UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

Energy-Efficient Neuromorphic Computing with CMOS-Integrated Memristive Crossbars

Permalink

https://escholarship.org/uc/item/4jp0w0jc

Author

Fahimi, Zahra

Publication Date

2021

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Santa Barbara

Energy-Efficient Neuromorphic Computing with CMOS-Integrated Memristive Crossbars

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Electrical and Computer Engineering

by

Zahra Fahimi

Committee in charge:

Professor Dmitri Strukov, Chair

Professor Tim Sherwood

Professor James Buckwalter

Professor Li-C Wang

December 2021

The dissertation of Zahra Fahimi is approved.

Tim Sherwood

James Buckwalter

Li-C Wang

Dmitri Strukov, Committee Chair

December 2021

Energy-Efficient Neuromorphic Computing with CMOS-Integrated Memristive Crossbars

Copyright © 2021

By

Zahra Fahimi

Dedicated to you.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Prof. Dmitri Strukov, for your patience, guidance, and support. I have benefited greatly from your wealth of knowledge. I am grateful that you took me on as a student and continued to have faith in me over the years.

Very special thanks to my colleagues especially my soulmate Dr. Mohammad Reza Mahmoodi who has supported me throughout this process and has constantly encouraged me when the tasks seemed difficult and back-breaking.

I also owe much gratitude to my parents and my brothers for their unwavering support and love.

VITA OF ZAHRA FAHIMI
Dec 2021

Education

Bachelor of Science in Electrical Engineering, Shahrekord University, 2011.
Master of Science in Electronics, Isfahan University of Technology, 2015.
Master of Science in Computer Engineering, University of California, Santa Barbara, June 2021.
Doctor of Philosophy in Electronics, University of California, Santa Barbara, Dec 2021 (expected).

Selected Publications

Z. Fahimi, *et al*. Mitigating Imperfections in Mixed-Signal Neuromorphic Circuits, *https://arxiv.org/abs/2107.04236*, (2021).

Z. Fahimi, *et al*. The Impact of Device Uniformity on Functionality of Analog Passively-Integrated Memristive Circuits, *IEEE Transactions on Circuits and Systems I*, (2021).

Z. Fahimi, *et al*. Combinatorial Optimization by Weight Annealing in Memristive Hopfield Networks, *Scientific Reports*, (2021).

Z. Fahimi, *et al*. Mixed-signal computing with non-volatile memories, *SRC Technical Conference (SRCTechCon'18)*, (2018).

M. R. Mahmoodi, **Z. Fahimi**, *et al*. A Strong Physically Unclonable Function with >280 CRPs and <1.4% BER Using Passive ReRAM Technology, *IEEE Solid-State Circuits Letters*, (2020).

M. R. Mahmoodi, H. Nili, **Z. Fahimi**, *et al*. Ultra-Low Power Physical Unclonable Function with Nonlinear Fixed-Resistance Crossbar Circuits, *Electron Devices Meeting (IEDM'19)*, (2019).

M. R. Mahmoodi, H. Kim, **Z. Fahimi**, *et al*. An Analog Neuro-Optimizer with Adaptable Annealing Based on 64×64 0T1R Crossbar Circuit, *Electron Devices Meeting (IEDM'19)*, (2019).

ABSTRACT

Energy-Efficient Neuromorphic Computing with CMOS-Integrated Memristive Crossbars

by

Zahra Fahimi

The von Neumann architecture has been broadly adopted in modern computing systems in which the central processor unit (CPU) is separated from the memory unit. During data processing, it is necessary to transfer data between the memory and CPU. For data-intensive applications such as deep neural networks, as the size of data increases, data movement between memory and CPU becomes a significant bottleneck for high throughput and energy-efficient implementation. In-memory computing is a paradigm that tackles this challenge by allowing computation within the memory, i.e., where data are stored. Hence, in-memory computing is a promising solution for implementing energy-efficient neuromorphic systems since it minimizes data transportation between memory and the processing units. The major component in developing neuromorphic circuits is a nanoscale memory device, which is responsible for weight storage and analog computation. Resistive Random-Access Memory (RRAM) is one of the most promising memory candidates due to its long-term retention, analog storage, low-power operation, and compact nanoscale footprint.

The first part of this thesis explores the nonidealities of RRAM technology, such as temperature dependency, stuck-at-fault, and tunning error, and their impact on the accuracy of neuromorphic hardware implementation. We show that these imperfections may significantly degrade the inference accuracy of neuromorphic circuits. To mitigate them, we have proposed a holistic approach based on hardware-aware training in which modifications are done in tunning, circuit, and training phase (ex-situ) of hardware development. The proposed method significantly decreases the accuracy drop across the 25–100 °C temperature range, allows 2.5× to 9× improvement in energy consumption of the memory arrays during inference, and improves the defect tolerance by >100×.

In the second part of this thesis, we also study the impact of device uniformity in passive memristive circuits and the tradeoffs between computing accuracy, crossbar size, switching threshold variations, and target precision. Nonidealities are investigated in two representative deep neural networks, and several solutions, including hardware-aware training, improved tuning algorithm, and switching threshold modification, are proposed to enhance the performance. These techniques allow us to implement advanced deep neural networks (DNNs) with almost no accuracy drop, using state-of-the-art analog 0T1R technology.

In the last part, we focus on integrating passive and active RRAM with CMOS circuits for implementing efficient demos for various applications such as neural networks. First, focusing on passive technology, we show the building block circuit that facilitates the forming, programing, reading, inference, and monitoring of RRAM circuits. We discuss several neuromorphic networks and prototype demos with integrated analog passive RRAM and CMOS. The designs are fabricated in two wafer-scale tapeout runs in 180 nm CMOS technology, and preliminary encouraging experimental results are obtained. Second, we demonstrate a massive DNN accelerator fabricated in a standard 65 nm CMOS process with integrated active analog RRAM devices. The main focus is on novelties in the design of the VMM and tuning circuits, which reduced the impact of IR drop, improved the area efficiency, and allowed massive parallel programming features in this chip.

TABLE OF CONTENTS

1. Introduction

1.1. Motivation

Deep neural networks (DNNs) are currently the foundation for modern artificial intelligence (AI) applications [1] and have been very successful in large-scale recognition and classification tasks [1-5]. This momentum stems from two main reasons: First, the exponential increase in the computational power exhibited by graphics processing units (GPU), which harnesses extreme parallelism by using many cores, each with a dedicated or shared high-throughput connection with memory. Second, the availability of a vast amount of labeled data has abled DNNs to extract high-level features from unprocessed data. Hence, the rise of powerful GPUs and massive labeled datasets paved the way toward training DNNs in a reasonable time with high accuracies. However, pure software DNNs executed in supercomputers with thousands of CPU/GPUs suffer from high energy consumption [6] because during the execution of various computational tasks, large amounts of data need to be traveled back and forth between the processing and memory unit, and this causes high costs in latency and energy.

To overcome the need for transferring data frequently between the memory and the processing unit, and improve the energy efficiency significantly [7], novel non-von Neumann computing models are developed that, e.g., rely on the idea of in-memory, in which calculations are carried out where the data are stored [8]. Therefore, it yields to suppress the energy- and time-consuming of memory-processor communications extremely.

Among different in-memory computing schemes, designs based on computational memory devices [9-12] are one of the most effective implementations as efficient in-memory computing generally requires fast, low-power, high-density, scalable memory devices. Analog-grade non-volatile memories (NVMs), such as those based on floating-gate transistor [13- 15], phase-change [16-18], ferroelectric [19, 20], magnetic [21], solid-state electrolyte [22-24], organic [25, 26], and metal-oxide [27-30] materials are enabling components for mixed-signal circuits implementing Vector-by Matrix Multiplication (VMM), which is the most common operation in any artificial neural network. Such circuits allow for physical level in-memory computations in the analog domain using the fundamental Ohm and Kirchhoff laws, thus enabling dramatically higher energy and area efficiency in comparison with digital solutions.

The general architecture of mixed-signal VMMs based on NVMs depicted in Fig. 1. Here, vector-by-matrix multiplication is defined as $y = Wx$, where $x \in \mathbb{R}^N$ is the input vector, $y \in \mathbb{R}^M$ is the output vector, and W is the weight matrix. N and M are the number of inputs and outputs, respectively. The input vector is presented to the digital-to-analog converters (DACs), which convert the digital input to analog signals. The predetermined weight vector is encoded to the conductance of the NVM cells. NVM cell at each crosspoint generates current proportional to the amplitude of the input signal times its conductance. As a result, the multiplication and summation operations within a memory array are performed in parallel using Ohm's law. The current in all columns is summed up based on Kirchhoff's law and sensed by the peripheral circuits (PC), which is then followed by analog-to-digital converters (ADCs).

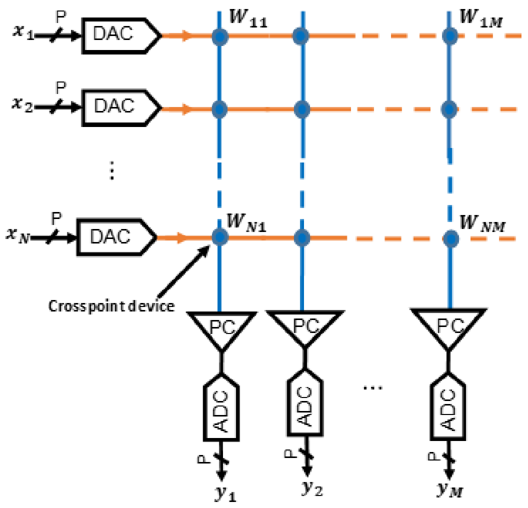


Fig. 1: The general architecture of a mixed-signal VMM with nonvolatile memory at each crosspoint.

The most important memory device characteristics in the context of analog VMM circuits are cell size and scalability. The main advantages of using passively integrated metal-oxide memristors [28], which are also referred to as RRAMs, are their superior density and lower fabrication cost [31]. The high scalability of RRAMs makes it possible to implement very dense resistive memory arrays. Such architectures using memristive devices are of high interest for their possible applications in in-memory computing based on nonvolatile memory design [32-34], digital and analog programmable systems [35, 36], and neuromorphic computing structures [37,38].

Developing circuits based on RRAM devices faces some challenges. The first concern is related to non-idealities of the devices such as temperature dependency, tuning error, device-to-device variations, and defects, which lead to considerably degrading the performance of neural network-based RRAMs. The second challenge is the integration of memristive crossbars with CMOS circuits, which implement nonfrequent peripheral functions. The main goal of this thesis is to address these two challenges. In particular, Chapter 2 explore different types of device imperfections and their impacts on the performance of neural networks based on RRAM crossbars. Then, we analyze each non-ideality very precisely and then introduce a holistic approach to mitigate these imperfections during training of the networks without any extra hardware. In Chapter 3, we study the impact of device uniformity on the functionality of memristor circuits, where they are used to implement either a neural network or an optimization platform.

Finally, in Chapter 4, we discuss our progress toward developing CMOS-integrated memristive crossbars and neuromorphic circuits and systems based on passive and active ReRAM circuits.

1.2. Background and Significance

1.2.1. CMOS Integrated RRAM Crossbars

Memristive devices are categorized into two different types: 1) zero-transistor-one-memristor (0T1R), which is mainly called passive memristors, and 2) one-transistor-one-memristor (1T1R) memristors, which are commonly referred to as active memristors. In comparison with passive crossbar arrays, the 1T1R structure improves the network performance in two aspects: First, proper current compliance determined by the transistor allows a more controllable conductance update compared with that in 0T1R arrays. Second, the gate transistor can prevent disturbance to the states of unselected devices during programming. The 1T1R structure can also eliminate the sneak path problem of the passive crossbar, although the sneak path is not relevant in the context of neuromorphic computing inference. On the other side, arrays based on passive memristors are more compact and scalable. Fig. 2b and 2c indicate the structure of a passive and active memristor device used in a memristive array (Fig. 2a).

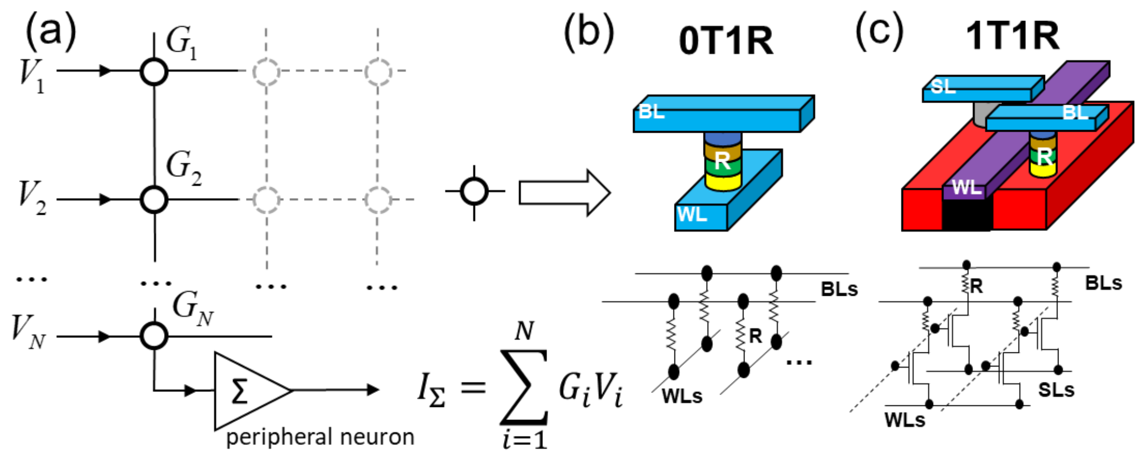


Fig. 2: (a) A memory array based on memristive devices. A structure of (b) 0T1R and (c) 1T1R memristor device

The specific focus of this section is on the state-of-the-art nonvolatile (filamentary) analog-grade 0T1R metal-oxide devices, while only a few representative works are listed for metal-oxide 1T1R and solid-state-electrolyte 0T1R circuits.

In 2015, for the first time, a 12×12 transistor-free memristive crossbar consisting of bilayer metal-oxide memristors was used to experimentally demonstrate an artificial neural network based on memristive devices [39, 40]. A 10×6- and 10×8- neuromorphic pattern classifiers were implemented and trained in-situ using the Manhattan-Rule algorithm. The retention and endurance of the devices were measured > 140 hrs and > 200 K, respectively.

One year later, in 2016, the first 3-D monolithic stack of two passive 10 × 10 crossbars for analog computing applications was reported in Ref. [41]. Furthermore, the crosspoint memristors are optimized for analog computing applications allowing successful forming and switching off all devices in the demonstrated crossbar circuit and, most importantly, precise tuning of the devices' conductance values within the dynamic range of operation. Using a low, less than 175 °C, temperature budget during the fabrication process and performing a planarization step before the deposition of the second crossbar layer improved yield (~100%) and uniformity of the crosspoint devices significantly.

In 2017, several milestone papers were published. The first one was on experimental implementation of sparse coding algorithms based on a 32 × 32 crossbar array of analog 0T1R devices [42]. In this experiment, a 25 × 20 sub-array of the memristor array was utilized to validate the algorithm. The retention of the devices is reported in the order of minutes. The second work [43] was based on the demonstration of using memristor arrays to perform principal component analysis (PCA), which is a feature extraction technique commonly used in unsupervised learning. Ref. [44] introduced an implementation of a grey-scale face classification based on a 1024-cell array of 1T1R memristive devices with parallel online training.

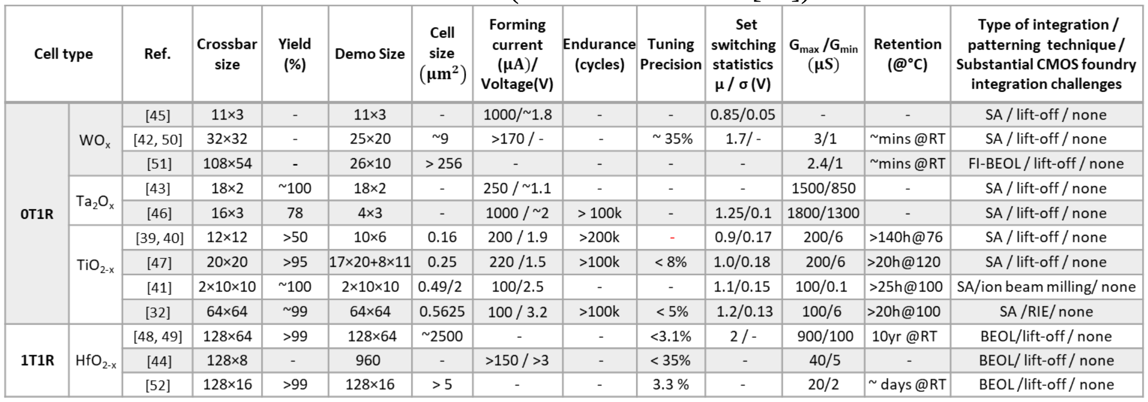
Some breakthrough works using both 0T1R and 1T1R memristive arrays were introduced in 2018. An 11 × 3 array of passive memristors was fabricated and used to implement the simulated annealing technique for solving a spin glass problem [45]. In Ref. [46], a 16 × 3 fabricated passive memristor crossbar was used to demonstrate neural networks for K-means clustering analysis. The first demonstration of a multilayer perceptron (MLP) using memristive crossbar arrays was proposed in Ref. [47]. In this work, two passive 20 × 20 metal-oxide memristive crossbar arrays are used to implement an MLP network, which includes 16 inputs, 10 hidden-layer neurons, and 4 outputs. More than 20 hrs of high-temperature retention and 100 k endurance are reported for the devices used in the array. Ref. [48, 49] partitioned a single 128 × 64 1T1R array built from high retention devices (> 10 years) to construct a 3-layer perceptron with 64 input neurons, 54 hidden neurons, and 10 output neurons trained on the MNIST dataset of handwritten digits.

In 2019, a reservoir computing hardware system based on a 32 × 32 WO_x memristor was reported [50] that can efficiently process temporal data. A passive memristor crossbar array was directly integrated with all the necessary CMOS circuitries on Ref. [51]. The size of the array is reported as 108 × 54, but a 26 × 10 sub-array of it was used to demonstrate three models—a perceptron network, a sparse coding algorithm, and a bilayer PCA system. Retention of the devices is measured in the order of minutes. A five-layer mCNN for MNIST digit image recognition was also designed using 2,048 1T1R devices [52] in 2020.

In 2021, the largest fully functional passive crossbar was introduced [32]. A 64×64 passive crossbar circuit with ~ 99% functional nonvolatile metal-oxide memristors was demonstrated to implement a vector-by-matrix multiplier used for the MNIST image classification.

Table. I show a summary of previous works implemented by memristor devices. The common feature between all discussed works is that the main operations can be performed efficiently with memristor crossbar arrays. These implementations have mostly relied on external printed-circuit boards (PCB) to provide the required parameter analyzers and control circuitries such as ADCs, DACs, peripheral/sensing circuits, tunning switching, etc., to generate and collect signals. So, the functionality of the systems can be limited to accessing these discrete circuits. To overcome this issue, implementing a system such that memristive crossbars are fully integrated with all necessary CMOS circuities is required. Therefore, integrating all the necessary circuits and memristor crossbar on one wafer allows us to scale the large computing systems and increase the computation speed. In Chapter 4, we introduce several fully integrated CMOS-RRAM chips designed for different neuromorphic applications.

Table I: A summary of previous work on implementing memristive crossbar arrays with 0T1R and 1T1R devices (some data are from [32])



Cell type		Ref.	Crossbar size	Yield (%)	Demo Size	Cell size (μm^2)	Forming current (μA)/ Voltage(V)	Endurance (cycles)	Tuning Precision	Set switching statistics μ / σ (V)	G_{max} /G_{min} (μS)	Retention (@°C)	Type of integration / patterning technique / Substantial CMOS foundry integration challenges
0T1R	WO_x	[45]	11×3	-	11×3	-	1000/~1.8	-	-	0.85/0.05	-	-	SA / lift-off / none
		[42, 50]	32×32	-	25×20	~9	>170 / -	-	~ 35%	1.7/ -	3/1	~mins @RT	SA / lift-off / none
		[51]	108×54	-	26×10	> 256	-	-	-	-	2.4/1	~mins @RT	FI-BEOL / lift-off / none
	Ta_2O_x	[43]	18×2	~100	18×2	-	250 / ~1.1	-	-	-	1500/850	-	SA / lift-off / none
		[46]	16×3	78	4×3	-	1000 / ~2	> 100k	-	1.25/0.1	1800/1300	-	SA / lift-off / none
	TiO_{2-x}	[39, 40]	12×12	>50	10×6	0.16	200 / 1.9	>200k	-	0.9/0.17	200/6	>140h@76	SA / lift-off / none
		[47]	20×20	>95	17×20+8×11	0.25	220 /1.5	>100k	< 8%	1.0/0.18	200/6	>20h@120	SA / lift-off / none
		[41]	2×10×10	~100	2×10×10	0.49/2	100/2.5	-	-	1.1/0.15	100/0.1	>25h@100	SA/ion beam milling/ none
		[32]	64×64	~99	64×64	0.5625	100 / 3.2	>100k	< 5%	1.2/0.13	100/6	>20h@100	SA /RIE/ none
1T1R	HfO_{2-x}	[48, 49]	128×64	>99	128×64	~2500	-	-	<3.1%	2 / -	900/100	10yr @RT	BEOL/lift-off / none
		[44]	128×8	-	960	-	>150 / >3	-	< 35%	-	40/5	-	BEOL/ lift-off / none
		[52]	128×16	>99	128×16	> 5	-	-	3.3 %	-	20/2	~ days @RT	BEOL /lift-off / none

1.2.2. Addressing non-idealities in neuromorphic circuits

The advancement of emerging technologies like RRAMs, which enable storing analog information and implement neural computation efficiently, has caused huge progress in the neuromorphic computing area. However, various device- and system-level non-idealities such as temperature dependency, defects, tuning error, etc., inherently rise in such analog circuits. These concerning factors may considerably degrade the performance/inference-accuracy of neural networks implemented based on nonvolatile memory devices.

