Read-disturb Detection Methodology for RRAM-based Computation-in-Memory Architecture

Yaldagard, M.A.; Diware, S.; Joshi, R.V.; Hamdioui, S.; Bishnoi, R.

DOI
10.1109/AICAS57966.2023.10168638

Publication date
2023

Document Version
Final published version

Published in
2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)

2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS) | 979-8-3503-3267-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/AICAS57966.2023.10168638

Read-disturb Detection Methodology for RRAM-based Computation-in-Memory Architecture

Mohammad Amin Yaldagard* Sumit Diware* Rajiv V. Joshi† Said Hamdioui* Rajendra Bishnoi*
*Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands.
Email: {M.A.Yaldagard, S.S.Diware, S.Hamdioui, R.K.Bishnoi}@tudelft.nl
†IBM Research Division, Yorktown Heights, NY, USA. Email: rvjoshi@us.ibm.com

***Abstract*—Resistive random access memory (RRAM) based computation-in-memory (CIM) architectures can meet the unprecedented energy efficiency requirements to execute AI algorithms directly on edge devices. However, the read-disturb problem associated with these architectures can lead to accumulated computational errors. To achieve the necessary level of computational accuracy, after a specific number of read cycles, these devices must undergo a reprogramming process which is a static approach and needs a large counter. This paper proposes a circuit-level RRAM read-disturb detection technique by monitoring real-time conductance drifts of RRAM devices, which initiate the reprogramming when actually it needs. Moreover, an analytic method is presented to determine the minimum conductance detection requirements, and our proposed read-disturb detection technique is tuned for the same to detect it dynamically. SPICE simulation result using TSMC 40 nm shows the correct functionality of our proposed detection technique.**

I. Introduction

Edge AI refers to algorithms and models implemented on devices such as smartphones, cameras, and sensors, rather than relying on cloud computing or remote servers. There have been many ASIC designs developed to implement deep neural networks (DNN) with a large number of multiply-and-accumulate (MAC) operations. By virtue of the massive data movement between computing and storage units, Von Neumann architecture approaches are energy-inefficient for real-time inference due to the significant power consumption and high latency [1]–[4]. The concept of computation-in-memory (CIM) involves performing computations directly within the memory, which can improve the energy efficiency as well as the performance of DNNs [5]–[7]. Resistive random access memory (RRAM) based CIM is promising due to its many advantageous features such as practically zero leakage, less access power consumption, high density, and massive parallelism [6], [7]. However, RRAM devices are subject to conductance drift over time during the read operations, leading to bit flips known as read-disturb. Any drifts in the conductance (or resistance) of the RRAM may result in computational errors due to the disruption of analog MAC values [8], [9].

Several approaches have been proposed to solve the read-disturb problem in RRAM-based CIM architecture. For instance, employing a low read voltage can reduce the impact of conductance drifts [10], [11]. Nevertheless, the voltage reduction can also increase the rate of incorrect computations, especially in the presence of process variation. Additionally, an architectural solution is proposed to improve the sensing margin that degrades the computational error due to the read-disturb [12]. However, these aforementioned solutions can just delay read-disturb occurrences and thus are not concrete solutions for the read-disturb. To address this problem, a periodic reprogram-based solution is proposed in which RRAM devices are rewritten after a certain number of read cycles [13]. Furthermore, a technique is proposed in which the read current directions can be periodically reversed to compensate for the drift effect [14], [15]. Both of these techniques use static approaches designed to consider the worst-case scenario for a correct computation, which makes the overall design too pessimistic. Additionally, a large counter is necessary to count the acceptable read cycles to reprogram periodically. Hence, there is a decisive need for a low-cost and process-independent read-disturb detection mechanism that can detect its occurrence dynamically in an efficient manner.

In this paper, we propose a methodology for RRAM read-disturb detection based on a real-time drift monitoring circuit. To introduce a realistic conductance drift scenario, the required resistance difference can be developed using a selector logic by activating/deactivating multiple RRAM devices, which can be compared using a sense amplifier. Our proposed detection methodology utilizes the same RRAM configuration as in a CIM array to detect drift effectively, considering variations in process, voltage, and temperature (PVT). Overall, the contributions to this paper are as follows:

- Develop a dynamic read-disturb detection methodology.
- Implement the circuit of the proposed read-disturb detection methodology.
- Present an analytic method to determine the minimum conductance states ratio to meet the accurate computation requirements.

