\Delta G_i\}$. Note that, during the searching, we force the RSM (Eq. (1)) to satisfy either one of the two margin conditions: 1) no margin (RSM=0%) and 2) 10% margin (RSM=10%). Besides, we consider two additional searching configurations: 1) with linear ΔG_i (i.e., $\mu_{i+1} - \mu_i = \Delta g, \forall i$), the searching will halt once the margin condition is not satisfied; 2) with nonlinear ΔG_i (i.e., $\mu_{i+1} - \mu_i = n \cdot \Delta g, \forall i$ and $n \in \{1, 2, ...\}$ is integer), the margin between two levels will be enlarged to satisfy the margin condition. We adopt linear conductance distribution



(a) Effective reset latency (ranges from stimulus start to reset terminate) distribution of 4 conductance levels in Fig. 5a.



(b) Programming delay comparison of prior writ&verify schemes (reported by w&v_1 [14], w&v_2 [15]), w&v_sim (reproduced result of [15] on our device) and our STW scheme on MLC ReRAM.

Fig. 6: Programming delay evaluation results of proposed self-terminating write scheme.

with no margin in neuromorphic computing because of its fault tolerance, and data storage needs to distinguish different states, the non-linear conductance distribution with a 10% margin scheme is suitable for it. Adopting the aforementioned two margin conditions and two searching configurations in the searching, we examine two settings and the results are shown in Fig. 5. In each setting, we scan the programming resolution (Δg) from 20nS to 600nS with a step size of 20ns. The Monte Carlo simulation with 10^4 trials per conductance level is conducted, taken the ReRAM variation using configuration tabulated in Fig. 1b and CMOS mismatch into consideration. Using MLC ReRAM for neuromorphic computing, no margin with linear ΔG_i result shown in Fig. 5a indicates 4 conductance levels ($\sim$ 2-bits MLC) at maximum. Using MLC ReRAM as the conventional data storage function, 10% margin with non-linear ΔG_i is the best candidate. As shown in Fig. 5b, 5 conductance levels can achieve as well. Furthermore, the RSM between the adjacent levels is also evaluated in Fig. 5 on the right y-axis.

C. Latency Evaluation

Since the storage usage is not the focus of this work, only the programming delay corresponds to the conductance levels in Fig. 5a is evaluated. Fig. 6a shows the reset latency distribution with programming to different levels, we can find that $1\mu s$ is long enough for this design. The attempt to shorten the programming pulse width will potentially cause programming failure for conductance level indexed by 1 in Fig. 6a. Moreover, as shown in Fig. 6b, the programming delay of the proposed STW scheme is compared with two write&verify schemes reported by [14], [15]. The proposed self-terminating write method can achieve up to $29\times$ programming speed-up on MLC ReRAM. Two factors severely hinder the speed of write&verify programming: 1) The long setup time in both the programming and verifying procedure. The setup time is proportional to the RC of the array, the parasitic capacitance of a ReRAM array is non-trivial, especially the read verification. 2) Due to the small read current (i.e., current used for computation), it takes a long time to form the stable read signal for the high resistance state. For a fair comparison, the write&verify delay of [14] is also evaluated based on our ReRAM model (w&v_sim in Fig. 6b), whose delay is only 16% compared to [14], since 1) The intrinsic switch speed is different in these works; 2) The overshoot is not taking into consideration in our write&verify simulation. Nevertheless, the proposed STW scheme still outperforms the simulated w&v scheme by 4.7x.