The dependency of the devices on temperature has a very significant impact on neuromorphic circuits and systems. The weights of the network change noticeably with temperature variation followed by modifying the pre-activation signals of the neurons that results in reducing the inference accuracy [53].

Another non-ideality in memristive devices is device-to-device variations. The switching variation between devices implies that various amounts of applied voltages will be required to tune different devices, which would lead to the half-select disturbance problem, i.e., programming a memristor device disturbs the state of other devices located in the same column or rows if they have a switching threshold less than the threshold of the programmed device.

Several promising studies investigate the effect of device imperfections on network performance and propose reliable techniques to improve the accuracy. The solutions could involve hardware modifications or software training approaches. Existing hardware-based solutions cause enormous overhead and power consumption while software-based methods are more efficient.

A memristor crossbar array modeled in Ref. [55], incorporating the line resistance and device nonlinearities. Simulation results show the line resistance causes voltage and current degradation, and it may affect array operation when arrays and devices are highly scaled. Therefore, line resistance should not be ignored for relatively large crossbars. In Ref. [56], the influences of some hardware limitations such as IR-drop, device variation, and programing error are analyzed. A variation-aware off-device training scheme called "Vortex" is developed to tolerate device imperfections and design constraints. The algorithm is tested on a two-layer neural network implemented by memristor crossbars for MNIST classification. A general conversion algorithm was developed and experimentally tested in Ref. [57] to map arbitrary matrix values to memristor conductances to compensate the accuracy drop due to IR drop and nonlinearity characteristics of the devices.

In 2017, a method to eliminate the parasitic resistance across memristor arrays was presented in Ref. [58]. This technique is based on adding extra resistors to ensure that every single device sees the same parasitic resistance. Due to the limitation of the immature fabrication technology, many types of defects may exist in RRAM-based computing systems, such as hard faults and soft faults. Soft faults can be easily calibrated because the resistance of the devices is adjustable. Hard faults like Stuck at-Faults (SAFs) which faulty devices get stuck at high-resistance state (HRS) or low-resistance state (LRS) are popular in memristive arrays. In order to mitigate this issue, a mapping algorithm with inner fault-tolerance is proposed to make multiple faulty RRAM columns eliminate the impact of SAFs on each other [59]. Another method to eradicate the effect of fault-tolerant and programing errors was proposed in Ref. [60]. The method is based on an on-chip training scheme that selects a small portion of the model parameters randomly and then, maps them to on-chip memory. After the model is mapped to the hardware, it adapts the parameters. The algorithm was demonstrated by improving the accuracy of CNNs networks for MNIST and CIFAR-10.

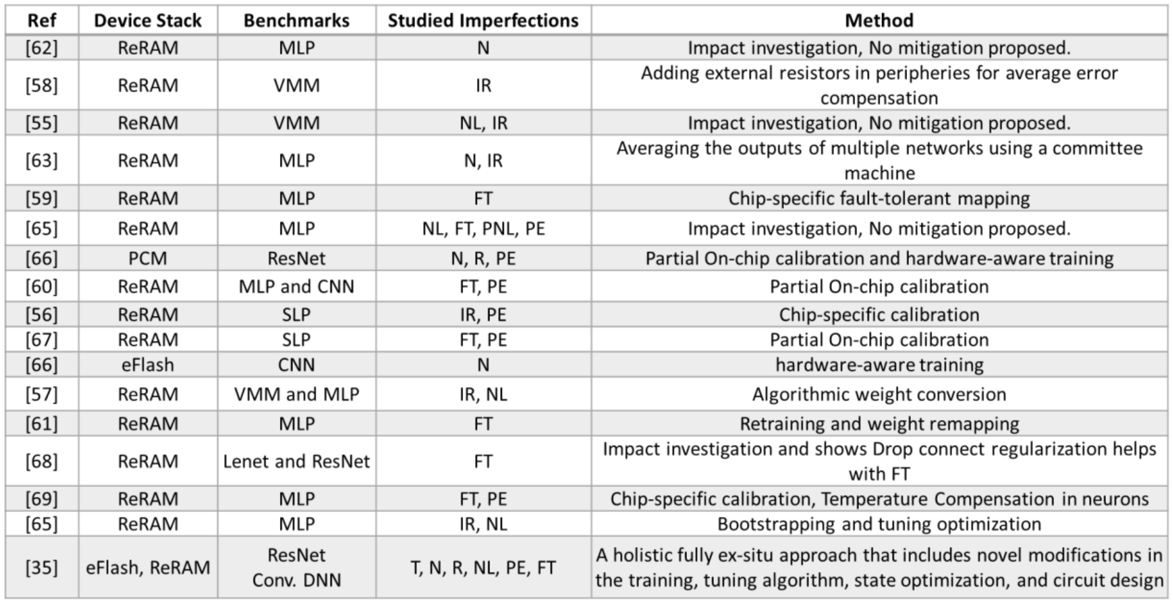
A defect releasing methodology to improve inference accuracy was demonstrated on an MLP network implemented by memristor arrays [61]. In this method, the weights are classified into significant and insignificant ones based on their impact on the performance. Then, a retraining algorithm is applied to compensate the device failure by re-tuning the trainable weights. A remapping algorithm that utilizes a redundancy scheme can further improve the computation accuracy, especially when a large number of defects occur in the weights.

RTN and thermal noise are critical issues in nanoscale semiconductor devices. The impact of RTN amplitude and its occurrence rate in both filamentary and non-filamentary RRAM devices are analyzed in Ref. [62]. The effect of RTN is evaluated on the pattern recognition accuracy of an MLP implemented by memristive crossbars. The investigations show that the non-filamentary RRAM has a tighter RTN amplitude distribution and a much lower occurrence rate than the filamentary devices. So, non-filamentary devices lead to less RTN impact on inference accuracy. Ref [63] proposed a method called Committee Machine (CM) employing ensemble averaging (EA) [64] to increase inference accuracy in the presence of RTN, and faulty devices, without increasing the number of memristors devices used in the network. The network is a fully connected network, including 25 hidden layers which was tested using experimental data to extract the effect of device non-idealities. Ref. [34] used bootstrapping and tuning optimization techniques to improve the computation precision of analog VMM circuits affected by device nonlinearity, variations, and line resistance.

So far, all the studied techniques are focused on increasing the accuracy of networks based on RRAM devices by eliminating their imperfections. However, Ref. [36] developed a noise model for a mixed-signal neural network hardware accelerator based on embedded NOR flash memory technology. Using a hardware-aware training method and combining the model distortions during off-chip training make the network more robust to noise.

For the first time, we report a comprehensive characterization of critical imperfections in two analog-grade memories, passive memristors and redesigned eFlash memories [35]. Imperfections are major hurdles in the path of further progress of these technologies. Hence, a practically viable approach is developed to deal with these non-idealities and release the full potential of nonvolatile memories in neuromorphic systems. An extensive characterization of imperfections in mainstream analog-grade synaptic devices is performed, and a holistic hardware-aware ex-situ approach is developed to reduce their negative impact on the performance of DNNs. Table II summarizes the current and previous works focused on the impact of RRAM and eFlash imperfections in mixed-signal neuromorphic circuits and vector-by-matrix multipliers.

Table. II: A Summary of previous works focused on the effect of device imperfections in mixed-signal neuromorphic circuits



Ref	Device Stack	Benchmarks	Studied Imperfections	Method
[62]	ReRAM	MLP	N	Impact investigation, No mitigation proposed.
[58]	ReRAM	VMM	IR	Adding external resistors in peripheries for average error compensation
[55]	ReRAM	VMM	NL, IR	Impact investigation, No mitigation proposed.
[63]	ReRAM	MLP	N, IR	Averaging the outputs of multiple networks using a committee machine
[59]	ReRAM	MLP	FT	Chip-specific fault-tolerant mapping
[65]	ReRAM	MLP	NL, FT, PNL, PE	Impact investigation, No mitigation proposed.
[66]	PCM	ResNet	N, R, PE	Partial On-chip calibration and hardware-aware training
[60]	ReRAM	MLP and CNN	FT, PE	Partial On-chip calibration
[56]	ReRAM	SLP	IR, PE	Chip-specific calibration
[67]	ReRAM	SLP	FT, PE	Partial On-chip calibration
[66]	eFlash	CNN	N	hardware-aware training
[57]	ReRAM	VMM and MLP	IR, NL	Algorithmic weight conversion
[61]	ReRAM	MLP	FT	Retraining and weight remapping
[68]	ReRAM	Lenet and ResNet	FT	Impact investigation and shows Drop connect regularization helps with FT
[69]	ReRAM	MLP	FT, PE	Chip-specific calibration, Temperature Compensation in neurons
[65]	ReRAM	MLP	IR, NL	Bootstrapping and tuning optimization
[35]	eFlash, ReRAM	ResNet Conv. DNN	T, N, R, NL, PE, FT	A holistic fully ex-situ approach that includes novel modifications in the training, tuning algorithm, state optimization, and circuit design

* T: Temperature, N: Noise, PE: programming error, in 0T1R, FT: Fault-tolerance, NL: Nonlinearity, R: Retention, IR: IR drop, PNL: programming nonlinearity.

2. Mitigating Imperfections in Mixed-Signal Neuromorphic Circuits

The progress in neuromorphic computing is fueled by the development of novel nonvolatile memories capable of storing analog information and implementing neural computation efficiently. The most notable candidates that excel in primary features such as long-term retention, high endurance, analog storage, low-power operation, and compact footprint are metal-oxide passive memristors [47] (Fig. 3b) and redesigned eFlash memories (Fig. 3c) [70]. Network weights are encoded into two the conductance of memristors (synapses) or the ratio of the state currents of two eFlash devices to a peripheral eFlash memory (Fig. 3b-c). The input/outputs are encoded as voltages (V_i) in memristive circuits and currents (I_i) in eFlash VMMs. Nevertheless, all synaptic devices are generally more or less prone to imperfection such as temperature dependency, yield, drift, tuning error, and static nonlinearity. While imperfections are not necessarily meant to be detrimental (see, e.g., [37]), they severely degrade the accuracy of currently popular DNNs. Indeed, imperfections are major obstacles in the path of further progress and the ultimate commercialization of these technologies. Hence, a practically viable approach should be developed to deal with these nonidealities and unleash the full potential of nonvolatile memories in neuromorphic systems.

The endeavors to improve device reliability are ongoing and actively pursued. A massive number of works focus on improving synapse reliability by harnessing novel materials and stacks, e.g., reducing noise [62], enhancing uniformity [41,71], linearity [72], which is vividly the most promising approach in the long run. In the meantime, a large body of research explores circuit, system, and algorithmic techniques to mitigate these nonidealities. Such efforts are categorized into 5 approaches, as shown in Fig. 4.

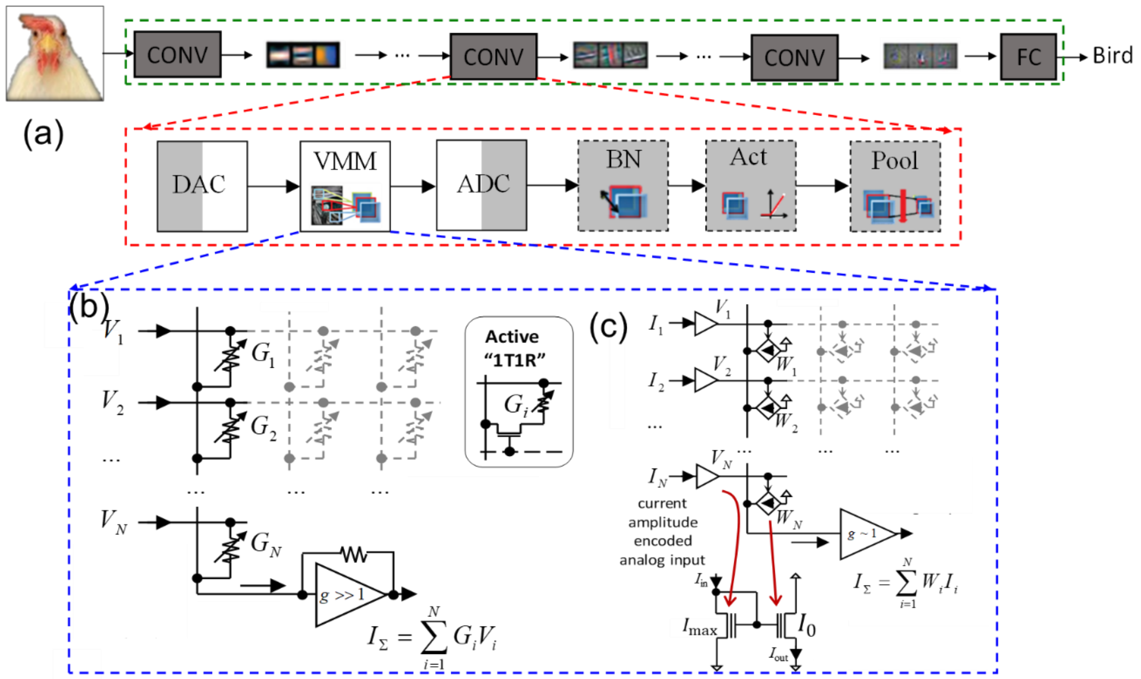


Fig. 3: Mixed-signal neuromorphic circuits. (a) Major computations involved in every layer of a mixed-signal neuromorphic classifier. Highlighted in gray (white) are those typically implemented in the digital (analog) domain (BN: batch normalization, Act: activation function, Pool: an optional pooling layer, FC: fully-connected classifier). VMM implementation using (b) memristive crossbars and (c) gate-couple eFlash memory arrays.

The most predominant approach in implementing neuromorphic systems is through ex-situ training, in which synaptic weights are calculated on a precursor server [36]. The synaptic weights are then transferred to numerous mixed-signal chips, which only support the inference task. A trivial approach to cope with imperfections is by adding redundancy [73]: the model and algorithm hyperparameters are selected such that the deployed model in the analog domain would tolerate a certain amount of unreliability. For example, an enormous network such as AlexNet can endure a large amount of noise that allows weight binarization [74]. In more compact models, it might be possible to lower computing precision without any impact on accuracy at the cost of redundancy. For example, the accuracy loss by changing weight precision from 4-bit to 2-bit in ResNet-18 can be compensated by doubling the model size [75]. Ref. [63] proposes to enlarge the capacity of a fully connected network through committee machines and validate the results on the MNIST dataset. However, the major concern with this approach is the lack of evidence that this approach is scalable to complicated tasks.

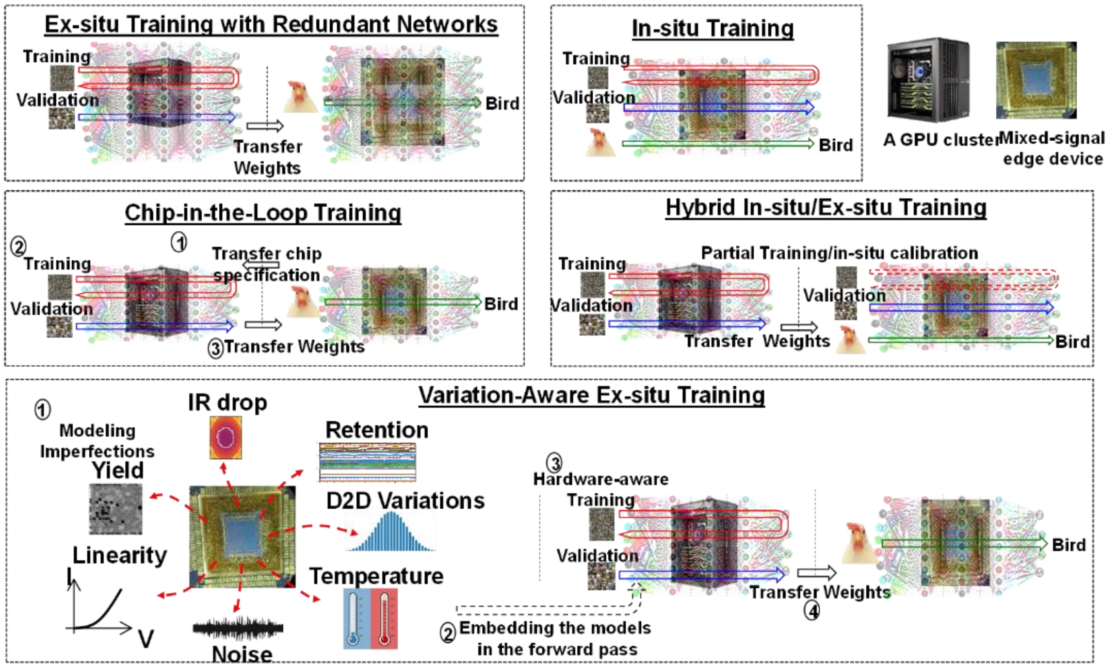


Fig. 4:Various approaches to mitigate imperfections in mixed-signal neuromorphic networks.

The second approach in dealing with imperfections is through in-situ training [76,77], in which the training and inference are both performed on the mixed-signal hardware. The first drawback is the substantial areal overhead needed for the infrequent training operations, e.g., to compute and store gradients. Besides, while the chip could become resilient to some imperfections, other nonidealities associated with the dynamic behavior of devices may arise. Some works propose a hybrid approach that imposes less resource overhead, e.g., the model is initially trained ex-situ, and then an online calibration scheme modifies the weights in the run time. For example, Ref. [66] proposes an adaptive batch normalization technique that effectively compensates for the retention loss in memory cells. The drift in phase-change memories is high (~50% conductance drift after 100 ms [78]) that an always-on compensation circuit is required.

Another solution is through chip-in-the-loop ex-situ training [57,61,79], in which specific features are measured for an individual chip (e.g., faulty synapse locations [67] or drift statistics [52]) and then applied in the training phase in the server. The adapted weights are then transferred to the chip in the deployment phase. Chip-in-the-loop ex-situ training could also be implemented by running all forward pass operations in the target device and all backward pass operations in the GPU cluster. But strategies that include device avoidance/reconfiguration/remapping or are chip-specific might present scalability challenges, despite the ability to boost the performance of an individual chip.

On the other hand, hardware-aware ex-situ training is a more scalable method in which hardware nonidealities are modeled and included in the training phase to generate a robust model. Ref. [36] uses device noise models during the training to improve the robustness of a moderate-size mixed-signal convolutional network. In Ref. [68], DropConnect regularization is introduced to enhance the accuracy drop originating from low yield without using nonideality models—at 98% yield, 17% accuracy drop on CIFAR-10 based on ResNet-18 model is reduced to 10%. Most previous works either study a particular nonideality [56,67,69] or consider redundant networks on smaller datasets [80,81]. Besides, some focus solely on simulations with no experimental data or employ practically nonviable devices for modeling purposes.

In this study, major imperfections on two prospective analog-grade synaptic devices, passively-integrated memristors, and eFlash memories, are characterized in order to determine nonidealities that severely impact mixed-signal DNNs. The choice of these promising technologies stems from the fact that, unlike most emerging technologies, they transcend all rudimentary features, including high endurance, analog storage, long-term retention, low-power operation, and nano-scale footprint (see supplementary Table 1 in [32] for a comparison). Besides, low conductance and switching voltage range and very dense cell size ($4F^2$ and ~110 F^2 for memristors and eFlash) allow practical implementation of large-scale mixed-signal DNNs with decently large VMMs (e.g., 64×64) [82].

In the case of eFlash memories, high precision tuning and superb analog-grade retention are reported, and excellent yield is deemed due to the maturity of the technology, making temperature variations the major issues. Passive TiO_2 memristive technology also offers high analog retention in an excellent areal density despite susceptibility to temperature variations, limited yield, and half select disturbance. Each factor is studied separately, and a holistic approach is proposed that includes modifications in the training, tuning, state optimization, and circuits and targets each issue individually. More importantly, the proposed method is practical in terms of implementation cost with negligible overhead and is validated on a hybrid experiment/simulation framework using two benchmarks: a moderate-size convolutional neural network (ConvNet) and ResNet-18 trained on CIFAR-10, and ImageNet datasets, respectively.

The accuracy drop is almost fully recovered in the 20 °C to 100 °C temperature range by employing three incrementally applied approaches: temperature-sweep batch training, k-reference batch normalization, and state optimization. The models are also resilient against the minor static nonlinearity (dot-product nonlinearity, i.e., *IV* nonlinearity in memristors and subthreshold slope nonlinearity in eFlash). Two techniques are proposed to overcome the limited yield in emerging technologies, pair modification that minimizes the weight mapping error in the tuning phase and average error compensation that prevents the propagation of error through cascaded layers. High precision individual-device tuning accuracy (<1%) is experimentally showed for both devices, but passive memristors suffer from half-select disturbance due to the lack of selector. This issue is studied comprehensively in chapter 3.