SPICE simulation results using TSMC 40 nm verify the functionality of the proposed technique. Our proposed method has the capability to detect read-disturb with proximity to its occurrence, thus reducing the frequency of reprogramming cycles. The proposed technique results in a 2x improvement

This work is supported by the TU Delft AI labs program and the EU H2020 grant "DAIS" that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No. 101007273.

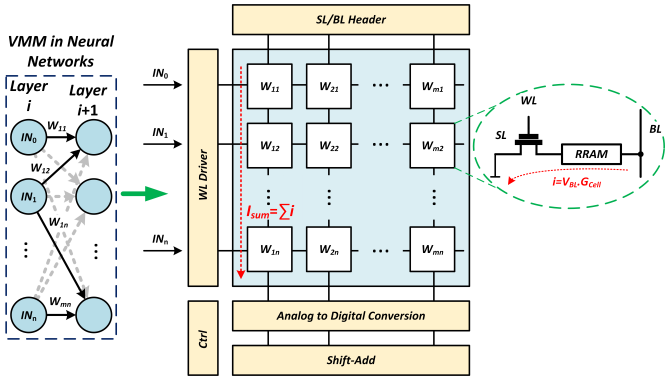


Fig. 1. A generic RRAM-based CIM architecture for neural networks.

in energy efficiency compared to the conventional periodic reprogramming approach.

The rest of the paper is organized as follows. A background on RRAM-based CIM and RRAM read-disturb is discussed in Section II. Section III presents proposed ideas, including the detection circuit and analysis. Section IV describes the simulation setups and results to verify the proposed circuit performance. The last section concludes the paper.

II. Background

A. RRAM-based CIM Architecture

Fig. 1 demonstrates a resistive random access memory (RRAM) crossbar array to implement vector-matrix multiplications (VMM) employing the computation-in-memory (CIM) paradigm. The weights of the DNN model are mapped as the resistance states of 1-transistor-1-resistor (1T1R) bitcells in the crossbar array. Multiple inputs are provided simultaneously to the bitcells in each row, where the current through bitcells perform multiplication of these inputs with the content of the bitcell (means weights). These current values are then summed up in a column to perform multiply-and-accumulate (MAC) operations. Such a MAC arrangement can result in high computation parallelism and improve energy efficiency [7].

B. Read-disturb Issue

DNN needs a large number of MAC operations [14], which means a significant of bitcell read operations are required. Due to such multiple read operations, the conductance state of the storage device can be drifted from its initial state, which can eventually switch its state, known as read-disturb [16]. Compared to standard memory, the RRAM-based CIM architectures can be more severely affected by this problem because it has to deal with small current/voltage margins during the MAC operations. Apart from the device-level parameters, the read disturb rate depends on the read voltage as well as the number of read operations [14]. Fig. 2 illustrates the impact of the aforementioned factors on the conductance state ratio of the RRAM, in which the read-disturb and conductance state ratio degradation are observed earlier as the read voltage increases. This deviation in conductance degrades computation accuracy significantly during the analog-to-digital conversion, which reduces the inference accuracy of DNNs [10], [13], [14].

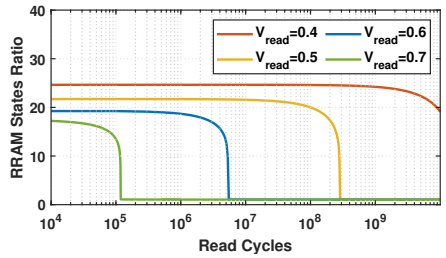
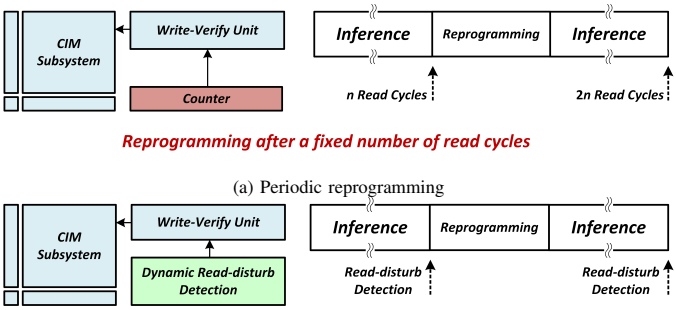