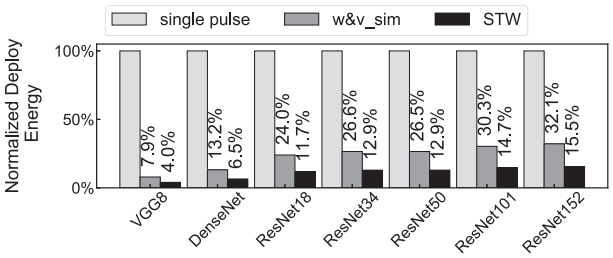


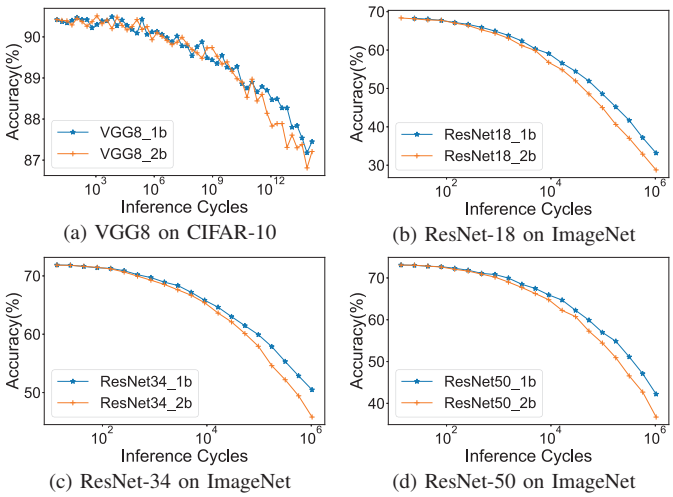
Fig. 7: Energy Consumption Comparison of single pulse programming scheme without STW, w&v_sim (reproduced result of [15]) and our STW scheme with deploying different DNNs.

D. Energy Evaluation

According to [10], the energy consumption of writing a bit can be estimated by $E \approx \int_{T_{\text{prog}}} I_{\text{write}} \cdot V_{\text{write}}\ dt$, where T_{prog} is the sustaining time of the applied programming voltage, and the I_{write} and V_{write} are the current through the ReRAM cells and voltage on 1T1R cells, respectively. Based on that, Fig. 7 shows the total energy consumption of mapping the weights of different DNN on MLC ReRAM arrays with single pulse programming scheme without STW, w&v_sim (reproduced result of [15]) and our STW scheme, the average energy consumption decreases to 23% and 11.2% by adopting write&verify scheme and proposed STW scheme respectively, and up to $12\times$ and $25\times$ energy saving with VGG8 deployed.

E. Read Disturbance Study

As introduced in Section II-A, read disturbance is a serious issue in practical applications, where the frequent inference operation can significantly degrade the inference accuracy of deployed DNN on the ReRAM crossbar accelerator. To investigate such read disturbance from the system perspective, we leverage the modified NeuroSim for system performance evaluation, where we configure the bit-width of inputs and weights to 8, crossbar size is 128-by-128. Since the configurable 1- to 2-bit cell precision can be achieved as shown in Fig. 5a, the system simulation with different cell precision is conducted. Take the worst case into consideration, the largest shift trend in Fig. 2 is used in all levels. And the read disturbance effect on the inference accuracy of different neural networks is shown in Fig. 8. From Fig. 8, we find that the ResNet models are vulnerable to read disturbance. After 1e3 inference cycle, the ResNet models show obvious accuracy drops, while VGG8 model has a strong immunity to conductance shifting. Thus the weights need to be refreshed just like DRAM once the obvious accuracy loss occurs.



(a) VGG8 on CIFAR-10 (b) ResNet-18 on ImageNet

(c) ResNet-34 on ImageNet (d) ResNet-50 on ImageNet

Fig. 8: The impact of read disturbance on the ReRAM-based accelerator's performance.

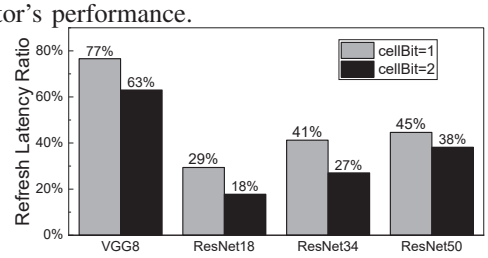


Fig. 9: Refresh latency ratio evaluation for restoring weight after every 1000 inference cycles.

In addition to such degraded accuracy caused by read disturbance versus the number of reading cycles, We also investigate the ratio of refresh latency. In a DNN accelerator system, we want the refresh latency ratio as low as possible, since the portion taken by the refresh phase can be viewed as idle state (i.e., the system is not serving). However, lacking timely refresh will lead to accuracy degradation. Thus, a trade-off is needed to balance the inference accuracy and ratio of a system idle for ReRAM refresh. We evaluate the latency ratio of the refresh phase after every 1e3 inference cycles based on that, and the result is depicted in Fig. 9. With our STW circuit, to maintain the accuracy via taking a refresh per 10^3 inferences, the refresh procedures only occupy 18% to 38% for latency (VGG8 on CIFAR-10 do not need to refresh for every 1000 inference cycles), which makes great progress compared with w&v scheme.