2.1. Neuromorphic Benchmarks

Fig. 5 shows the architecture of the neuromorphic benchmarks. The ConvNet model is based on Lenet-5 [83] architecture that includes 6 layers: Conv1, a convolutional layer with 5×5 filters and 65 feature maps; Pool1, a max-pooling layer of 2×2 regions; Conv2, a convolutional layer with 5×5 filters and 120 feature maps; Pool2, a max-pooling layer of 2×2 regions); FC1, a fully connected layer with 390 neurons; and finally FC2, a fully connected layer with 10 output neurons. Batch normalization is applied after each non-pooling layer, and rectified linear is used as the activation function in all the layers. The CIFAR-10 dataset consists of 60k 32×32 color images in 10 classes, with 6k images per class. The model is trained with 50k images and tested on the remaining 10k images of the dataset.

Standard data augmentation techniques such as zero-padding with two pixels, cropping a random 32×32 region, and performing random horizontal flipping of images are employed. No mean subtraction is performed (all input values are positive). We use ADAM optimizer, cross-entropy cost function, a batch size of 64, a learning rate of 0.001, and 220 epochs. Model initialization is performed following suggestions in [84].

The ResNet-18 implementation is based on the pre-trained model available at the official model zoo of the Pytorch. It includes 21+2 layers: a convolutional layer with 7×7 kernels and stride of 2, a max-pooling layer with 3×3 kernels and stride of 2, 4 convolutional blocks with residual connections, each including 4 convolutional layers based on 3×3 kernels and strides of 2 and 1, a 7×7 average-pooling layer with the stride of 7, and finally a 512×1000 fully-connected layer that provides the output prediction corresponding to 1000 classes. The network is tested on 50k images and trained on ~1.3M images for 150 epochs with a batch size of 256, the learning rate of 0.1 that is divided by 0.1 every 30 epochs (step scheduling), cross-entropy cost function, weight decay of 0.0001, and stochastic gradient descent optimization with a momentum of 0.9. The two models are trained using 32-bit floating-point precision on Nvidia Titan X GPUs, and the learned parameters achieving the highest test accuracy are used as the baseline model. During the mixed-signal simulation, we convert weights into device conductance/current, incorporate the developed models and techniques in the simulation platform and baseline architecture, and execute training and inference tasks. Note that we have not mapped the network into any mixed-signal architecture (e.g., see [82]) since simulating the targeted massive benchmarks within these (mixed-signal) architectures is incompatible and practically impossible with current GPU platforms and will make our results architecture-specific.

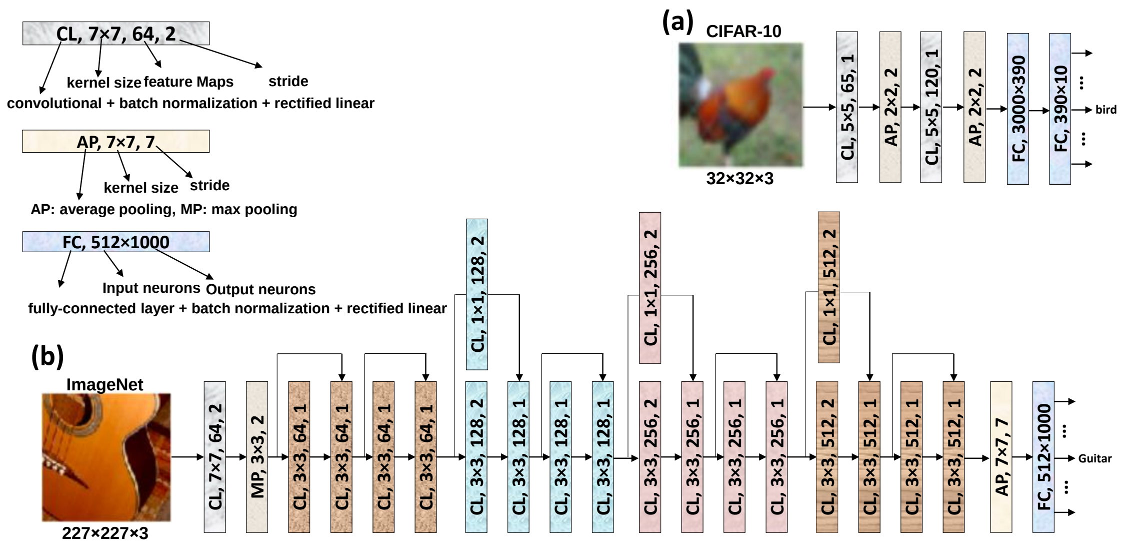


Fig. 5: (a) A 6-layer convolutional neural network trained on the CIFAR-10 dataset. (b) The ResNet-18 model trained on the ImageNet dataset.

2.2. Device Characterization

Two mainstream driving force technologies in neuromorphic circuits are emerging memristive crosspoint devices and industrial-grade redesigned eFlash memories [85]. The excellent density and scaling prospects of the former enable the efficient implementation of large DNNs. However, the slow advancing pace of this technology signifies immense fabrication challenges, e.g., high uniformity requirements in the *IV* characteristics of memristors. In [32], a successful development 64×64 passive crossbar circuit with record-breaking ~ 99% yield and < 26% normalized uniformity based on a foundry-compatible fabrication process is reported. As evidenced by the promising results from the recent demonstrations of large-scale neural networks [86], the situation is much better for floating-gate devices due to the availability of industrial-grade eFlash embedded in most CMOS processes.

A comprehensive characterization of imperfections in both memory technologies is initially performed. The experimental measurements are then used to model the average behavior of the devices and circuits. A unified parameter to describe major nonidealities in both synapses is used: the relative error of the state current, $\Delta I/I_0$, where I_0 is the reference tuning current measured at the nominal biasing condition, and ΔI is the current deviation from the ideal behavior. The models are then incorporated into simulation platforms (PyTorch-based libraries) to predict the fidelity loss in the benchmarks.

Fig. 6 shows the scanning electron microscope image of the fabricated crossbar that includes 4096 TiO_2 memristors–see Ref. [35] for more details on the fabrication process and relevant details on electroforming, tuning, and operation procedures.

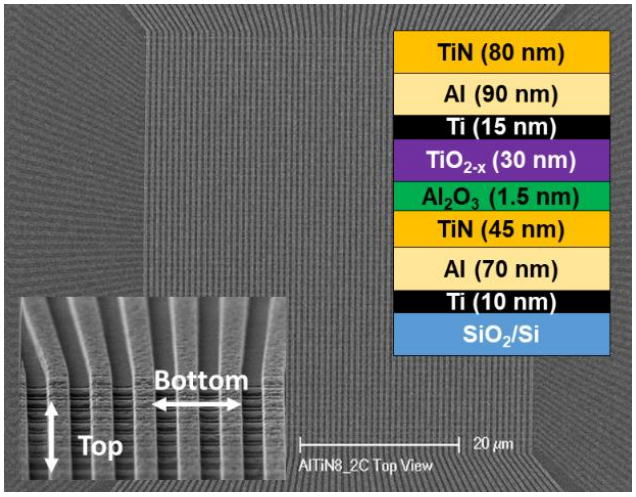


Fig. 6: SEM image of the full 64×64 memristor crossbar array [35]. The bottom left, and bottom right insets show material layers at the device cross-section with corresponding thicknesses in nanometers and zoomed-in to a portion of the crossbar, respectively.

Fig. 7a shows the measured *IV* characteristics of 350 randomly selected devices in the non-disturbing low-voltage regime. Upon the application of a voltage in this regime (<0.5 V), the conductance (state) of crosspoint devices remains unchanged at a fixed voltage. However, due to the tunneling or thermionic emission charge transport mechanism, the devices become more conductive in higher voltages and hence nonlinear. Fig. 7b shows the average relative static nonlinearity error versus applied voltage for various conductance states. It is computed with respect to the measured state current at the tuning voltage (V_{tune}) of 60 mV and 0.1 V maximum voltage.

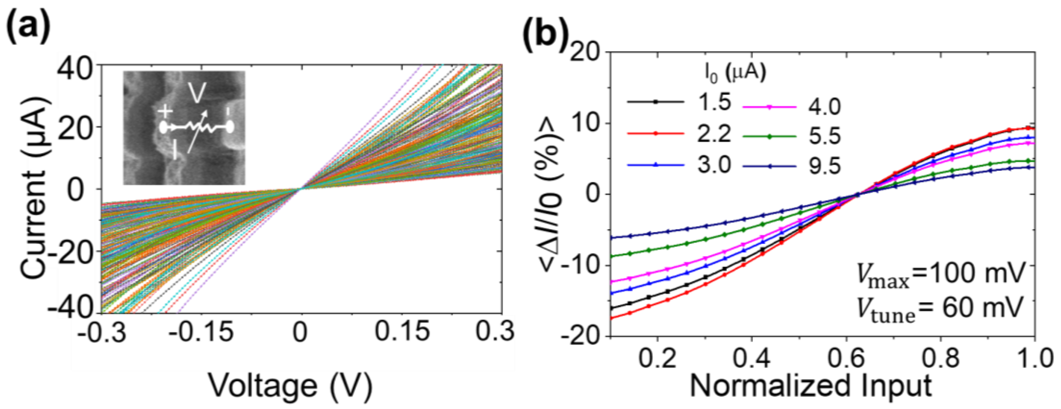


Fig. 7: (a) Low-voltage IV characteristics of 350 devices programmed to various states. The inset figure shows the cross-section of a device. (b) The average relative static nonlinearity error in memristor synapses $\Delta I/I_0$ ×100 for the same 350 devices, tuned in various states.

Fig. 8 shows the measurement results for the relative changes of conductance in 350 devices concerning variations in the die temperature (25–100 °C). The device conductance has proportional to absolute temperature and complementary to absolute temperature dependency in low conductive and high conductive states, respectively, due to the insulator-metal phase transition. In the case of our memristive devices, such transition occurs at ~70 μS, on average. A large error, particularly in low conductive states, is observed, which could severely degrade the computational accuracy of mixed-signal models at elevated temperatures.

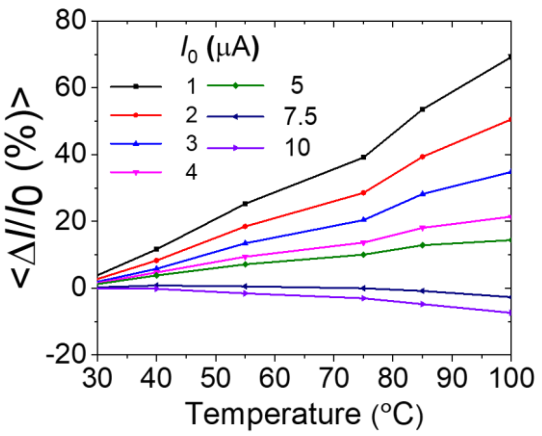


Fig. 8: The average relative change in state current versus temperature for 350 memristive devices tuned to various states.

The switching characteristics of memristors determine how precisely their conductances can be adjusted. Individually, a device can be tunned with high accuracy, e.g., <1% relative error regardless of its initial conductance. The experimental results in Fig. 9a corroborate this observation on 50 randomly selected devices tuned to 1.7 μs, 50 μs, and 10 μs conductances consecutively. For each device, the desired accuracy is achieved in less than 100 pulses using a naive write-verify algorithm. However, tuning dynamics are more complicated at the crossbar level since the half-select problem imposes disturbance on already tuned 0T1R memristors. Using additional gate lines in active crossbars (1T1R) solves this problem at the cost of at least a two orders of magnitude increase in the cell size. Fig. 9b shows an example of the ultimate relative tuning error distribution after the entire 64×64 crossbar is programmed to the states that correspond to the grayscale quantized Einstein image [87]. The final tuning error distribution depends on the switching threshold distributions and the tuning algorithm. We return to this issue in the next chapter.

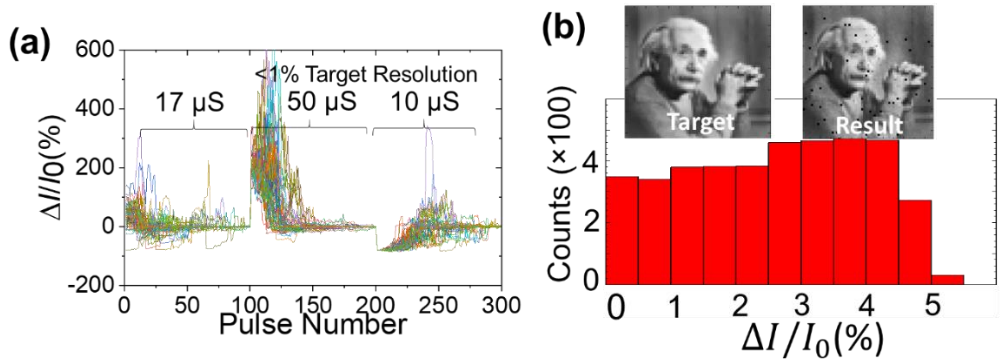


Fig. 9: (a) shows how relative error changes when devices are tuned using a write-verify algorithm with <1% target relative error. (b) The final tuning error distribution in 64×64 crossbar after all devices are tuned.

To investigate the impact of long-term retention loss, we perform accelerated retention tests and use the Arrhenius equation for the room temperature projection of the results. Fig. 10a shows the extremely stable analog-grade operation of 30 devices tuned in various states, subjected to 100 °C baking for >25 hours —translating into >14 years of room temperature operation assuming 1.1 eV activation energy [87]. Fig. 10b shows the distribution of relative retention loss error ($\Delta I/I_0$) for 400 memristors after 14 years of projected room temperature operation where I_0 is the initial sensed current for 400 memristors, each tuned to seven random states after projected 14 years of room temperature operation. Finally, Fig. 10c shows the corresponding standard deviation of the relative conductance change versus time binned to different states for these devices. The measured data show that the relative shift in conductance for most devices is expected to be <2% after several years of operation, which is adequately high for the practical implementation of ex-situ trained DNNs.

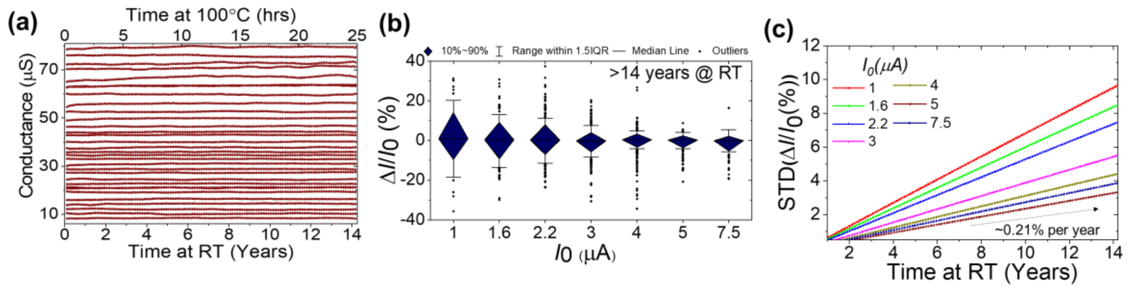


Fig. 10: (a) Stable analog operation after >14 years of room temperature operation. (b) The distribution of relative retention loss error ($\Delta I/I_0$ ×100). (c) the corresponding standard deviation of the relative conductance change versus time binned to different conductances. The conductance is measured at 0.1 V in these experiments.

More details of the statistical analysis of data for different states are provided in Fig. 11. Accelerated retention tests are performed at 100°C at 0.1 V for more than >25 hours. The results are then projected to room temperature using the Arrhenius equation and 1.1 eV activation energy. The insets show the histogram of the error for the case of 14 years. The results indicate that the retention loss is a bidirectional process for most devices and analog intermediate states, particularly midrange conductances. Unlike binary memristors [88], the distribution of retention loss error is relatively symmetrical in midrange analog states, i.e., the devices could move toward higher or lower conductive states. Note that we also observe unidirectional retention loss in very high (shifting toward low conductive states) or low (shifting toward high conductive states) conductance states, but we generally avoid switching the devices to extreme values. Nevertheless, the bilateral trend of retention loss of analog states is a positive feature since the tiny retention-induced errors become even smaller when they average out in large matrix multiplier kernels.

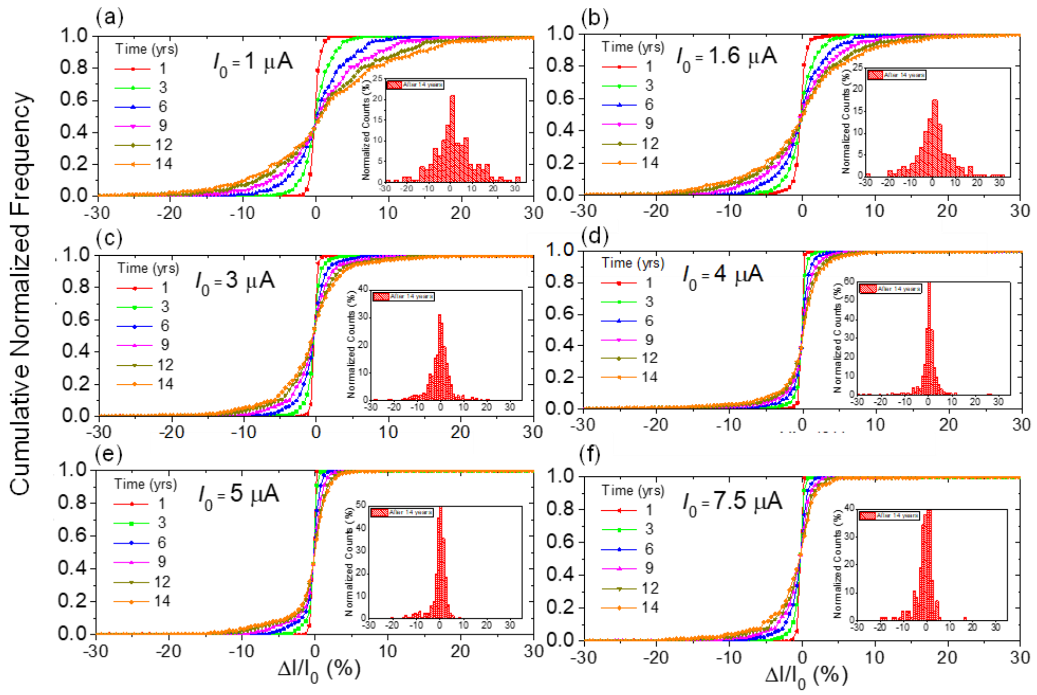


Fig. 11: Extended measurement results of accelerated retention test in memristive devices. Panels (a-f) show the cumulative normalized frequency of relative retention loss error among 400 devices tuned to various states.

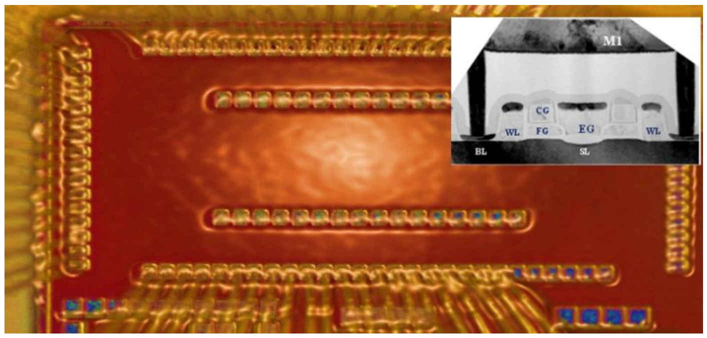


Fig. 12: Micrograph of the fabricated 12×10 eFlash array in Global Foundries' 55 nm CMOS process.

Fig. 12 shows the scanning electron microscope image of the fabricated redesigned eFlash memory array–see Ref. [35] for more details on tuning and operation procedure. First, we measure the average static input/output characteristics of 200 synapses in the gate-coupled structure (peripheral devices are tuned to the maximum state current, I_{max}=30 nA) and find the relative static nonlinearity error, which originates from the voltage-dependent capacitive coupling. Fig. 13a-b show the static nonlinearity measurement results for multiple synaptic weights. In Fig. 13b, the devices are tuned to the desired state at I_{tune}=I_{in}=21 nA.

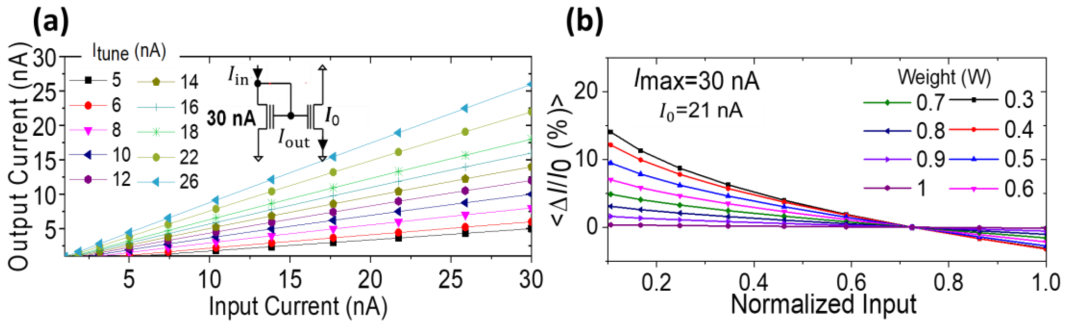


Fig. 13: (a) the average static input/output characteristics of 200 gate-coupled synapses given a peripheral cell tuned at 30 nA for various weight values. (b) the corresponding average nonlinearity error.

The temperature dependency of state current is also measured and demonstrated in Fig. 14a for 100 eFlash cells tuned to various states. The corresponding relative weight error in the gate-couple structure is also provided in Fig. 14b, indicating significant errors in high temperatures, which could significantly impact the accuracy of neural circuits.