Fig. 2. SPICE simulation of RRAM conductance states ratio degradation



(a) Periodic reprogramming

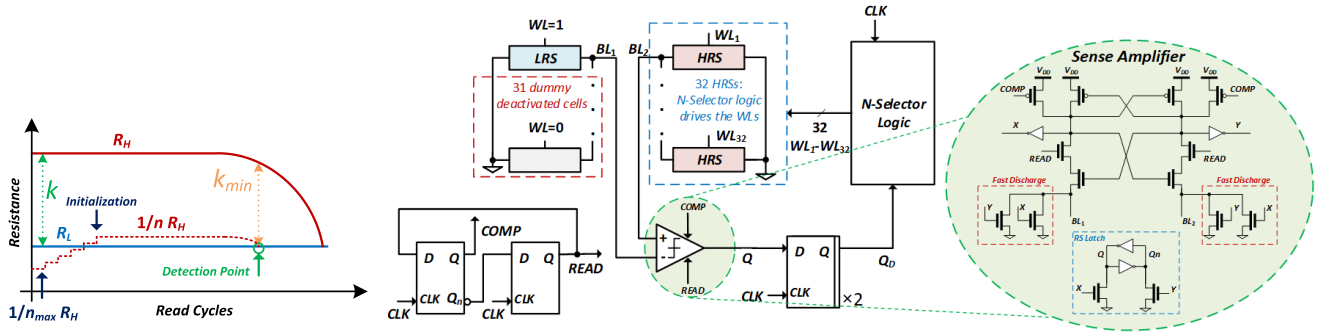
(b) Dynamic reprogramming

Fig. 3. Reprogramming approaches and their operation details.

III. Proposed Read-disturb Detection Approach

A. Overview

The most effective solution to address the read-disturb problem is to reprogram the RRAM devices after a certain number of read cycles. A static approach is proposed to program the RRAM devices after a fixed number of read cycles which is obtained considering the worst-case number of read operations on a set of test cells [13]. However, the number of read cycles for MAC values in a neural network follows a normal distribution [17], which means most of the bitcells experience it close to their mean value of read operations. As RRAMs in the CIM crossbar array do not face the same number of read operations, applying the worst iteration of read operations to test cells results in the definition of a pessimistic periodic reprogramming window. Although the majority of bitcells do not require reprogramming, it may be necessary to rewrite the entire array, even if a few conductance values are degraded for a few cells. This conservative approach ensures the long-term functionality of the RRAM-based CIM crossbar array at the cost of high energy consumption due to the higher write-verify iterations. According to our SPICE simulations, resetting a single-bit RRAM cell consumes 6x more energy than reading a 32x32 crossbar array of one-bit RRAM cells. Furthermore, defining a reprogramming period at design time cannot guarantee its accuracy in the presence of PVT variations of RRAMs at runtime. In this paper, we propose an on-chip read-disturb detection unit, as demonstrated in Fig. 3(b), which can be based on an input-profiled number of read operations on test RRAM bitcells. When the detection unit detects a read-disturb, it triggers the write-verify unit to reprogram the CIM array and test cells.



(a) (b)

Fig. 4. Proposed read-disturb detection methodology: (a) Conceptual diagram to describe the read-disturb detection mechanism (b) Circuit diagram.