V. Conclusion

In this work, we investigate and develop a Self-Terminating Write scheme for MLC ReRAM in a fast, accurate, and energy-efficient fashion. As far as we know, it is the first STW design for MLC ReRAM. The comprehensive cross-layer (device/circuit/system) simulation is conducted to demonstrate the necessity and superior performance of our design. We want to highlight that such fast and precise resistive memory cell programming is extremely critical to a variety of resistive memory-based neuromorphic computing designs, such as crossbar dot-product engine [5]. It may also benefit other in-memory computing designs, e.g., bit-line computing-based logic-in-memory.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (Grant No. 61834006, 62102257), National Key Research and Development Program of China (2018YFB1403400).

References

[1] F. Liu *et al.*, "Im3a: Boosting deep neural network efficiency via in-memory addressing-assisted acceleration," in *GLSVLSI*, 2021.
[2] H. Akinaga and H. Shima, "Resistive random access memory (reram) based on metal oxides," *Proceedings of the IEEE*, 2010.
[3] F. Liu *et al.*, "Sme: Reram-based sparse-multiplication-engine to squeeze-out bit sparsity of neural network," in *ICCD*, 2021.
[4] Z. He *et al.*, "Noise injection adaption: End-to-end reram crossbar non-ideal effect adaption for neural network mapping," in *DAC*, 2019.
[5] F. Liu *et al.*, "Bit-transformer: Transforming bit-level sparsity into higher preformance in reram-based accelerator," in *ICCAD*, 2021.
[6] S. Yu, X. Guan, and H.-S. P. Wong, "On the switching parameter variation of metal oxide rram—part ii: Model corroboration and device design strategy," *TED*, 2012.
[7] W. Chen *et al.*, "A 16mb dual-mode reram macro with sub-14ns computing-in-memory and memory functions enabled by self-write termination scheme," in *IEDM*.
[8] E. J. Merced-Grafals *et al.*, "Repeatable, accurate, and high speed multi-level programming of memristor 1t1r arrays for power efficient analog computing applications," *Nanotechnology*, 2016.
[9] J. Chen *et al.*, "A parallel multibit programing scheme with high precision for rram-based neuromorphic systems," *TED*, 2020.
[10] X. Xue *et al.*, "A 0.13 μm 8 mb logic-based cu $\_x$ si $\_y$ o reram with self-adaptive operation for yield enhancement and power reduction," *IEEE Journal of solid-state circuits*, 2013.
[11] M. Alayan *et al.*, "Switching event detection and self-termination programming circuit for energy efficient reram memory arrays," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019.
[12] B. Zhang *et al.*, "Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations," in *ICASSP*, 2019.
[13] P.-Y. Chen and S. Yu, "Compact modeling of rram devices and its applications in 1t1r and 1s1r array design," *TED*, 2015.
[14] L. Zhao *et al.*, "Multi-level control of conductive nano-filament evolution in hfo2 reram by pulse-train operations," *Nanoscale*, 2014.
[15] L. Gao, P. Chen, and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," *IEEE Electron Device Letters*, 2015.
[16] X. Guan, S. Yu, and H.-S. P. Wong, "On the switching parameter variation of metal-oxide rram—part i: Physical modeling and simulation methodology," *IEEE Transactions on electron devices*, 2012.
[17] R. Yang *et al.*, "2d molybdenum disulfide (mos2) transistors driving rrams with 1t1r configuration," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017.
[18] R. Yang *et al.*, "Ternary content-addressable memory with mos2 transistors for massively parallel data search," *Nature Electronics*, 2019.
[19] Y. Cai *et al.*, "Read disturb errors in mlc nand flash memory: Characterization, mitigation, and recovery," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015.
[20] T. O. Iwasaki, S. Ning, and K. Takeuchi, "Forward and reverse biasing in resistive memories for fast, disturb-free read, and verify," *Japanese Journal of Applied Physics*, 2013.
[21] H. Lv *et al.*, "Beol based rram with one extra-mask for low cost, highly reliable embedded application in 28 nm node and beyond," in *IEDM*, 2017.
[22] M.-F. Chang *et al.*, "19.4 embedded 1mb reram in 28nm cmos with 0.27-to-1v read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme," in *ISSCC*, 2014.
[23] J. Yang *et al.*, "24.2 a 14nm-finfet 1mb embedded 1t1r rram with a 0.022µm2 cell size using self-adaptive delayed termination and multi-cell reference," in *ISSCC*, 2021.
[24] P. Chen *et al.*, "Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *TCAD*, 2018.