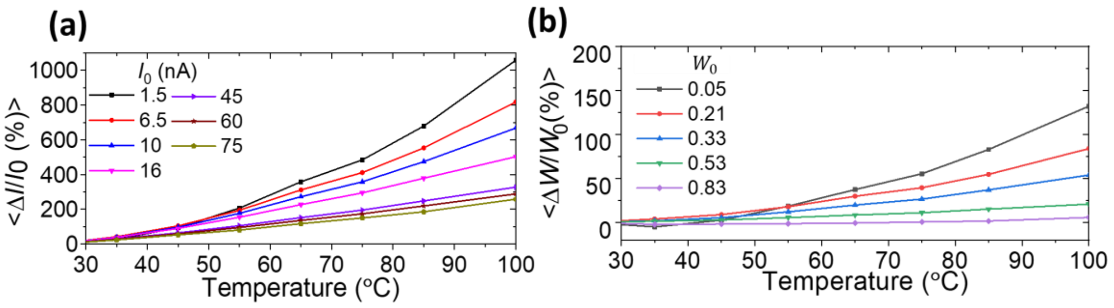


Fig. 14:(a) the average relative change in current measured using 100 devices, (b) the average relative change in the synaptic weight (assuming I_max=30 nA) of the gate-coupled structure versus temperature.

The retention characteristics of 100 eFlash memories are measured at 100°C. The measurements are performed by tuning the devices to different states within the relevant dynamic range. Fig. 15a shows the stable operation of 25 devices at 100°C for >6 hours. Regardless of the initial state, we confirm that the relative state change for most devices is comparable with the noise floor of the measurement setup (Fig. 15b). This superior performance partially stems from much effort spent on optimizing the technology for industrial-grade applications.

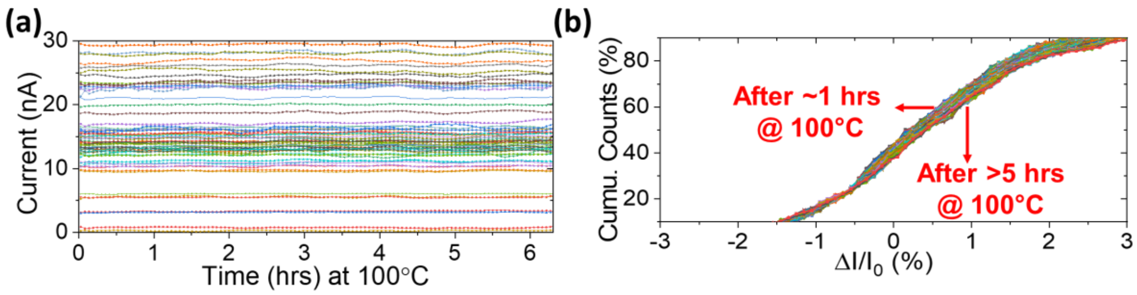


Fig. 15: (a) Accelerated retention test for 100 eFlash devices tuned in 5 different states measured at 100°C and nominal tuning conditions. (b) the trend in the cumulative distribution function of the relative change in the current (@100 °C) for these devices. The relative change is within 1% for most of the devices.

Finally, in Fig. 16, the high precision tuning capability of eFlash memories is shown for 50 devices by tuning them with 1% targeted accuracy to 100 nA, 50 nA, 30 nA, and 15 nA, consecutively, each using up to 50 pulses.

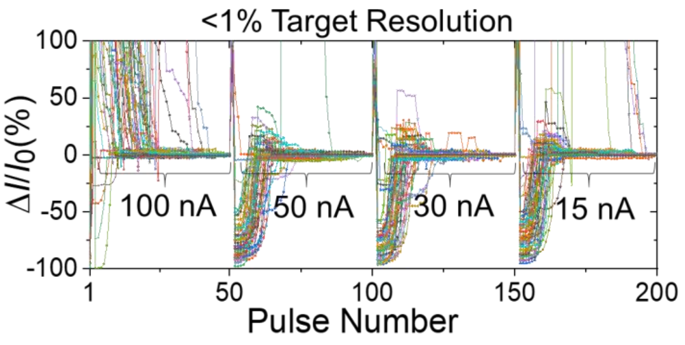


Fig. 16: High precision tunability (<1% target relative error) in 50 analog-grade redesigned eFlash memories, tuned to various target states.

The initial assessment of the experimental data indicates that the analog retention is promising in both devices; however, they are prone to variations in temperature that result in significant shifts in synaptic weights. Static nonlinearity is a fundamental bottleneck in most analog systems, and neural circuits are no exception. In both eFlash- and memristor-based neuromorphic systems, we need to optimize the circuit with respect to static nonlinearity. For redesigned eFlash cells, high precision tuning is obtained due to the redesigned memory cell [70], and excellent yield [86] is deemed due to the maturity of the technology. However, for passive memristors, the half-select disturbance bounds the weight tuning accuracy in neuromorphic circuits built with practically viable kernel sizes, and limited percent-scale yield is a major hindrance. These identified imperfections are then modeled to study their deleterious effect in massive neuromorphic networks simulated in the PyTorch environment.

2.3. Simulation Framework and Device Modeling

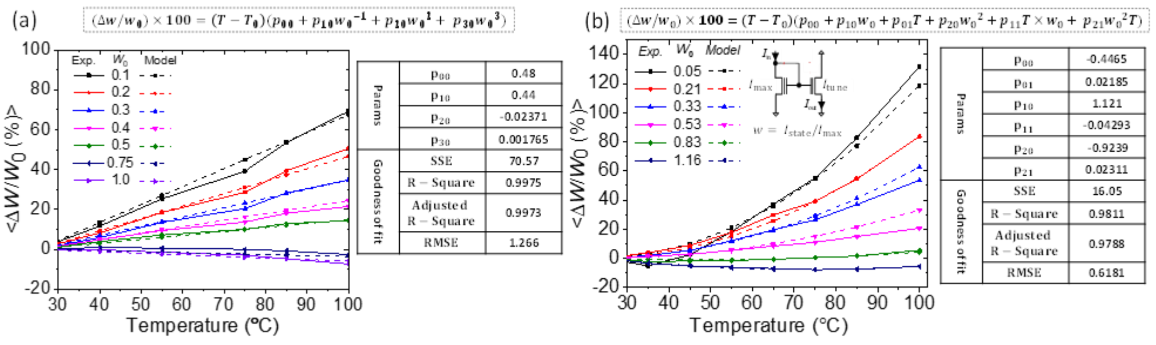
Fig. 17 elaborates on the phenomenological modeling procedure for the temperature dependency of eFlash and memristors. Instead of using complex physics-based models that would significantly slow down the simulation time in the massive neuromorphic benchmarks, multi-order polynomial functions that decently and efficiently predict devices' average behavior were used. In both cases, the most optimum polynomial function is found manually by an exhaustive brute force search, and nonlinear least squares optimization with a trust-region algorithm is applied to find the optimum fitting parameters.

2.3.1. Temperature

To study temperature variations, the relative change in the weight $(\Delta w/w_0)\ \times 100)$ of every device in a synaptic pair is modeled using $(T-T_0)(p_{00}+p_{10}{w_0}^{-1}+p_{20}{w_0}^{2}+p_{30}{w_0}^{3})$ for metal-oxide memristors and $(T-T_0)(p_{00}+p_{10}w_0+p_{10}T+p_{20}{w_0}^{2}+p_{11}T\times w_0+p_{21}{w_0}^{2}T)$ for eFlash memories in which w_0 is the measured weight at nominal biasing conditions and $T_0=25,\ T$ is the die's temperature in Celsius, and $p_{ij}s$ are the fitted parameters. The fitting results show excellent goodness of fit across the temperature range for both synaptic device candidates. In Fig. 17a, a weight exactly corresponds to a device conductance (in a synaptic pair and μS), i.e., $w_0=0.1$ and $w_0=1$ correspond to G_{min} and

G_{max}, respectively. In Fig. 17b, a weight corresponds to a device state (in a synaptic pair) over the peripheral device state, i.e., $w_0 = 0$ and $w_0 = 1$ correspond to $I_{state} = 0$ and $I_{state} = I_{max}$, respectively. Since the peripheral state is often tuned at I_{max} (that is 30 nA in this figure), w_0 equalizes the normalized weight. Note that in the case of eFlash memories, the model parameters change when a different I_{max} is used.

Note that most synaptic devices exhibit similar trends, and the similar modeling formats would apply to other devices as well. Only model parameters would be different. High-order nonidealities such as temperature dependency of nonlinearity, etc., are neglected in the simulations because they are far less impactful, and they typically devitalize each other, e.g., they become more linear and less noisy at elevated temperatures. Hence, they are neglected in our modeling here.



Params	p_{00}	0.48
	p_{10}	0.44
	p_{20}	-0.02371
	p_{30}	0.001765
Goodness of fit	SSE	70.57
	R – Square	0.9975
	Adjusted R – Square	0.9973
	RMSE	1.266

Params	p_{00}	-0.4465
	p_{01}	0.02185
	p_{10}	1.121
	p_{11}	-0.04293
	p_{20}	-0.9239
	p_{21}	0.02311
Goodness of fit	SSE	16.05
	R – Square	0.9811
	Adjusted R – Square	0.9788
	RMSE	0.6181

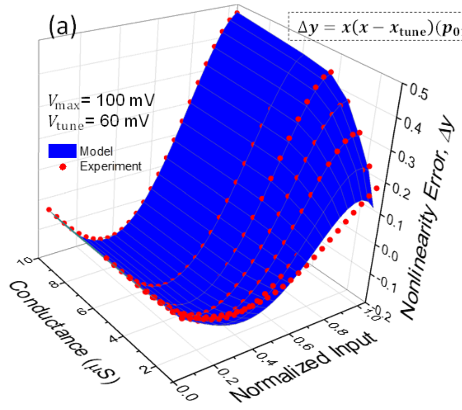
Fig. 17: Temperature modeling in analog-grade (a) memristors and (b) eFlash memories.

2.3.2. Nonlinearity

Fig. 18 shows high goodness-of-fit in modeling the static nonlinearity of both analog memory candidates and discusses how static nonlinearity varies with the tuning condition. Like temperature modeling, a multi-order polynomial function that perfectly describes devices' average behavior without slowing down massive neuromorphic networks' simulation time is used. In both Fig. 18a and 18b, the polynomial functions' shape is manually optimized and nonlinear least-squares optimization with a trust-region algorithm to obtain the model parameters is used.

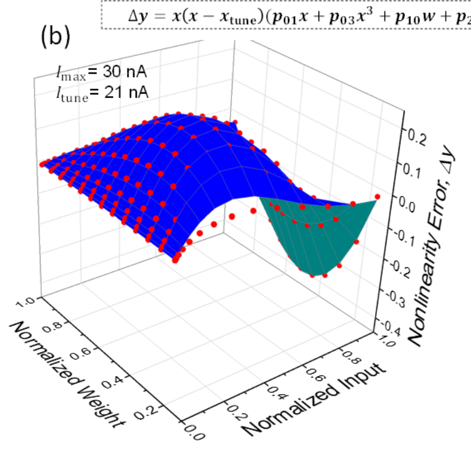
Note that to ease the network simulation, in this part, the nonlinearity error (not the relative nonlinearity error) is modeled. The amount of nonlinearity error is a function of both conductances of the device in the tuning biasing condition, the maximum applied input signal, and the applied input signals. Hence, to avoid complicating the nonlinearity model and enhance the fitting results, we decouple it from the tuning conditions and maximum applied input signals, i.e., we perform the modeling and find the parameters for each design case once (separately). Here, the results are provided for one case in memristive circuits and one case of eFlash designs. In the former, the error in the synaptic current of a device tuned to w (the conductance of a single device in the differential pair in μS) at the normalized input x_{tune}, when stimulated by x is modeled by $\Delta y = x(x - x_{tune})(p_{01}x + p_{03}x^3 + p_{10}w + p_{20}{w_0}^2 + p_{30}{w_0}^3)$. For the latter, since the gate-coupled structure is studied, using both normalized weights and inputs makes the modeling easier. Here, when a normalized input *x* is applied to a synaptic device, tuned to the normalized weight of *w* at the normalized input x_{tune}, it creates a nonlinearity error that can be obtained by $\Delta y = x(x - x_{tune})(p_{01}x + p_{03}x^3 + p_{10}w + p_{20}{w_0}^2 + p_{30}{w_0}^3 + p_{11}xw + p_{22}x^2w^2)$. In both models, $p_{ij}s$ are the fitted parameters that are provided in the inset tables. For memristors, the parameters correspond to the case with V_{max}=0.1 and V_{tune}=0.06, while for eFlash, I_{max}=30 nA and I_{tune}=21 nA, i.e., the devices are tuned at the condition in which the input signals are 0.06 V (for memristors) and 21 nA (for eFlash). The fitting results show excellent goodness of fit across for both synaptic device candidates.

$$\Delta y = x(x - x_{\text{tune}})(p_{01}x + p_{03}x^3 + p_{10}w + p_{20}w^2 + p_{30}w^3)$$



Params	p_{01}	1.368
	p_{03}	-1.662
	p_{10}	0.7345
	p_{20}	-0.1184
	p_{30}	0.006132
	x_{tune}	0.6076
Goodness of fit	SSE	0.0054
	R – Square	0.9990
	Adjusted R – Square	0.9990
	RMSE	0.0055

$$\Delta y = x(x - x_{\text{tune}})(p_{01}x + p_{03}x^3 + p_{10}w + p_{20}w^2 + p_{30}w^3 + p_{11}xw + p_{22}x^2w^2)$$



Params	p_{01}	3.109
	p_{03}	-1.222
	p_{10}	-12.71
	p_{20}	20.41
	p_{30}	-7.816
	p_{11}	-2.733
	p_{22}	0.9496
	x_{tune}	0.7247
Goodness of fit	SSE	0.002536
	R – Square	0.9995
	Adjusted R – Square	0.9995
	RMSE	0.002665

Fig. 18: Modeling nonlinearity for (a) passively integrated memristive devices and (b) redesigned eFlash memories.

2.4. Simulation Results

2.4.1. Temperature Variations

Temperature variations have the most drastic impact on mixed-signal neuromorphic circuits. The synaptic weights change dramatically with temperature, modulating the preactivation signals of the neurons. Fig. 19a shows how the preactivations received by the first neuron in the fully-connected layer of ResNet-18 change with the temperature. The modulation of the preactivation distributions occurs in all layers and neurons but with different rates. Fig. 19b shows the temperature dependency of multiple percentiles of the preactivation distributions in 2 different layers. Interestingly, such shifts are almost monotonic in most neurons, partly because the conductance of synaptic devices (eFlash or memristors) changes monotonically with respect to the temperature.

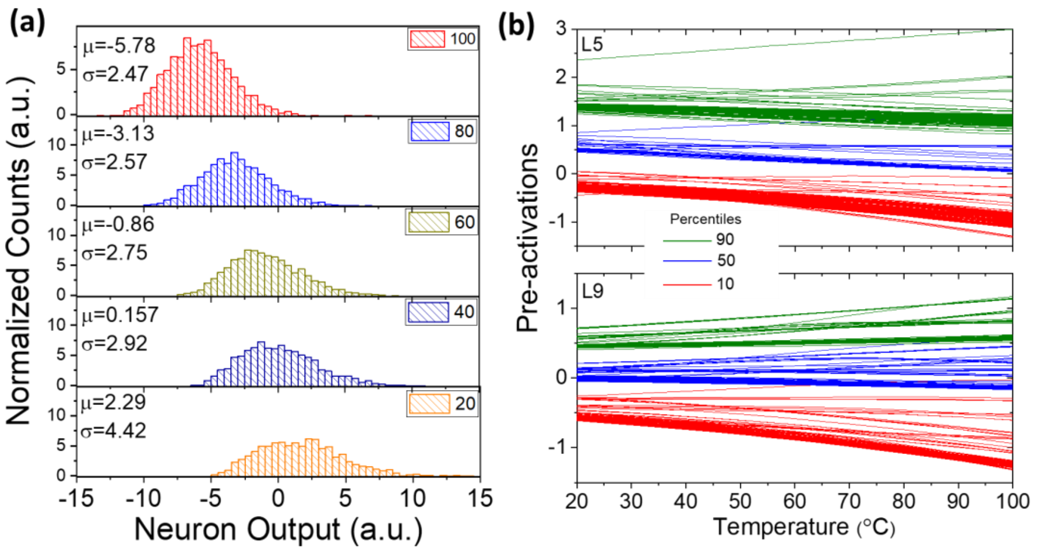


Fig. 19: (a) The distribution of the first neuron's output activations in the fully-connected layer of ResNet-18 for 10 inference batches in several temperatures (FM2). (b) The temperature dependency of 10, 50, and 90 percentiles of the preactivation distributions (100 batches) of 100 random neurons in 2 different layers (RM1). FM2: eFlash, mapping 2. RM1: RRAM, mapping 1.

Fig. 20 also shows an almost monotonic shift in 10, 50, and 90 percentiles of preactivation statistics in 100 randomly selected neurons in various ResNet-18 layers and modulation rates. The statistics are obtained by processing 100 batches, and the temperature model of the RRAM devices using mapping 1. The proposed temperature compensation method consists of three incrementally applied approaches and aims to reduce the worst-case accuracy across the studied temperature range by modifying the circuit and training algorithm.

The first method is temperature-sweep batch training, in which we include the temperature model of synapses in the training process by considering a new hyperparameter called training temperature (T_θ). Before running each forward pass of the training, we assume the model is ready for deployment in a chip that operates at an ambient temperature T_θ and convert all weights to their corresponding synaptic current values. Using the device model, we adjust the resultant synaptic current values in every step based on the training temperature value. The altered synaptic currents are converted back to the equivalent software weights before the forward pass is executed. Triangular scheduling of the training temperature is adapted, i.e., T_θ is swept from 25 °C to 95 °C and vice versa by 10 °C steps in every batch. Fig. 21a shows the reduction of the worst-case accuracy drop for different stacks and mappings in ResNet-18. Baseline corresponds to the evaluation case without any mitigation technique, while approaches 1, 2, and 3 refer to temperature-sweep batch training, k-reference batch normalization, and state optimization methods. These techniques are applied incrementally on the network. In approaches 2 and 3, we use 4 reference points.

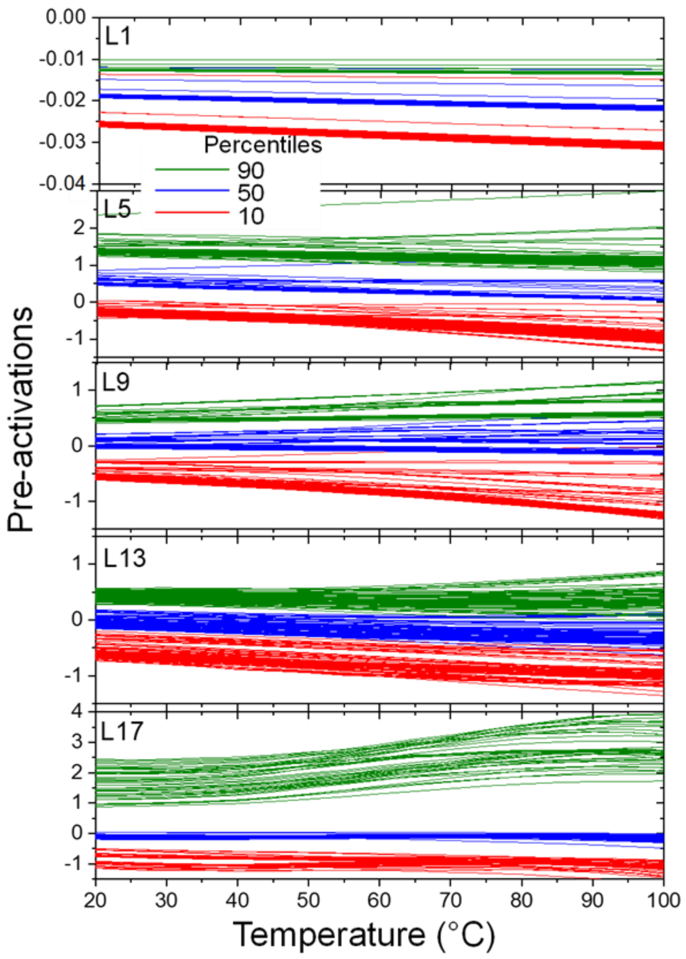


Fig. 20:Preactivation statistics versus temperature.

For example, for RM1 (RRAM stack, mapping 1), the worst-case drop (occurs at 100 °C) is reduced from ~66% to 23% after applying approach 1. The most optimum performance is also achieved in midrange temperatures (60 °C), as expected. The improvements in ConvNet are also encouraging since the worst-case drop is decreased from ~25% to 3.3% (for RM1), as shown in Fig. 21b.