B. Detection Methodology and Circuit

In our proposed read-disturb detection methodology, we arrange a set of RRAM devices with a high-resistance state (HRS) configuration in parallel so that its equivalent resistance can be compared with that of the RRAM device with a low-resistance state (LRS) configuration. For that, we employed two test columns of 32 RRAM devices, in which one column has a bitcell with LRS configuration, and the rest other are dummy cells. The other column has 32 bitcells, and all are configured in HRS. We consider the read-disturb of resistance states as illustrated in Fig. 4(a) to describe the proposed detection methodology. To detect RRAM read-disturb, two columns of RRAMs (Fig. 4(b)) are read by a read pulse with 25% duty-cycle to monitor the RRAM states ratio ($k = R_H/R_L$). Due to the large initial states ratio, n HRS bitcells (R_H) are connected in parallel to compare with one LRS bitcell (R_L) under the following condition:

$$R_H/n > R_L \tag{1}$$

The initial equivalent resistance of all parallel connected HRS bitcells in one column is adjusted to be lower than that of with LRS configuration:

$$\frac{R_{H0}}{n_{rows-max}} < R_{L0} \tag{2}$$

where R_{H0}, R_{L0}, and $n_{rows-max}$ are the initial high resistance value, the initial low resistance value, and the maximum number of activated rows in one column, respectively. In order to meet the condition described in (1), the active bit-cells with HRS configurations are turned off one by one during each cycle until the initialization point is reached. The resistance of n parallel-connected HRS bitcells equals one LRS bitcell when the read-disturb is detected at the detection point using a comparator to generate a detection signal. This means that the RRAM states ratio at the detection point is the number of active rows, n_{rows}. Also, the ratio at the detection point can be defined as the minimum required ratio, k_{min}.

Fig. 4(b) demonstrates the proposed detection circuit, which utilizes two columns of the crossbar array. The first column consists of one active LRS bitcell and 31 deactivated dummy cells, while all the wordlines of the second column, all programmed in HRS, are driven by a logic circuit called N-Selector logic. The N-Selector logic is utilized to initialize the number of active HRS bitcells. This block deactivates the last active wordline when the sense amplifier produces a high output. In other words, this block operates only if the condition described in (1) is not met. Therefore, it is employed to initialize the detection unit and remains inactive until the detection point.

A sense amplifier does the comparison between the current values of the two columns. We have employed a pseudo-differential StrongARM latch-based sense amplifier [18], as illustrated in Fig. 4(b). The sense amplifier was modified to apply a read voltage on the bitline using two overlapping $READ$ and $COMP$ pulses. Furthermore, a fast discharging path is considered to prevent post-comparison voltage on RRAMs and to rapidly reset the bitline voltages at the end of the comparison. High input offset is one of the issues of pseudo-differential sense amplifiers. To alleviate the input offset impact on initialization, the output of the sense amplifier has been delayed for two clock cycles. Consequently, the detection circuit initializes with two lower active wordlines than the first initialization point. This technique reduces the sensitivity of the circuit initialization to PVT variations.

C. Minimum RRAM Conductance States Ratio

As shown in Fig. 5, we considered the two low current states (I_0 and I_1) to have an equal relative error, e, which can be defined as $3\sigma/\mu$. In order to prevent the overlap of these current states, the following condition must be fulfilled:

$$I_0(1+e) < I_1(1-e) \tag{3}$$

Considering states ratio of $k(t)$ ($R_H = k(t)R_L$), condition described in (3) can be rewritten as:

$$\frac{n_{rows}V_{read}}{k(t)R_L}(1+e) < (\frac{(n_{rows}-1)V_{read}}{k(t)R_L} + \frac{V_{read}}{R_L})(1-e) \tag{4}$$

Therefore, the minimum states ratio (k_{min}) derives as:

$$k_{min} = \frac{2n_{rows}e - e + 1}{1-e} \tag{5}$$

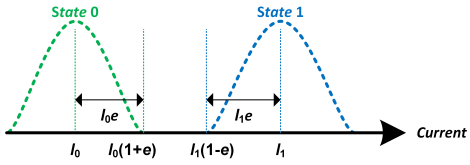


Fig. 5. Conceptual diagram of MAC current states.

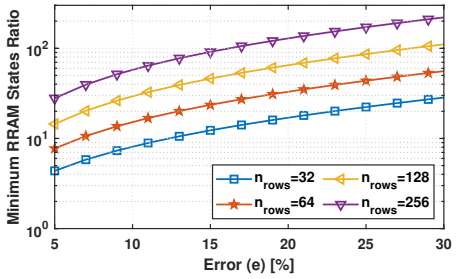


Fig. 6. Minimum required RRAM states ratio for the different numbers of rows and error values.