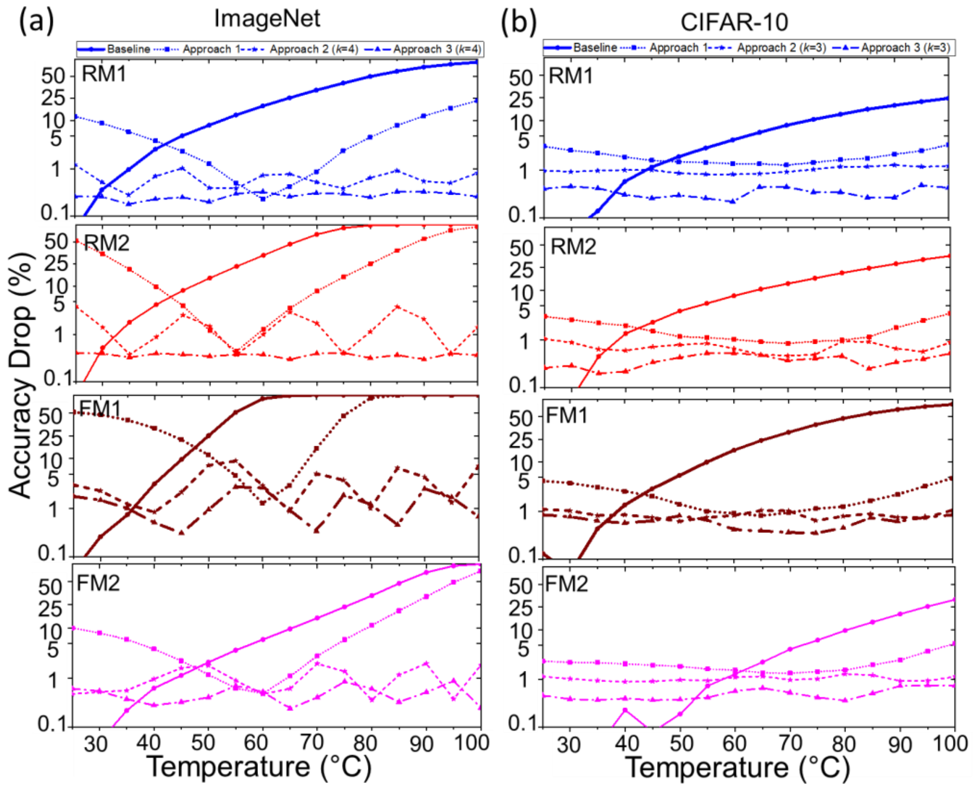


Fig. 21: The accuracy drop versus temperature in (a) ImageNet benchmark when using various synapse options to implement ResNet-18 model, (b) ConvNet to implement CIFAR-10. RM1: RRAM, mapping 1; RM2: RRAM, mapping 2; FM1: eFlash, mapping 1; FM2: eFlash, mapping 2.

Inspired by our work on increasing the reliability of hardware security primitives [89], *k*-reference batch normalization is adopted that further enhances the performance by using *k* temperature-optimized batch normalization parameters per neuron. Owing to the monotonic change in the statistics of preactivations (i.e., the shift and stretch of the preactivations) with respect to the temperature (Fig. 19a-b), a temperature-dependent correction signal allows us to minimize the induced error. Since generating such neuron-specific signals with adjustable temperature-dependency are costly, we use a quantized version of it through multiple batch normalization weights that effectively shift and scale preactivations. After the model is trained with the first approach, we find *k* reference batch normalization parameters by retraining it in a single epoch with a learning rate of 0.001 in *k* reference temperatures. During the inference, the temperature of the chip is sensed by a low-cost on-chip sensor and used to determine the proper batch normalization parameters that correct the distributions Fig. 21a shows a considerable reduction of the worst-case drop in the ImageNet benchmarks after applying the second approach (*k*=4), e.g., from ~23% to 1.25% for RM1. In the CIFAR-10 benchmark, the worst-case drop for RM1 cases decreases from ~3.3% to 1.22%, with only 3 reference points.

As depicted in Fig. 22a (for ResNet-18) and Fig. 22b (for ConvNet), the results can be further improved by increasing the number of reference points; a sub-percent accuracy drop is achieved with a few references (depending on the stack and mapping). Note that the model is still trained ex-situ entirely with a negligible overhead (2*k* parameters per neuron). Though the second approach significantly reduces the worst-case accuracy drop, if needed, the results can be improved even further by optimizing the weight mapping parameters (I_{min}, I_b) for each weight.

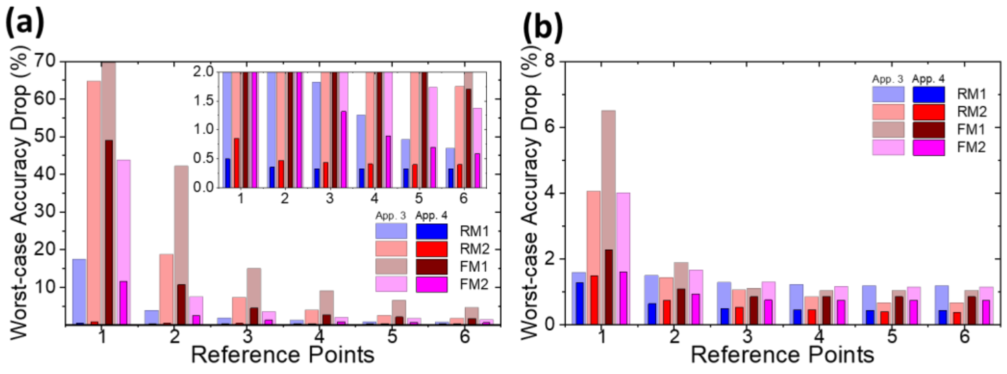


Fig. 22: The worst-case accuracy drop in the 20-100 °C temperature range versus the number of reference points for approaches 2 and 3 for (a) the ResNet-18 and (b) the ConvNet model.

The state optimization approach is the third technique that mitigates the accuracy drop in a wide temperature range. Here, the mapping parameters are optimized individually for every weight targeting the lowest weight error across the full temperature range. Such design parameters are often selected to minimize the power consumption in eFlash memories or maximize the dynamic range in memristors. However, these design parameters are not necessarily the most optimum in terms of temperature variations and reliability. Since this approach comes with power consumption addition or dynamic range reduction, a methodology that finds quasi-optimal design points in either weight-conductance mapping functions is developed.

A numerical analysis of the experimental data is provided in Fig. 23. It shows a procedure for finding the quasi-optimum design parameters of each device stack and weight mapping functionality. The panels in the first column of Fig. 23 show the normalized energy consumption in synaptic arrays for a network of 10M normally distributed weights versus the dynamic range (ΔI_{max}). The choice of the initial design point (ΔI_{max}, I_{min}, and I_b) is often in the direction of minimizing the energy consumption in eFlash circuits and maximizing the dynamic range in 0T1R memristive systems, regardless of the mapping type (the red star in the first column panels shows the initial design point).

However, the optimum sensitivity concerning temperature variations is not necessarily this design point. Since we intend to apply a secondary cost-free technique to further compensate for temperature variations, the goal in this step is to trade energy or dynamic range to improve the accuracy and find a quasi-optimum operating point that is less sensitive to temperature variations. To find a quasi-optimum design point, the cost function $C = \max_{T_0 \leq T \leq T_{\max}} \int_{\epsilon^+}^{1} |E_r(W,T)| dW$ is defined, which represents the worst sum of relative errors among all (normalized) weights across all temperatures. To numerically compute the cost function, we use $T_0 = 25$, $T_{\max} = 100$, and $\in^+ = 0.01$. The minima of the cost function give the optimum design point averaged over the weights.

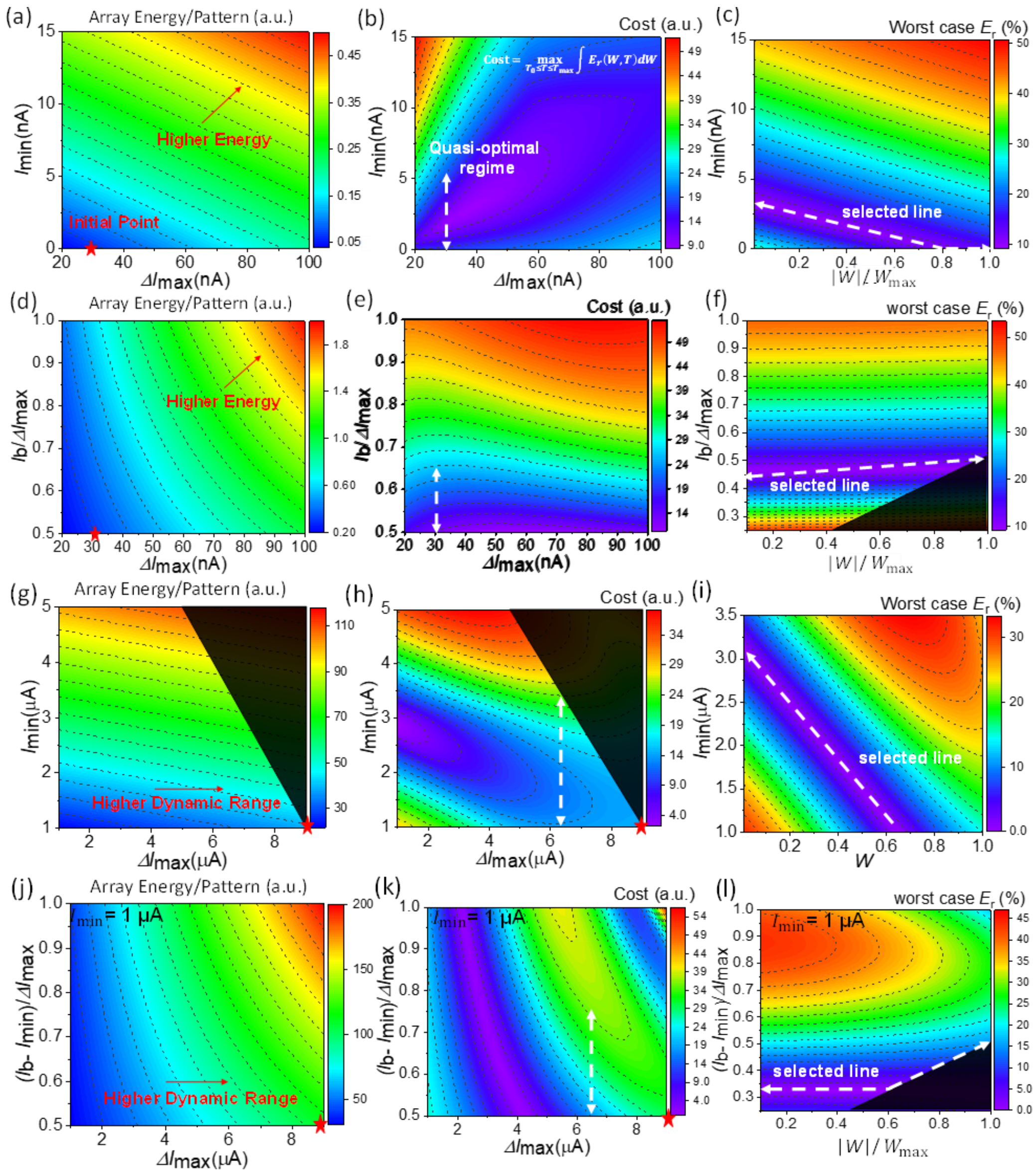


Fig. 23: State optimization for temperature sensitivity. Panels (a-c), (d-f), (g-i), and (j-l) show the state optimization simulation results corresponding to mapping 1 of eFlash, mapping 2 of eFlash, mapping 1 of memristors, and mapping 2 of memristors, respectively. The shaded areas denote out-of-range regimes.

The panels in the second column show how C changes across the design space. Based on the heatmap of C, ΔI_{max} can be selected close to the minima without overspending on energy (in eFlash) or dynamic range (in memristors). By definition, ΔI_{max} is weight independent; however, the other design parameter (I_{min}or I_b) may be optimized at the cost of slight power increase or dynamic range reduction.

Unlike previous works that choose a fixed minimum current or bias current for all weights, a more optimum weight-dependent choice of minimum synaptic current (mapping 1) and bias current (mapping 2) are found by using third column panels that show the heatmap of the worst-case relative error across all temperatures versus normalized weight. Panel (b) shows the cost function for eFlash mapping 1. A white dashed line (ΔI_{max}=30 nA) indicates a quasi-optimal regime that features low energy and is close to the minima of C. The error is further optimized by finding an optimum weight-dependent I_{min}. Panel (c) shows that the worst-case error is minimum when I_{min} (nA) $= \max(0, 3 - 3.75(|w|/w_{max}))$. For mapping 2 (second row panels), it is observed that the cost function and energy are both minimized when the minimum bias current is used, i.e., $I_b = \Delta I_{max}/2$.

To minimize power, ΔI_{max} = 30 nA (the same as mapping 1) is used, and the optimum bias current for a given weight is obtained by I_b (nA) $= 2.35\ (|w|/w_{max}) + 12.65$. The same procedure is used for the memristors, and similar results are obtained. Operating eFlash in deep weak inversion enables low power operation and high dynamic range. Hence, trading a slight increase in energy consumption for improved reliability makes a lot of sense. Unlike eFlash devices (at least in the present technology), metal-oxide memristors are more power-hungry and have a limited dynamic range, limiting the options for finding the quasi-optimized state. In mapping 1 of memristors (panel h), it is observed that the minimum cost is obtained in a region that has a very low dynamic range (which is impractical to tune the weights and realistically map the weights to it). Instead of using a low dynamic range, a practically viable dynamic range (6.5 μA) is chosen and 3.5 μA for finding an optimum weight-dependent I_{min} is reserved. Panel (i) shows that I_{min} (μA) $= \max\,(0, 3.1 - 3.23(|w|/w_{max}))$ is the weight-dependent quasi-optimal equation for our devices for the 6.5 μA dynamic range. Similarly, 6.5 μA dynamic range is selected for mapping 2, and the optimum bias current per weight is obtained by I_b (nA) $= \max\,(3.1, 1.48 + 2.76\,(|w|/w_{max}))$.

The state optimization approach, combined with temperature-sweep batch training and k-reference batch normalization, recovers the accuracy drop significantly across the entire temperature range regardless of selected device or mapping. The worst-case accuracy drop in the full temperature range diminishes to ~0.4% in ResNet-18 (k=4) and ~0.49% in ConvNet (k=3) in the RM1 case. Fig. 22a-b highlights that a sub-percent accuracy drop is easily feasible across the full temperature range in both benchmarks after applying the temperature compensation techniques.

2.4.2. Defect Tolerance

Two techniques have also been adopted that increase the resiliency of the mixed-signal hardware against defective devices. Note that the information that a specific device is defective is only available during the tuning phase. Fig. 24 shows how the accuracy drop increases with the surge of faulty devices. Specifically, when using mapping 1, the network becomes more sensitive to devices stuck at high conductance and less susceptible to stuck at low conductance. This stems from the fact that the weight distribution in these benchmarks is such that most devices are tuned near the reset state for mapping 1 and near the midrange state for mapping 2. In Fig. 24a and 24d, the defective devices are stuck at high conductance ($G_{\max}$).

In Fig. 24b, the defective devices are stuck at low conductance (G_{min}), and in Fig. 24c and 31f the conductance of faulty devices are uniformly distributed in the considered conductance range (G_{min}<G_{x}<G_{max}). For every point, the statistics are obtained over 20 runs.

Fig. 25 compares the results when considering all three fault cases happening with equal probabilities and shows that mapping 2 outperforms mapping 1 (since the error distribution is statistically smaller). In the first approach, it is exploited that each weight is mapped to a pair of memory devices, and regardless of the mapping function, either of the devices can be returned to minimize the mapping error. When $G_{\pm}$ is stuck to G_{max}, we use $G_{\mp} = G_{\text{max}} \mp \Delta G_{\text{max}}((W \pm |W|)/2|W|_{\text{max}})$ for pair-device retuning. When $G_{\pm}$ is stuck to G_{min}, $G_{\mp} = G_{\text{min}} + \Delta G_{\text{max}}((|W| \mp W)/2|W|_{\text{max}})$ is used to retune the paired device. When $G_{\mp}$ is stuck to $G_{\text{min}} < G_{\text{x}} < G_{\text{max}}$, $G_{\mp} = G_{\text{x}} \mp \Delta G_{\text{max}}(W/|W|_{\text{max}})$ is (clipped to G_{max} or G_{min}, if needed) used to retune the paired device. Fig. 24 shows the improvement achieved by this technique in every fault case individually, and Fig. 25 shows the result of the general case. In a fabrication process with 2×10^4 ppm defective devices, both ResNet-18 and ConvNet generate almost entirely random classes without applying this technique. The proposed method diminishes the accuracy drop to only 14.3% for ConvNet and 23.4% for ResNet-18.

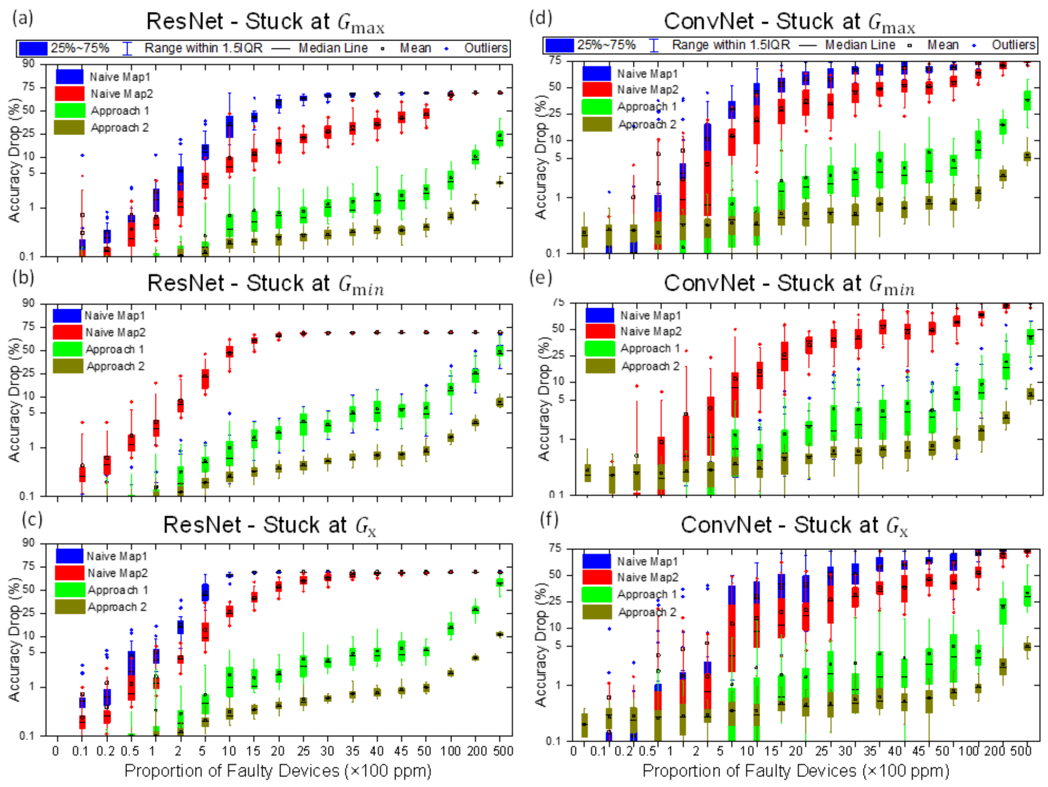


Fig. 24: Yield Analysis simulation results. Fault-tolerance analysis in (a-c) ResNet-18 and (d-f) ConvNet.

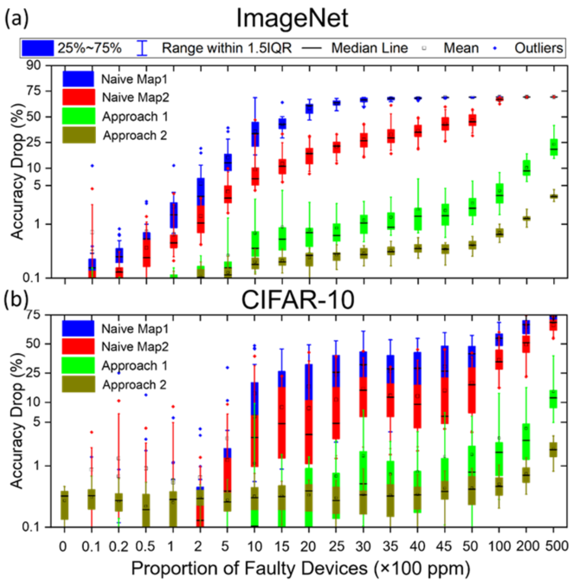


Fig. 25: Defect-tolerance improvements in (a) ResNet-18 and (b) ConvNet using the two incrementally applied approaches.

Clearly, if two defective devices constitute a synapse, it is not feasible to compensate for its weight mapping error. Besides, the limited dynamic range of preactivation makes them susceptible to a small constant shift in a synapse output. The second method alleviates this issue by compensating for such shifts through an extra pair of analog memories (single column) per neuron per processing kernel. Such devices are always driven by a fixed maximum range signal, and their states are adjusted during the tuning phase to minimize the average shifts. To automate the procedure, after the pair-wise tuning is performed on a defective model, we use a tiny part of the training set ~7k and 1.5k images in ResNet-18 and ConvNet, respectively, to recompute the biases in batch normalization layers (that simply shift the preactivation signals) and find the conductance of extra devices.

Note that the area overhead of this method is negligible (unlike previous attempts to overcome this issue by adding redundancy), as there is no need for an additional or general-purpose routing at the input or output of the kernels. For the same case, this method reduces the accuracy drop to 0.3% for ConvNet and 3.2% for ResNet-18. Simulation results in Fig. 25a and 25b indicate that, for a sub-percent average accuracy drop, these two (low overhead) techniques enable tolerance of ~1.5×10^4 ppm defective devices in ResNet-18 and ~3×10^4 ppm faulty devices in ConvNet, both numbers >100× better than the initial resiliency.

2.5. Discussion

The results presented in this study establish robust predictions on the performance of analog neuromorphic networks in the presence of detrimental imperfections. So far, research in this area has focused on commercially unscalable techniques such as in-situ or chip-in-the-loop methods. Other than that, most previous works study the impact of a single nonideality on redundant networks using small datasets solely based on the simulations or data from practically nonviable devices. This work performs a comprehensive characterization of major imperfections in the most prospective analog-grade memory devices. Characterization results are then harnessed to develop accurate device models, which are then incorporated to train and test two massive DNNs. The experimental work confirms that the synapse imperfections are major obstacles in the path of further progress of mixed-signal neuromorphic systems. We show that eFlash and TiO_2 memristors have excellent retention characteristics and tolerable static nonlinearity. Using the balancing technique methodology [34], which optimizes the tuning voltage for minimum error, we report only $< 0.4\%$ and $< 0.1\%$ accuracy drop for RM1 and RM2, respectively, after including the static nonlinearity model in the forward pass of the ResNet-18.

Temperature variations intrinsically change the state of any analog synapse and dramatically impact the performance. A naively designed mixed-signal DNN could randomly behave when operating at 100 °C. We propose three modifications in the training (temperature-sweep batch training), circuit (k-reference batch normalization), and tuning (state optimization) for designing reliable neuromorphic hardware that can operate in a wide temperature range. The incremental incorporation of these techniques enables a sub-percent accuracy drop even in a complex classification task such as ImageNet.

Further, this study shows that the intrinsic defect tolerance of deep neural networks falls short in larger and more complex tasks: with >500 ppm defective devices, the accuracy drop increases drastically beyond 1%. For a mature technology like eFlash, the fault probability is well below this intrinsic range, while for the emerging passive RRAM, we introduce two approaches, both applied during the tuning phase, to enhance the margin by a factor of >100×.

The passive RRAM technology offers the highest device density and monotonic 3D integration.

Although the proposed approach is examined using two specific memory technologies, it is not tied down to particular features of these devices. Hence, this holistic approach could be applied in any mixed-signal neuromorphic implementation. For any memory technology, whether it is a FET-style synapse like eFlash or a resistive switching device like our memristive stack, imperfections may be modeled and included in the process of developing, training, and tuning the neuromorphic network. This study is also decoupled from the choice of a mixed-signal architecture in part because changing the structure of these massive networks in the simulation environment has a significantly destructive impact on the inference and training runtime of the model. Besides, the impact of the studied imperfections is expected to be the same in different architectures, and our holistic approach does not depend on a specific feature of the mixed-signal accelerators.

High-order nonidealities such as temperature dependency of static nonlinearity, noise, etc., are neglected in our simulations because they are far less impactful. Besides, although the proposed techniques are analyzed and simulated individually, they are entirely independent and could be applied together. Nevertheless, in many cases, imperfections devitalize each other, e.g., memristive devices become more linear and less noisy at elevated temperatures.

The IR drop [55] is neglected in our study because it is nearly impossible to simulate its effect in large-scale neuromorphic systems. Ref. [34] proposes a bootstrapping method that effectively tackles it at the expense of monopolizing two CMOS metal layers. Ref. [57] uses an efficient conversion algorithm to mitigate the impact of IR drop, and Ref. [58] resistance in peripheries to equalize the parasitic resistance seen by all the devices. The impact of endurance failure is not covered in this study since endurance requirements for ex-situ training of mixed-signal neuromorphic circuits are relaxed (e.g., <105) compared to in-situ approaches that rely on frequent write operations and most nonvolatile memories, including the demonstrated devices in this study, can offer such specifications.

In these studies, we found that when no particular technique is used to mitigate imperfections, mapping 2 outperforms mapping 1 in terms of reliability at the cost of extra energy consumption. However, the proposed holistic approach allows us even to employ mapping 1 for weight to conductance conversion and saves extra energy that was previously inevitable. The most appealing feature of this approach is its scalability and the fact that it can be easily integrated into the design flow of these massive systems. The modifications performed in the training phase do not require any specific knowledge of imperfections (e.g., location of faulty devices) or individual chips and could be integrated with the typical ex-situ training procedure. The circuit modifications include additions of a simple temperature sensor circuit, low-cost hardware to support multiple batch normalization parameters per neuron, and an extra column in each kernel, with the total overhead that barely reaches 1% of an entire DNN chip.

The state optimization and advanced tuning algorithms also do not require any extra hardware and are applied simply for every chip during the ex-situ tuning. Although the proposed approach might slightly increase the training time, for most of the networks, the extra imposed training time is comparable with the training time of the baseline model, which is also negligible since training is performed only once in ex-situ trained systems and the developed model is used for a generation of deployed mixed-signal inference accelerators.

In conclusion, we have performed extensive characterization of imperfections in mainstream analog-grade synaptic devices and developed a holistic hardware-aware ex-situ approach to combat their detrimental impact on the performance of DNNs.

The proposed approach includes modifications in training, circuit, state optimization, and tuning algorithm and has minimal areal or power overhead. The proposed methods are successfully tested on two large-scale deep neuromorphic networks. We believe that the results significantly improve the accuracy and efficiency of mixed-signal DNNs. Future research should focus on developing generalized device models to evaluate the effectiveness of our approach as a general solution and implementing the proposed methodology in fully-integrated neuromorphic circuits.

3. The Impact of Device Uniformity on Functionality of Memristive Circuits

Despite all the rich properties, experimental demonstrations of passive memristive crossbars have been limited to circuits with a few thousand devices until now, which stems from the strict uniformity requirements on the *IV* characteristics of memristors [90]. The stochastic nature of oxide rupture in such small scales complicates the reproducibility of device parameters, e.g., the voltage required for electroforming and switching. Indeed, such variabilities are the very reason for the limited demonstrations of memristive neuromorphic networks so far. One solution to alleviate this issue is the usage of selector transistors (1T1R memories), which is inconsistent with the main driving force of this technology (i.e., scalability and three-dimensional integration compatibility [41,91]).

The recent work [32] overcomes this challenge and demonstrates the successful integration of a 64×64 passive metal-oxide memristor crossbar circuit. This technology features analog-grade memories with ~99% device yield based on a foundry-compatible fabrication process with etch-down patterning and a low-temperature budget, conducive to vertical integration. The crossbar also features excellent analog properties such as long retention and high endurance characteristics. The cell size is 10^4× denser at a similar yield, and the average conductance is 10× less than state-of-the-art 1T1R technology [57,92]. Besides, the reported uniformity is sufficient for <5% average tuning precision that is slightly worse than ~3% reported in analog 1T1R memories [92].

Despite the vital importance of uniformity in 0T1R memristor crossbars, it has not been thoroughly investigated in the context of neuromorphic computing to the best of our knowledge. The key open questions include: How does the crossbar uniformity impact the computing accuracy of memristive crossbars? From this perspective, what are the critical factors that affect computing accuracy? How can we improve the performance? How much crossbar uniformity is needed to achieve software-equivalent accuracy and build a large-scale deep neural network? This study aims to expand upon these important questions and the critical role of switching threshold variations in the computing precision of neuromorphic networks.

In this chapter, first, we discuss the preliminaries, motivation, and previously fabricated analog-grade memristor crossbars. Then, we use a dynamic model based on fabricated crossbars in UCSB and develop a simulation framework to study the aforementioned problems. Further, extensive simulations of VMMs and representative neuromorphic networks that are performed to assess the tradeoffs and trends are introduced. All simulations of matrix multipliers and deep neural networks on CIFAR-10 and ImageNet datasets have been carried out to evaluate the role of uniformity on the accuracy of computing systems. Finally, we study three post-fabrication methods that increase the accuracy of nonuniform 0T1R neuromorphic circuits: hardware-aware training, improved tuning algorithm, and switching threshold modification. Applying these techniques allows us to implement advanced deep neural networks with almost no accuracy drop, using current state-of-the-art analog 0T1R technology.

3.1. Passive Memristive Crossbars

3.1.1 Preliminaries and Motivation

Fig. 26 shows the scanning electron microscope image of the latest fabricated 64×64 memristive crossbar in UCSB [32]. The inset shows the zoomed-in view to a portion of the crossbar, showing top electrodes passing on top of the bottom electrodes. A memristive device is located at every intersection of each top/bottom electrode. Such an array of conductance-adjustable devices could be used to implement vector-by-matrix operation in the analog domain by utilizing Ohm and Kirchhoff laws [85].

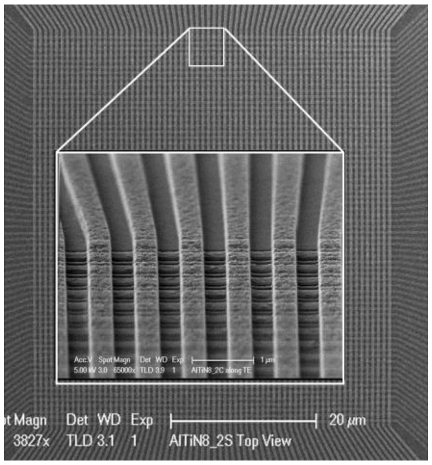


Fig. 26: The SEM image of the fabricated 64×64 crossbar [32]

In this study, we are interested in investigating how the parameters of a 0T1R memristor technology, i.e., the variations in the switching thresholds, impact the target accuracy (∈) and, in turn, the computational accuracy of memristive neuromorphic networks. When a high-precision readout circuit is available and memristive devices have excellent retention characteristics, ∈ is almost entirely bounded by the dynamic switching characteristics of the devices.

To clarify this, consider the practical *V*/2 approach [32,90] of tuning memristive crossbars (Fig. 27a). The voltage applied on the selected device (by peripheral decoders and switch matrix) is V_{set}. Unselected electrodes are pinned to $V_{\text{set}}/2$ to minimize the disturbance on other devices. The applied voltage on the unselected devices is zero; however, $V_{\text{set}}/2$ is dropped on the devices which share an electrode with the selected device (i.e., half selected devices). If the switching threshold of these devices is ~$V_{\text{set}}/2$ or less, their state shifts undesirably, resulting in an imprecise tuning. A similar idea also holds for the reset operation. Fig. 27b shows the measured *IV* characteristics of a device (R_0) in the 64×64 crossbar. Two hypothetical switching threshold distributions, as well as presumptive *IV* characteristics of two corresponding devices, are included to clarify the point. When we set R_0, $V_{set,R0}/2$ drops on both R_1 and R_2. The state of R_1 is expected to alter negligibly since the set threshold of R_1 is much larger than $V_{set,R0}/2$, unlike R_2 that switches considerably. Hence, when tuning the entire crossbar, the total disturbance is correlated to the variations in the distribution of switching thresholds, and the smaller variations (or higher uniformity) result in a higher tuning precision.

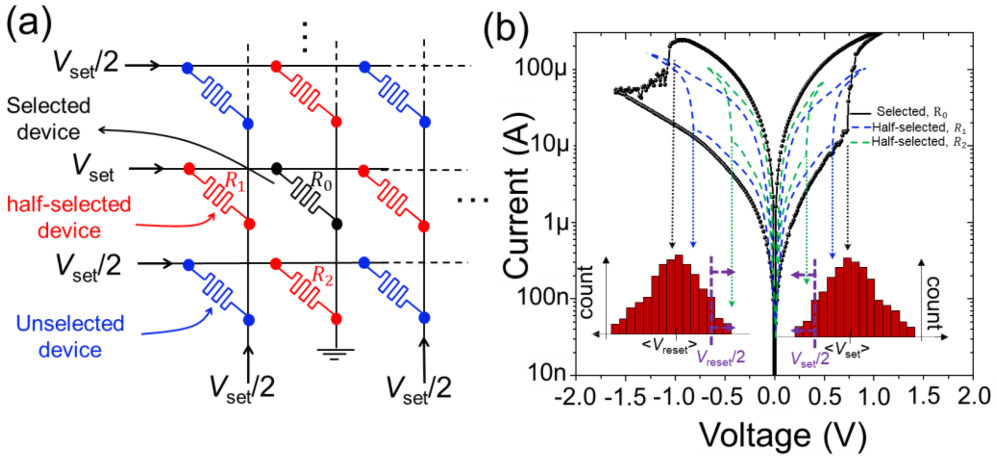


Fig. 27: (a) The schematic of the 3×3 portion of the crossbar and the V/2 tuning scheme with highlighted selected, unselected, and half-selected devices. (b) A typical *IV* characteristic of a device that reveals why the tight distribution of switching voltage is critical.

3.1.2. Analog-grade Passive Crossbars Circuits

Retention, endurance, yield, G_{on}/G_{off}, and variations are critical factors for analog-grade passive crossbars, particularly in the context of neuromorphic computing, in which the weight precision is paramount and utterly important. A 32×32 WO_x memristor crossbar is reported in [42] with $G_{on}/G_{off} = (3\ /1)$ μS and >35% tuning error, though it is not clear if this reported precision is obtained after programming the entire crossbar or otherwise. A 108×54 crossbar made of 126 subarrays of 6×8 ~600 μm^2 WO$_x$ devices are integrated on CMOS in [51] despite low yield and minute-scale room-temperature retention. Ref. [23] demonstrates low normalized variations (~3.75%), excellent retention and endurance on a 16×1 crossbar of 100 μm^2 SiGe devices. Ref. [71] also demonstrates passive crossbars using two-dimensional materials with 98% yield and 12.3% (5.7%) normalized variations in the set (reset) switching distributions.

TiO$_2$ memristors have been used in designing 10×2 [1], 12×12 [40], and 20×20 [47], and 64×64 [32] crossbar circuits, with excellent retention (>20 hrs in 100°C), endurance (> 106 analog switching cycles), and close to 100% yield. The normalized variations in these works are 10%, 11%, 18%, and 26%, respectively. The same stack is also used in the only analog-grade 3D integrated demonstration [41] using two layers of 10×10 memristor array and reporting ~13.6% normalized variations. The consistent trends in the TiO$_2$ crossbars indicate that the larger the crossbar size, the higher the normalized variations.

This trend partially stems from the fact that (the largest) forming current required for electroforming increases with the crossbar size owing to the increase in leakage currents. Hence, a larger compliance voltage/current is required as more and more devices are electroformed, which increases electrical stress and, ultimately, leads to a higher device variability. When forming our 64×64 crossbars, the maximum electroforming current is set to ~50 μA at the beginning, but it is raised to ~ 1 ÷ 5 mA at the end. In addition to this observation, the more devices in the crossbar, the more disturbance created during tuning. As it will be shown in this study, these factors amplify each other and exponentially entangle the design and operation of larger analog crossbar circuits. Fortunately, preliminary architectural studies show that for many computing applications, e.g., deep learning accelerators, the optimum crossbar dimension is in the range of 64×64 as choosing enormous crossbar modules underutilizes the hardware resources and reduces the overall performance [82].

On the other hand, the relationship between the variations in the switching threshold voltages and the crossbar size with circuit fidelity was not studied earlier. To clarify it, we used a developed dynamic model [32] for the memristor that relates the conductance change to the switching thresholds and the applied voltage to emulate the tuning process of ex-situ weight transfer and find the relationship between the accuracy, block size, and normalized variations in general VMM blocks and representative neuromorphic circuits.

3.2. Simulation Framework

In ex-situ training of a neuromorphic network, synaptic weights are calculated on a precursor software-based network and then imported sequentially into the crossbar circuits. Networks are typically composed of many crossbar blocks which are programmed in parallel or sequentially. However, within a crossbar, the devices are tuned into their corresponding predetermined desired states individually (one-by-one). Due to the stochastic nature of the switching mechanism in memristors, particularly analog-grade devices, often require multiple pulses to reach an absolute accuracy. This is executed using the well-known write-verify algorithm [90].

In every simulation case, the conductances of devices are initially randomized using a Gaussian distribution with an average of 36.25 μS (midrange conductance) and a standard deviation of 9 μS. Then, conductances are adjusted one by one using the write-verify algorithm and the dynamic model. We reconstruct the exact procedure that we employ in the experiments when tuning the devices [32,90]. The devices within any crossbar block are tuned in raster order. More importantly, to increase the tuning speed, we progressively increase the pulse amplitude (set/reset) starting from 0.5 V with 10 mV steps to the switching voltage of the device. This idea also prevents overstressing the device. The tuning direction (setting or resetting) is alternated whenever we pass the target conductance. To avoid overstressing the memristors, creating too much disturbance, and reducing the tuning time, we limit the tuning process for every device to 5 rounds. The algorithm is aborted (and restarted with the next device) whenever it reaches the desired tuning accuracy or the maximum permitted pulse per device. In the simulation, the half-select disturbance is applied for every applied pulse and every device by updating the state of devices sharing either top or bottom electrodes with the V/2 rule. The entire crossbar is tuned for 10 rounds to diminish the disturbances.

3.3. Computing Precision in Nonuniform Crossbars

VMM is the most critical operation in inference accelerators and most neuromorphic tasks. The fidelity of most neural network models closely follows the computing precision in their VMMs. Here, we consider $N\times N$ two-quadrant VMM circuits, which are implemented in the analog domain by two separate $N\times N$ memristive crossbars. VMM size, variations in switching thresholds, and target precision are variables of this research. For every case study, 20 crossbars with random log-normally distributed switching thresholds and 20 different normally distributed weight matrices with zero mean are generated. The mapping function $G_{ij}^{\pm} = G_{\min} + (1 \pm W_{ij})(G_{\max} - G_{\min})/2$ in which W_{ij} is the normalized weight and $G_{\max}$ and $G_{\min}$ are upper and lower conductance bounds are used to convert dimensionless weights into device conductances [36]. For each VMM, we randomly generate 1k input voltage vectors, with elements uniformly distributed in the range 0 to 0.1 V. VMM computing errors are then calculated over the output current (I) and defined by $|I_{\mathrm{actual}} - I_{\mathrm{ideal}}|/I_{\max}$. Ideal currents ($I_{\mathrm{ideal}}$) are obtained directly from the mathematical vector-by-matrix multiplication of the input voltage vector and conductance matrix, actual currents (I_{actual}) are obtained from the circuit simulation after all devices are tuned, and $I_{\max}$ is the maximum absolute pre-activation current over all input combinations.

First, the half-disturbance issue is investigated for 64×64 VMMs with considering 5% and 25% variations in switching threshold voltages. Fig. 28a shows the tuning error for 50 devices (in the crossbar that implements G^{+}) during 10 rounds of the programming phase in the case with 5% variations. Specifically, each curve shows how the tuning error for each device evolves, starting from the first attempt to the last one. One curve is highlighted for better clarity. The steep drops in each curve denote the moments the device is tuned. For the highlighted curve, the device is initially tuned with <1% accuracy, but the disturbance moves its state leading to ~5% error by the end of the 1st round. The device is returned in the 2nd round, and the disturbance alters it to ~3% of the target. Less disturbance generated in the 2nd round stems from the fact that some devices are within the target accuracy by the end of the 1st round. So, the total number of pulses (and hence overall disturbance) decreases in each round. The state of most devices stabilizes by the end of the 4th round. The conductance error distributions and related statistics, shown in Fig. 28b, confirm these findings as well.

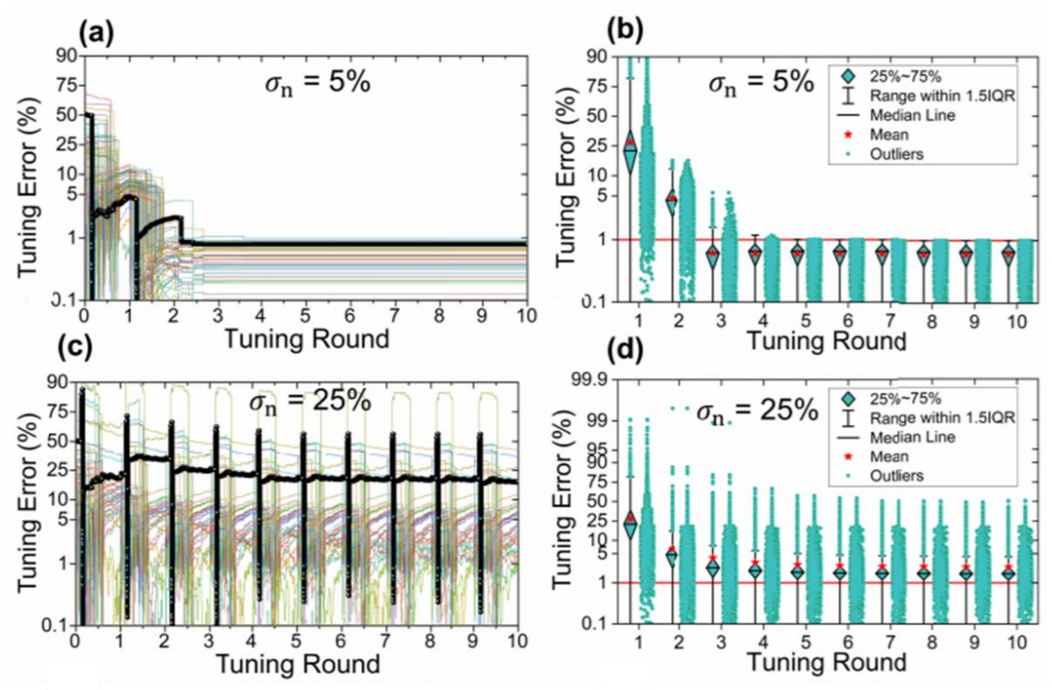


Fig. 28: Evolution of tuning error for tuning the crossbar with (a) 5% and (c) 25% of normalized variations in switching thresholds. The error distribution for all devices at the end of each round for (b) 5% and (d) 25% of normalized variations in switching thresholds.