According to (5), the minimum required ratio is an ascending function of the number of rows, which indicates that it is essential to meet the conductance ratio criterion according to the memory size in design time. Please see Section IV-B for the simulation results for the minimum required ratio.

IV. Simulation Results

A. Setup

The proposed read-disturb detection unit has been designed using TSMC 40 nm CMOS technology with 0.9 V supply voltage and simulated using Cadence Virtuoso tools. Due to the inactive nature of N-Selector logic after initialization, it has been implemented using Verilog-A. Moreover, we have employed the JART VCM v1b RRAM model [19], [20]. To simulate the read behavior, a 200 MHz clock frequency is used in the proposed circuit, which applies a 460 mV voltage for a 5 ns period on bitlines followed by a 15 ns bitline reset according to the sense amplifier, augmented by the considered $READ$ and $COMP$ signals in the proposed circuit.

B. Analytic Results

The curves illustrated in Fig. 6 demonstrate the minimum required states ratio for the different numbers of rows and error values. Equation (5) states that in cases where the desired error is less than 20%, the minimum required ratio is less than the number of rows in the crossbar. This means that the monitoring circuit requires only one column of parallel-connected HRS bitcells, while multiple columns of parallel-connected HRS bitcells are essential for higher error values. However, the relative error value can be reduced significantly below 20% by employing the write-verify scheme during the design time [21].

C. Circuit-level Results

Simulation results demonstrate 75.9 μW power consumption, 1.6 fJ power-delay product (PDP), and less than 3.1×10^{-20} Js energy-delay product (EDP) for a 20 ns read-comparison operation in initialized condition. It is seen from

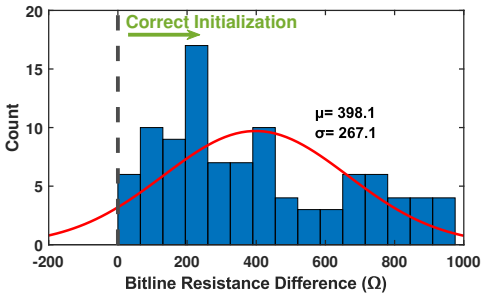


Fig. 7. Distribution of the resistance difference after initialization.

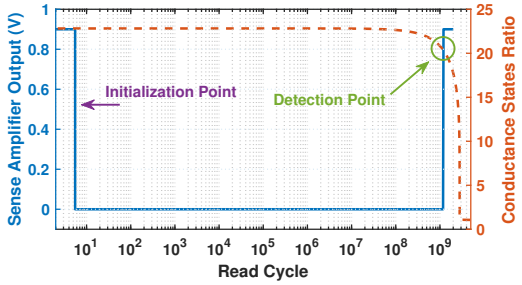


Fig. 8. Simulated transient response of the proposed circuit.

the Monte-Carlo simulation results of Fig. 7, that the detection circuit was initialized correctly in all of the 100 run points and satisfied the condition described in (1). The proposed detection unit was also evaluated in both the initialization and detection phases, as depicted in Fig. 8. It is seen that when the RRAM states ratio is degraded by 10% to its original value, the proposed circuit generates a detection signal.

Comparing the detected point in Fig. 8 with the ratio drop for a worst-case inputted RRAM cell, a reduction of 19% is indicated in the worst-case inputted RRAM cell. The worst-case input is rare [17] and its detection point occurs approximately 6×10^8 read cycles prior to the averaged input detection point. In other words, the proposed detection methodology triggers the write and verify unit around 50% less than the pessimistic reprogramming approach, which means $2\times$ write energy efficiency. Note that the acceptable conductance ratio reduction can be determined based on the calculated minimum required ratio and the desired error margin (e) for the actual crossbar in a specific application.

V. Conclusion

In this paper, an approach for detecting RRAM read-disturb in real-time by monitoring the RRAM states ratio was presented. Our proposed method facilitates dynamic reprogramming by providing superior energy efficiency and computation accuracy compared to the static method. Further, an analytical method was developed to determine the minimum required RRAM states ratio in a CIM architecture. According to SPICE simulation results using TSMC 40 nm CMOS Technology, the proposed methodology promises a 2x increase in the energy efficiency of the writing operation over conventional periodic reprogramming through the reduction of redundant write operations. The proposed approach is a potential solution for developing RRAM-based CIM architectures.