Note that the assumption of 5% variations in a 64×64 crossbar is too ideal with the current technology. Figs. 28c and 28d show the result from the simulations of crossbars with 25% variations in the switching thresholds. Though the result slightly improves in the first 4 rounds, many devices remain in imprecisely tuned states after that. The periodic state evolution of many devices (e.g., the highlighted curve) in Fig. 28c is because of the large disturbance and strong dependencies, making the tuning effectively unstable for many devices. Fig. 29a compares the ultimate distribution of conductance error for both cases. The 99 percentiles of the tuning error are ~14.4 % and ~1.0 % for 25% and 5% variations, respectively. The huge gap between the realistic and ideal case signifies the importance of variations in passive crossbars. Imprecise tuning results in a large error in the output signal, as expected. Fig. 29b shows the VMM error distribution for both cases. The 99 percentiles of the distributions are ~7.0 % and ~3.7 % for σ_n=25% and σ_n=5% variations, respectively.

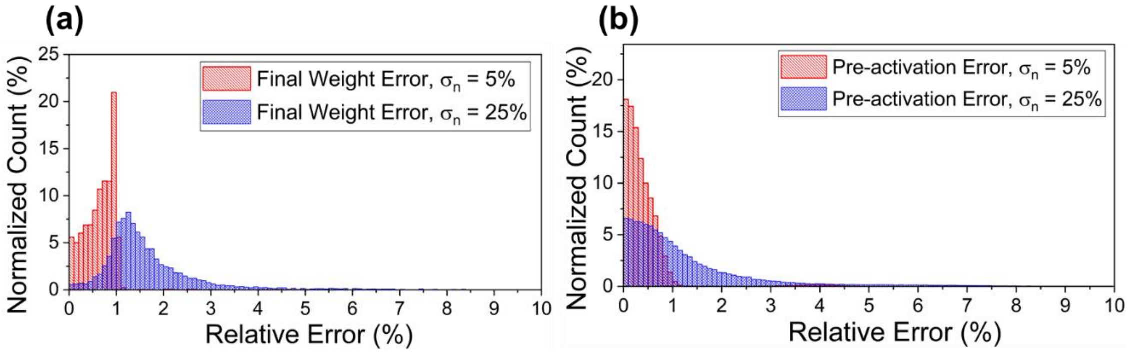


Fig. 29: The distribution of (e) weight and (f) VMM errors (at the end of the 10th round) for the two cases (5% and 25% of normalized variations in switching thresholds).

Fig. 30 summarizes our VMM-level simulation results in which the role of VMM size, switching threshold variations, and target tuning error are studied. In every data point, we consider 400 VMM instances (20 different sets of weights and 20 crossbars) to characterize the worst-case error statistics (99 percentiles of the output error among 10^3 patterns), the de facto parameter to evaluate the computational accuracy. Note that the VMM size and normalized variations are increased exponentially and linearly, respectively.

In every panel of Fig. 30, the dashed red line serves as the intrinsic bound and shows the expected intrinsic error resulting from the imprecise tuning of individual devices (without the half-select problem). Such intrinsic error is often linearly proportional to the target tuning error. The first observation is that the median worst-case error increases exponentially with variations, more evidently for N>30 (here, the median refers to the statistics over 400 VMM instances). It also increases exponentially with respect to VMM size for low variations and super-exponentially in large variations. This stems from the fact that when variations become more extensive in large circuits, the tuning condition becomes unstable (Fig. 28c), and the disturbance of half-select devices overwhelms the tuning of individual devices (Fig. 28c). The spread of the worst-case VMM error distribution among different instances also extends with increasing size or variations in high precision tuning cases since the chance of hitting worse corner cases raises when disturbance escalates. This issue becomes particularly important in high-precision computing tasks with tight error margins.

For small VMMs (e.g., N<16), the error follows the intrinsic trend even in the presence of large variations because the total disturbance is low enough to be fully recovered after running the algorithm for several iterations. In moderate VMM sizes (e.g., N=32), the error tends to increase for high precision tuning cases (e.g., <4%), particularly when the variations are high. This error escalation originates from an increase in the number of applied pulses for achieving a better tuning precision, which in turn leads to a larger disturbance. For large VMMs, variations become more prominent such that the computational accuracy is adversely impacted. For N=64, the drastic change for σ>0.25 also stems from the exponential growth of the overall severe half-select disturbance cases. To clarify this, let us look at n_{HS}, the fraction of the devices in the crossbar circuit, which are disturbed during write operation with half write voltages exceeding their switching threshold. The value of n_{HS} for each case study is provided in each panel of Fig. 30. When the standard deviation of the set threshold distribution increases from 0.25 to 0.3, n_{HS} soars by a factor of ~10, indicating a surge in the cases of severe disturbances.

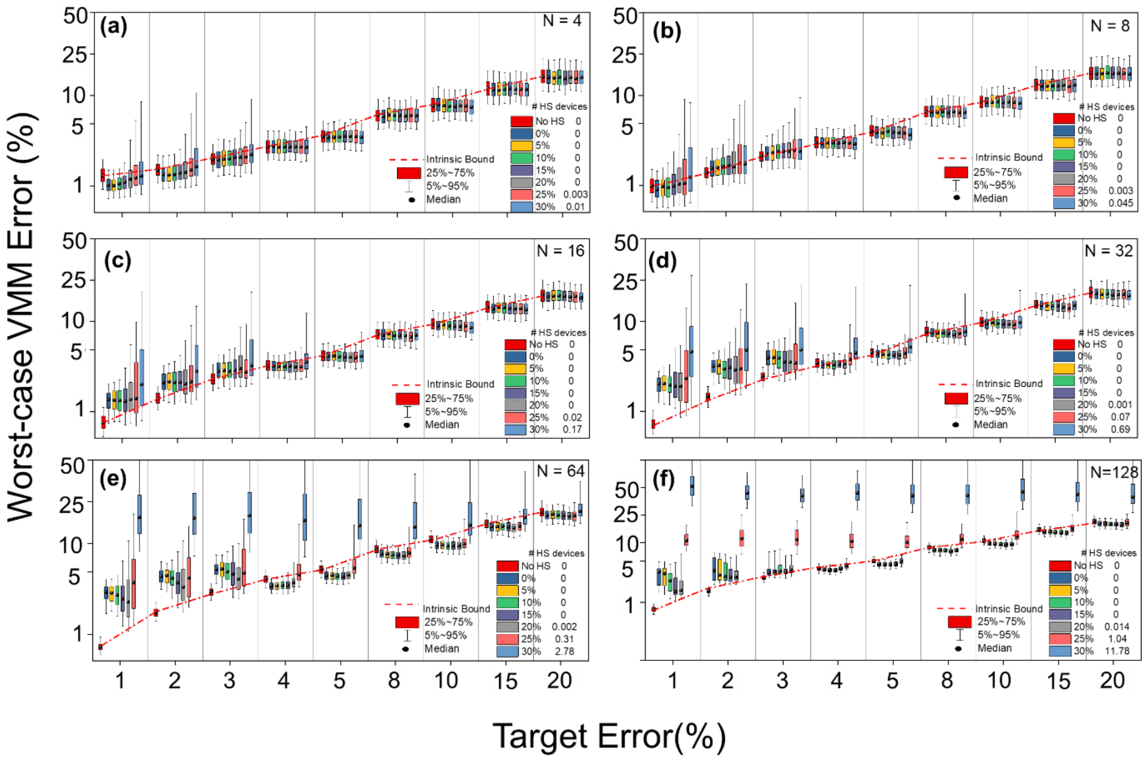


Fig. 30: The distribution of the worst-case VMM error among 400 instances of (a) 4×4, (b) 8×8, (c) 16×16, (d) 32×32, (e) 64×64 and (f) 128×128 VMMs. No HS: no half-select, n_{HS}: the average number of devices affected with V/2 disturbance normalized by N^2. Note the log-scale of the y-axis in all panels.

Another subtle point is related to the reduced computational accuracy in cases with even no variations. For instance, comparing the case of no half-select (no HS) with σ_n=0, we observe a ~4.3% increase of the average error in the case of N=64 and 1% target. Even with no variations, the voltage drops on other devices could have a slightly disturbing effect (non-zero changes in the conductive state at half of the write voltages), which could become potentially noticeable when the total number of pulses grows very large. The slight improvements beyond the intrinsic error (e.g., see the case of N = 4 and target error = 5%) originate from the regularization impact of half-select disturbance, which slightly improves the accuracy.

Finally, the computational accuracy of state-of-the-art 1T1R and 0T1R crossbars can be compared in Fig. 30. For practical VMM sizes (e.g., N=64), considering state-of-the-art reported 1T1R tuning accuracy (~3% in [10]), the computational accuracy of 0T1R VMMs (σ<30) is the same as 1T1R, when tuned with 1% target precision or worse by ~1% when tuned similarly with 3%. The results of this comparison give hope for using 0T1R design in large-scale neuromorphic computing networks.

The final takeaway is that the computational accuracy in passive crossbars is a function of the total number of applied pulses or total disturbance from the tuning precision perspective. Note that the larger the VMM, the larger the number of pulses; the larger the variations, the more pulses needed to tune the devices in multiple rounds; the smaller the tuning target, the higher the number of pulses. Consequently, assuming system-level and architecture considerations determine an optimum kernel size (N) to optimize the functional performance, we end up with two options for mitigating the half-select disturbance and improving the computational accuracy in passively-integrated-based neuromorphic systems namely fabricating more uniform crossbars that lead to tighter variations and developing more optimum tuning algorithms that directly reduce the total disturbance and the number of applied pulses. Further, the most efficient and accurate circuit is not necessarily obtained when the device is pushed to its high precision limit. Hence, extensive simulations are required to find the optimum tuning margin for a given technology, kernel size, and the computing model.

In this study, we consider CIFAR-10 and ImageNet classification tasks and use two relatively compact benchmark networks. Specifically, we use a 4-layer network with a decent size for the CIFAR-10 task that 1) allows the simulations to be conducted in a reasonable time window and 2) more importantly, by using a relatively compact network, we make sure that the half-select disturbance problem shows its true impact on the output (note that the purpose of this work is not to find a network with the highest accuracy on this task, but to show the uniformity problem and how it can be mitigated). The latter point is due to the fact that a highly overparameterized fat network is more sustainable to the half-select problem (but very inefficient in terms of implementing the problem). These two reasons are why we chose this moderate-size DNN. Note that all simulations are also equally performed on the much more extensive ResNet-18 network based on the much more complicated ImageNet task as well.

3.4. Neuromorphic Network Simulation Results

In every model, the VMM operations are partitioned to nonoverlapping $N\times N$ kernels (see partitioning in general-purpose mixed-signal deep neural networks [82]). In other words, to conduct this study using GPUs, we are obligated to employ a one-to-one mapping of weights and two memristive devices (each weight is implemented with two adjustable devices), i.e., there is no weight sharing or duplication in simulations. Let us emphasize that since this study aims to explore the role of device uniformity (rather than any other nonideality), it is indispensable to employ ideal peripheral transfer functions and pooling layers (which are implemented in the digital domain or using winner-take-all circuits). The peripheral circuits and pooling layers are also simulated in software as we intend to study only the impact of uniformity in the accuracy drop. The choice of the peripheral circuits (e.g., time domain or current domain) or data converter is not related to the conducted study. Such considerations are important in architecture works, e.g., Ref. [82].

Similar to the VMM study, the obtained weights are mapped into target device conductances. The conductance tuning process for the constructed VMM kernels is then emulated using the dynamic model and previously discussed tuning algorithm. The imprecise tuned weights are then imported back to the simulation setup. Subsequently, the inference tasks are performed on the generated models, and the classification drop is recorded for each data point. For every case study, 12 model instances are generated (by using 12 sets of randomly generated switching threshold distributions).

Fig. 31 shows the accuracy drop of running the inference test on both benchmarks versus the crossbar uniformity for various VMM sizes. Fig. 31a and 31b correspond to ConvNet and ResNet-18, respectively. The box plot is obtained by simulating 12 random hardware instances (note that tuning simulations are extremely slow even when performed on a powerful server). The destructive impact of crossbar half-select disturbance is evident in both benchmarks, especially in ResNet-18 that performs the more complex ImageNet classification. The trends are consistent with VMM simulations: The accuracy drops exponentially when the VMM size and normalized variations are increased. Notably, with 25% normalized variations and 64×64 crossbars, we achieve ~9% accuracy drop in the ConvNet and 18.5% on ResNet-18. In the next section, we introduce several methods, which restore this accuracy drop and improve the performance.

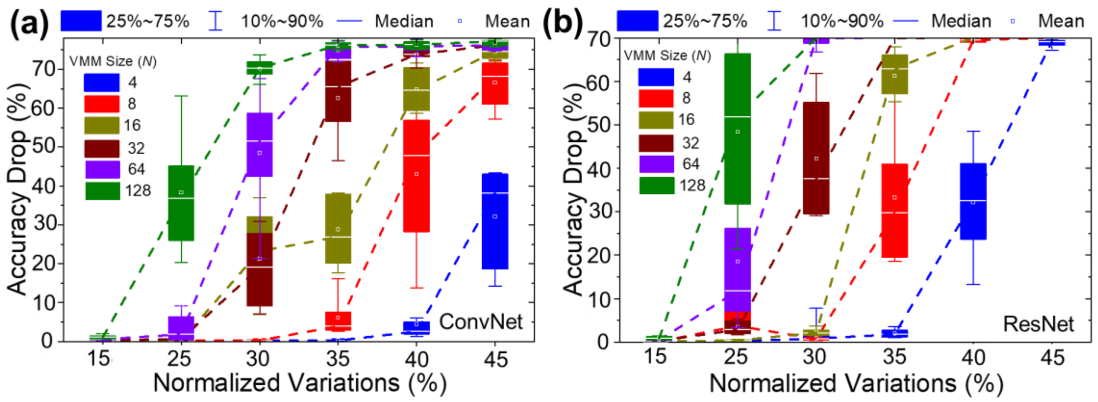


Fig. 31: The accuracy drop in deep neuromorphic networks versus crossbar uniformity for (a) ConvNet, (b) ResNet-18. The dashed lines connect the median of the boxes.

3.5. Improving the Accuracy

The most straightforward solution to cope with the destructive impact of variations in the switching thresholds is to improve the fabrication process and device properties. The switching threshold variations in metal-oxide memristors depend on multiple factors. Forming voltage and current overshoot during the forming significantly contribute to device variability and can be tuned by a combination of oxide layer thickness and stoichiometry adjustment and optimized annealing conditions [40]. In this section, we focus on some post-fabrication techniques for mitigating the disturbance.

3.5.1. Hardware-Aware Training

In Chapter 2, we discussed imperfections of synaptic devices such as noise, temperature dependency, stuck-at fault, and defects that are compensated by the method of hardware-aware training: The training is performed fully ex-situ (no extra hardware cost), with the only subtle difference of including the device models and imperfections in the training phase for the purpose of generating more robust models.

The simulation results of the previous section indicate that variations in switching thresholds lead to random tuning errors in the devices. Note that the tuning errors remain fixed during the inference, assuming devices have adequate retention. Nevertheless, tuning errors are chip-dependent, device-dependent, and unpredictable because of the intrinsic chip-specific distribution of switching thresholds. Despite that it is not feasible to predict and include the exact number of errors in the training phase, the error distribution is predictable due to the uniform shape of weight distribution in a neural network model, especially when using the same crossbar sizes and tuning algorithm (see modular accelerator architectures, e.g., aCortex [82]).

Modeling the tuning error during the training may increase the robustness of the trained model against half-select disturbance during the inference of the neural hardware. Note that this technique does not transform the shape of the tuning error distribution. Before computing the activation values in each update, the weights are converted to memristor conductances. Built-in uniform random number generator with the parameter ζ is then used to perturb conductances (both G^+ and G^- in the differential implementation). After computing imprecise preactivations, the ideal weights are then restored before proceeding with the rest of the training operations. Note that ζ is optimized for a given network model and overall disturbance, which is a function of VMM size, switching threshold variations, and target accuracy. Also, using a more complicated distribution for perturbation might be beneficial.

Fig. 32a shows the performance improvement achieved by this technique on the ConvNet benchmark implemented with 64×64 VMM blocks. The figure shows the accuracy drop versus the normalized variations for various values of ζ. The robustness of the deployed model is obviously increased with this method. For 15%, 25%, and 30% normalized variations, the optimum performance is achieved when ζ is set to 5%, 20%, and 30%, respectively. Notably, in the case of 25% normalized variations and 64×64 crossbars, the ~9% average accuracy drop is now reduced to ~1.87% using ζ=20%. The same trends of improvements are also observed in the case of ResNet-18 implemented with 64×64 crossbars (Fig. 32b). For example, using ζ=3% diminishes the average accuracy drop from 18.5% to 3.5% (6.1%) for σ=25%.

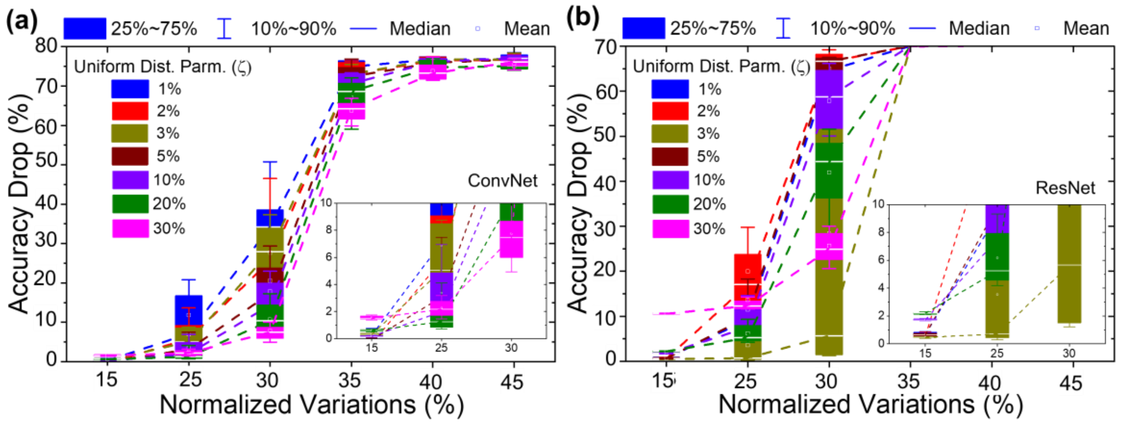


Fig. 32: Reducing the accuracy drop in (a) ConvNet and (b) ResNet-18 (both with 64×64 VMMs) using the hardware-aware training technique. The inset shows the zoomed-in to the lower portion of the figure.

3.5.2. Improved Tuning Algorithm

In Ref. [32], we propose a novel crossbar tuning procedure consisting of two methodologies for reducing the tail of tuning error distribution. First, the maximum write voltage amplitudes are limited to a specific voltage, which is decreased gradually within each tuning round. The consequences of restricting the maximum applied write voltage within each round are gradual reduction of net disturbance in each round and large (final) tuning error in high threshold devices. The former stems from the fact that low-to-moderate threshold devices become disturbed less and less as the tuning algorithm advances. The latter merely originates from the fact that the write voltage is not enough for high threshold devices to switch.

In the second method, we initially identify devices with a large set (reset) switching thresholds and switch them to the highest (lowest) conductive state prior to executing the first tuning round. Then, we take advantage of the possibility to encode the same weight with different target conductances in the differential pair implementation. In every round, when tuning a disturbed device with a threshold higher than the maximum voltage limit imposed by the first methodology, the state of the paired device is adjusted rather than the high voltage device. The application of these two novel techniques significantly reduces the tail of disturbed devices.

Fig. 33a demonstrates the effectiveness of using these novel tuning algorithms with and without applying the hardware-aware training technique. When no hardware-aware training is applied, the novel tuning algorithm reduces the accuracy drop, especially when variations are higher than 20%. When the two techniques are both applied, the results are even better. A sub-percent accuracy drop is now feasible even with 30% normalized variations. For the notable case of 25%, the average drop now becomes insignificant when *ζ*=2% is used in the hardware-aware training.

The simulation results of the ResNet-18 benchmark are also promising (Fig. 33b). For example, in the case of *N*=64 and *σ*=25%, the improved tuning algorithm solely reduces the accuracy drop to 1.88%. Combined with the hardware-aware training (*ζ*=3%), we can decrease the average accuracy drop to just ~0.4%. In the initial simulation, we observe that the model generates almost random outputs (~70% accuracy drop) when the variations are *σ*=35% and larger. While the two proposed techniques enable 6.9% and 17.2% average accuracy drops, utilizing *ζ*=20% and *ζ*=3%, respectively.