REFERENCES

[1] A. Sebastian, M. L. Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[2] L. Ni, H. Huang, Z. Liu, R. V. Joshi, and H. Yu, "Distributed in-memory computing on binary rram crossbar," *Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, 2017.

[3] S. Diware, A. Gebregiorgis, R. V. Joshi, S. Hamdioui, and R. Bishnoi, "Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021, pp. 1–4.

[4] A. E. Arrassi, A. Gebregiorgis, A. E. Haddadi, and S. Hamdioui, "Energy-efficient snn implementation using rram-based computation in-memory (cim)," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2022, pp. 1–6.

[5] H. Jiang, W. Li, S. Huang, S. Cosemans, F. Catthoor, and S. Yu, "Analog-to-digital converter design exploration for compute-in-memory accelerators," *IEEE Design & Test*, vol. 39, no. 2, pp. 48–55, 2022.

[6] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290–299, 2019.

[7] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.

[8] W. Shim, Y. Luo, J.-s. Seo, and S. Yu, "Impact of read disturb on multilevel rram based inference engine: Experiments and model prediction," in *2020 IEEE International Reliability Physics Symposium (IRPS)*, 2020, pp. 1–5.

[9] G. Pedretti, E. Ambrosi, and D. Ielmini, "Conductance variations and their impact on the precision of in-memory computing with resistive switching memory (rram)," in *2021 IEEE International Reliability Physics Symposium (IRPS)*, 2021, pp. 1–8.

[10] S. Yu, W. Shim, X. Peng, and Y. Luo, "Rram for compute-in-memory: From inference to training," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 7, pp. 2753–2765, 2021.

[11] Y. Lin, J. Tang, B. Gao, Q. Qin, Q. Zhang, H. Qian, and H. Wu, "High-speed and high-efficiency diverse error margin write-verify scheme for an rram-based neuromorphic hardware accelerator," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2022.

[12] Z. Jing, B. Yan, Y. Yang, and R. Huang, "Vsdca: A voltage sensing differential column architecture based on 1t2r rram array for computing-in-memory accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 10, pp. 4028–4041, 2022.

[13] W. Li, X. Sun, S. Huang, H. Jiang, and S. Yu, "A 40-nm mlc-rram compute-in-memory macro with sparsity control, on-chip write-verify, and temperature-independent adc references," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 9, pp. 2868–2877, 2022.

[14] W. Shim, Y. Luo, J.-S. Seo, and S. Yu, "Investigation of read disturb and bipolar read scheme on multilevel rram-based deep learning inference engine," *IEEE Transactions on Electron Devices*, vol. 67, no. 6, pp. 2318–2323, 2020.

[15] B. Yan, J. Yang, Q. Wu, Y. Chen, and H. Li, "A closed-loop design to enhance weight stability of memristor based neural network chips," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 541–548.

[16] C. Bengel, J. Mohr, S. Wiefels, A. Singh, A. Gebregiorgis, R. Bishnoi, S. Hamdioui, R. Waser, D. Wouters, and S. Menzel, "Reliability aspects of binary vector-matrix-multiplications using reram devices," *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 034001, 2022.

[17] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao *et al.*, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, 2022.

[18] B. Razavi, "The strongarm latch [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 7, no. 2, pp. 12–17, 2015.

[19] C. Bengel, A. Siemon, F. Cüppers, S. Hoffmann-Eifert, A. Hardtdegen, M. von Witzleben, L. Hellmich, R. Waser, and S. Menzel, "Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using spice level compact models," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4618–4630, 2020.

[20] [Online]. Available: emrl.de/JART.html

[21] Y. Luo, X. Han, Z. Ye, H. Barnaby, J.-S. Seo, and S. Yu, "Array-level programming of 3-bit per cell resistive memory and its application for deep neural network inference," *IEEE Transactions on Electron Devices*, vol. 67, no. 11, pp. 4621–4625, 2020.