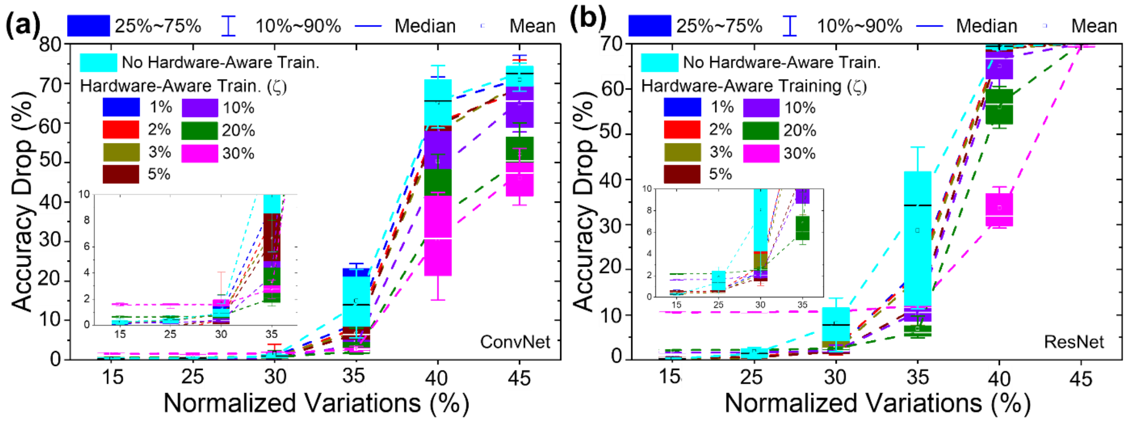


Fig. 33: Reducing the accuracy drop in (a) ConvNet and (b) ResNet-18 (64×64 VMMs) using the novel tuning algorithm with and without the hardware-aware training. The inset shows the zoomed-in to the lower portion of the figure.

3.5.3. Modifying Switching Thresholds

Modifying the switching thresholds of outlier devices is another method for reducing the impact of variations in the switching thresholds. This correction process includes an unconventional continuous hard reset operation, which pushes the outlier device close to its virgin state, followed by a voltage-controlled reforming procedure, which revives the device with slightly shifted switching characteristics. Our experiments show that the correction process results in a stochastic shift in the switching threshold of devices, which means the refreshed device could have improved switching properties. Applying this technique to outlier devices (that feature low voltage thresholds) reduces the spread of variations, which in turn improves the accuracy of the implemented model.

Fig. 34 shows the results of the experiments developed to validate this idea. First, a virgin device in the crossbar is formed and tuned to 50 kΩ. Then, its switching thresholds are measured by tuning it repeatedly to 100 kΩ and 10 kΩ. After 10 rounds, the device is hard reset to >1MΩ and then revived and tuned to 50 kΩ. Switching thresholds are measured again in a similar fashion. The process is performed one more time just to make the results more illustrative. Fig. 34a shows the experimental results, and the inset of Fig. 34a shows how the switching thresholds are changed after the initial forming (i.e., the original thresholds) and each reviving round. The stochastic shift of the switching thresholds is quite obvious from these data.

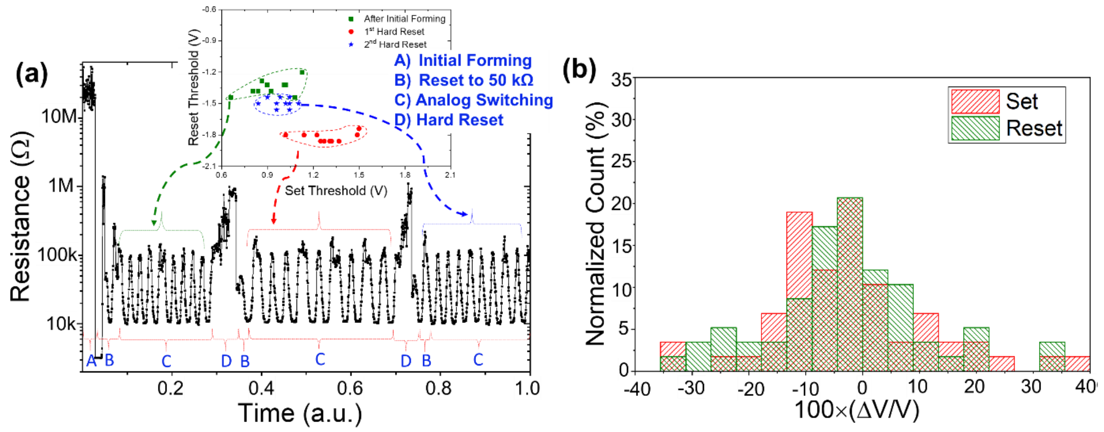


Fig. 34: (a) The experimental results of the modification process applied on a virgin device in the crossbar. (b) The histogram of the stochastic relative change (100×ΔV/V) in the average set/reset switching thresholds (V) for 60 devices after performing the hard reset and revival processes on them.

In addition, a one-time switching threshold modification is applied on 60 devices with various initial switching thresholds, and Fig. 34b corroborates the stochastic nature of this shift in the symmetric histogram of relative change in the switching threshold after the modification process is applied. Specifically, as an example of modifying an outlier device, we apply the modification process to a device with an average set and reset switching thresholds of +0.8 V and -0.8 V, respectively (among ten switching rounds). Then, it is observed that the refreshed device has an average set voltage of 0.95 V and an average reset voltage of -1.2 V, which are significantly better and closer to the typical average switching thresholds of the crossbar. Due to the limitations of our experimental setup, we can not validate the impact of this method with direct system-level experimental results. The study of the impact of this technique on large-scale neuromorphic architectures is important for future work.

3.6. Discussion

Previous works [93, 94] have focused on other nonidealities, e.g., IR drop, static nonlinearities, retention loss, focusing on devices with selectors with inferior density with respect to 0T1R crossbars. Such nonidealities are also essential and have been adequately addressed in [93, 94]. However, this study is the first work exploring the indispensable role of uniformity and presents several novel methods and guidelines to improve the circuit performance regarding that.

The findings in this study confirm the encouraging prospects for using 0T1R crossbars in neuromorphic computing. In the general VMM study (section 3.4), we uncover exciting opportunities and interesting tradeoffs about these circuits for the first time: 1) the relationship between the computational error and the crossbar size, uniformity, and target tuning error is thoroughly investigated, 2) we present the periodic and instability of tuning error in large nonuniform crossbars in addition to the linear and exponential dependency of computing accuracy to uniformity at small, moderate, and large VMM sizes, 3) it is shown that in large VMMs, very precise tuning of devices requires many pulses, which in turn may lead to more disturbance and reduction of the ultimate computing accuracy, 4) slight increase in the computational error is inevitable in very large 0T1R crossbars even with zero variations since even a small half-select voltage drop could become potentially noticeable when the total number of pulses grows very large, 5) we compare the computational accuracy of state-of-the-art 1T1R (~3% target error reported in [92]) and 0T1R crossbars (1% target tuning and σ~25% reported in [32]) and report similar computing accuracy when the 0T1R crossbars are tuned with 1% target precision or worse by only ~1% when using the same (as 1T1R) tuning precision of 3%.

Three techniques are explored for mitigating the impact of nonuniform *IV* characteristics of 0T1R memristors in neuromorphic circuits. The simulation results indicate that these techniques enable software-equivalent accuracy on both ResNet-18 and ConvNet benchmarks, in the case of *N*=64 and 25% normalized variations, which corresponds to the features of the recently fabricated crossbar [32]. In addition, the presented data in Fig. 33 suggest that a sub-percent accuracy drop is achievable in advanced neuromorphic circuits with even ~30% normalized variations using 64×64 crossbars, which leads to a balanced resource utilization at system levels, as promised by theoretical architectural studies [82].

Also, several limitations of these mitigation techniques can be pointed. First, hardware-aware training is not a viable option in some neuromorphic tasks, e.g., neurooptimization [95], in which the weights are fixed and predetermined by some constraints of the applications. In such cases, the practical solutions are improved tuning algorithm, fabrication process, outlier correction, and, if needed, reducing the crossbar dimensions. Second, the switching threshold modification method should only be used for outlier devices once or a few times to prevent damaging the devices or reducing their endurance life.

4. CMOS Integrated Memristive Circuits

In order to implement bio-inspired neural network models, the analog and mixed-signal (AMS) approach is very promising. The energy efficiency of the AMS approach highly depends on improvement in the physical level implementation of the VMM blocks. RRAM technology, as it is mentioned in the previous chapters, is one of the favorable candidates to achieve these goals due to its long-term retention, high endurance, low power consumption, scalability, and analog-storage ability [96].

There are also some infrequent operations in neuromorphic networks that synaptic devices (RRAM) cannot implement, or they could be inefficient to do so. Fortunately, CMOS technology is exceptionally flexible and can efficiently implement such functionalities. Therefore, integrating CMOS and RRAM technologies are extremely important, and designing efficient peripheral circuits in CMOS technology that perform the less frequent and sparser functionalities of these systems is very crucial. This chapter presents our large wafer-scale designs to alleviate the challenges, such as implementing a system integration of CMOS and RRAM technologies and designing efficient peripheral circuits in neuromorphic circuits.

Fig. 35 shows two scenarios for integrating the emerging RRAM technology with a standard CMOS process: the Back-end of the line (BEOL) process in which memristors are integrated on the top of the CMOS stack, e.g., on top of M5 in a process with 5 metal layers, and the Middle-end of the line (MEOL) process in which the RRAM crossbar is inserted somewhere in the middle of CMOS metallization layers, e.g., between M4 to M5 in a process with 9 metal layers. In this chapter, we first discuss the details of two tapeouts with the purpose of conducting BEOL integration of UCSB's passive memristors on 180 nm CMOS. Then, we discuss a fabricated DNN accelerator chip in 65 nm technology using 1T1R industrial memories integrated with CMOS in a MEOL process.

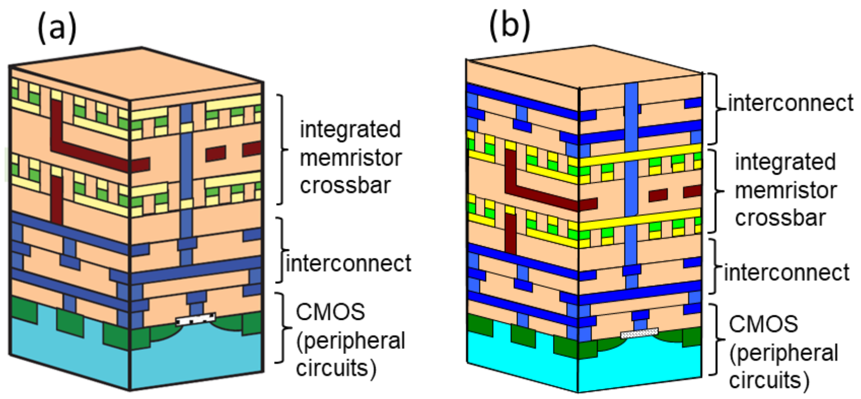

(a)
integrated memristor crossbar
interconnect
CMOS (peripheral circuits)
(b)
interconnect
integrated memristor crossbar
interconnect
CMOS (peripheral circuits)

Fig. 35: CMOS/RRAM integration using (a) BEOL (b) MEOL processes.

4.1. Wafer-Scale 0T1R Integration

One favorable approach to implement VMM blocks is based on passively resistive RAM technology, which has demonstrated excellent scalability and area efficiency. This section discusses the design and fabrication of two large tapeouts, including integrated hybrid monolithic CMOS/memristor circuits for AMS computing applications in Silterra's 0.18 µm 6-metal CMOS technology and 250 × 250 nm^2 Pt/Al_2O_3/TiO_{2-x}/Ti/Pt passive memristive devices. CMOS and memristor circuitries are fabricated in Silterra fab and UCSB's nanofab, respectively. The fabricated wafers include several circuits and full-scale proof of concept demos, such as different complexity and styles of mixed-signal VMM circuits, medium-scale neuromorphic network, e.g., multi-layer perceptron (MLP), physical unclonable function (PUF), neuro-optimizer, and random number generator.

4.1.1 Building Block (X-block) Design

Although the main purpose of a memristive crossbar is to implement specific functionality, e.g., VMM operation, it needs some other peripheral circuitries which allow us to form, program, and operate the circuit in different modes. A building block, which we call X-block, is used in all of our memristive computing demos and consists of the memristive crossbar, tuning and inference switches, decoders, level shifters, and programming shift registers. X-block supports five modes of operations: forming, programing (i.e., set and reset), read, monitoring, and inference.

To support forming, programming, read, and monitoring (each mode is discussed in the next section), the address of a specific device (selected device) will be stored in the designated shift register. The shift register drives a custom-designed decoder that controls the modes using two auxiliary inputs, converts a binary m-bit input to one-hot 2^m outputs, and drives the level shifters connected to tuning switches. Note that since the tuning/forming voltages often require voltage more than the core voltage (here 1.8 V in 180 nm), we need to use thick oxide MOSFETs as switches and hence use level shifters (1.8 $\rightarrow$ 3.3) to drive the tuning switches. The inference switches protect the core-voltage input generator and readout circuits during the (relatively) high voltage forming and programming modes. These sets of circuits (shift register, decoder, tuning switches, and inference switches) are used for both bottom electrodes and top electrodes. Fig. 36 shows the overall topology of the X-block circuit. In the following, we will discuss the five operation modes.

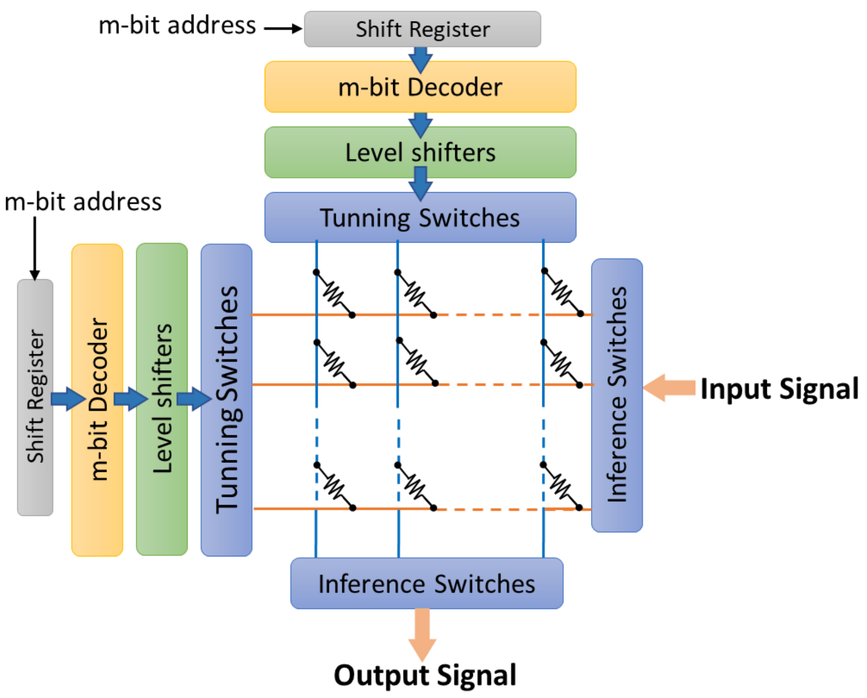


Fig. 36: X-block used in all our analog computing demos.

1) Forming: Upon fabrication, memristor devices are initially in the pristine state and require a one-time forming process before becoming adjustable memristors. A device can be formed by applying a relatively high voltage (e.g., 2.5 V - 5 V) to it and continuously monitoring its conductance. When the device reaches a certain threshold, a conductive filament forms inside it, and its conductance jumps significantly, enabling subsequent analog-state tuning and storage (Fig. 37).

2) Programming: In this phase, the conductance of the device is adjusted to a desirable value (G) through the modulation of the impurity profile. We may increase (set) or decrease (reset) the conductivity of the device by applying a moderately large voltage to the device. The programming voltage is about (or slightly larger than) the switching threshold—a device-unique voltage that alters its conductance by, e.g., 20%. Harnessing the write-verify algorithm [90], we keep programming and checking the state of the device (G') until reaching a certain relative tuning accuracy $(\hat{G} - G)/G < \in$. In chapter 3, we discussed how the *V*/2 scheme is used to set/reset a device with the amplitude of *V*.

3) Read: During the programming (set/reset) of a device, we need to read the state of the device (after each set or reset pulse) to verify if it has reached the desired value or not. The read mode allows us to read the low voltage conductance of a single (selected) device that is being programmed. In this mode, a low voltage V_{read} is applied on top of the device, and its generated current I_{read} is sensed. The conductance of the device is given by I_{read}=$G\,V_{\text{read}}$.

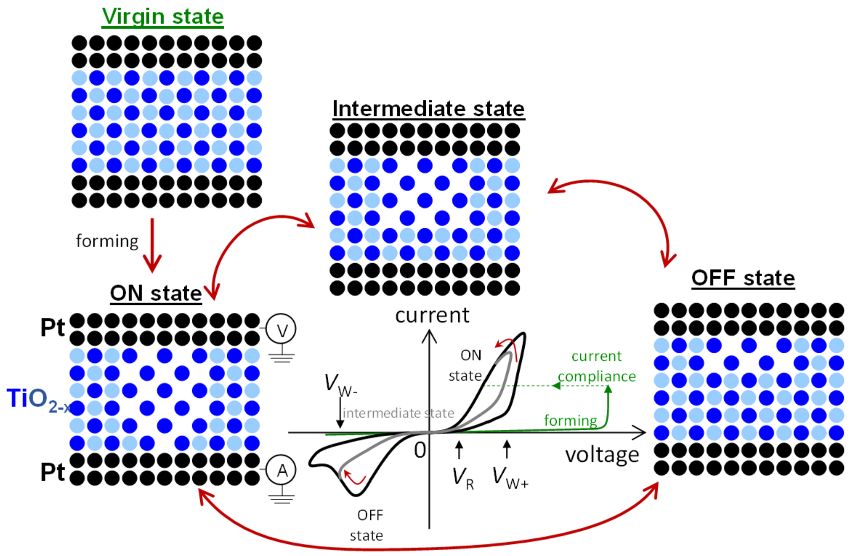


Fig. 37: Forming, programming, and read phases of a memristor device.

4) Inference: Unlike the read mode in which the current from a single selected device is sensed, in the inference mode, typically, most devices are used with the goal of implementing or performing the main intended task. Hence, the inference mode is task-specific, e.g., in a neural network design, the inference mode is when the crossbar is used for the VMM operation. Note that in order to implement multiplication, summation, or useful computational tasks, all devices are operated in the non-disturbing (inference) phase: e.g., a relatively low voltage (V) is applied to the devices and the generated current, $I_j = \sum G_{ij} V_i$ is sensed with a sensing CMOS circuity (to implement the dot-product operation) from every electrode.

5) Monitoring: This mode is necessary for checking the characteristics of the peripheral core voltage circuits, for example, measuring the offset or nonlinearity of the input generator or readout circuits. Such a feature becomes even more critical when we are interested in performing offset compensation using memristive devices.

In the following, we will discuss how the X-block components are designed in order to implement these five modes of operation.

4.1.1.1 Tuning and inference switches:

The switch circuitry, including tuning and inference switches, connected to every electrode is shown in Fig. 38. The tuning switches are mainly used to select and unselect the devices and are controlled by the custom decoder. They are used in forming, programming, read, and monitoring modes. In inference mode, they are turned off. In all modes except the inference mode, in which all select and unselect electrodes are turned off, only one top and one bottom electrode are selected. When a certain electrode is selected, its corresponding select switch is turned on, and its unselect switch pair is turned off. For all other electrodes, the select switches are turned off, and unselect switches are turned on. The inference switch protects the core voltage peripheries from the IO level voltages applied to the electrodes during forming and programming of the devices. Hence, it is only turned on during inference and monitoring modes. SD_p, SD_n, UD_P, UD_n signals are IO level signals driven by level shifters which are connected to a custom decoder. The required voltage/current to form/set/reset/read/monitor the devices are provided by V_{SD} and V_{UD} signals, which are typically generated in off-chip circuits. V_R is also routed to the core voltage peripheral circuits.

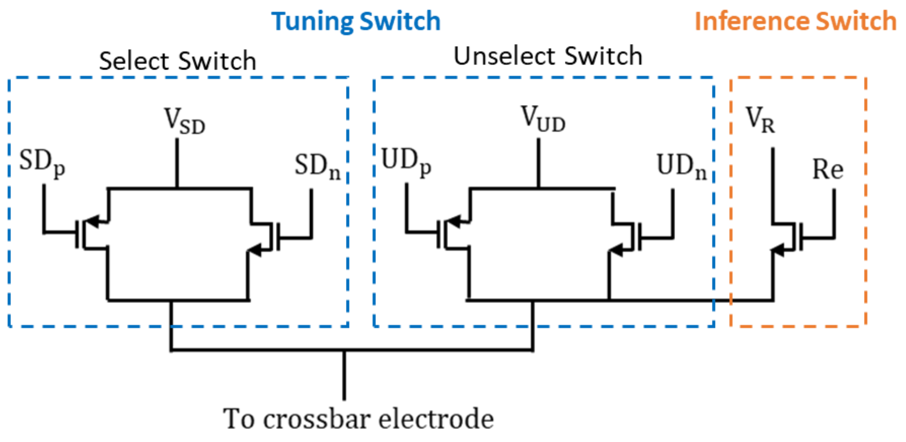


Fig. 38:Tuning and inference switches.

4.1.1.2 The shift register, custom decoder, and level shifter:

In forming, programming, tuning, and monitoring modes, we are targetting a single electrode in the crossbar. Prior to executing any of these operation modes, the binary address of the selected electrode is serially shifted into a shift register, which drives a custom decoder. The decoder shown in Fig. 39a generates auxiliary signals used to control the tunning switches in different operation modes. Table. III shows the logic of the decoder signals in each operation mode. Shift registers and decoders are designed in the standard digital flow (synthesis and automatic GDS generation) with the goal of minimizing the area. Note that since these circuits are not used in the inference mode, their energy consumption and speed are not crucial. Fig. 39b also shows the circuit schematic of the level shifter, which translates the digital Q and $\overline{\mathrm{Q}}$ at 1.8 V to OUT and $\overline{\mathrm{OUT}}$ at 3.3 V.

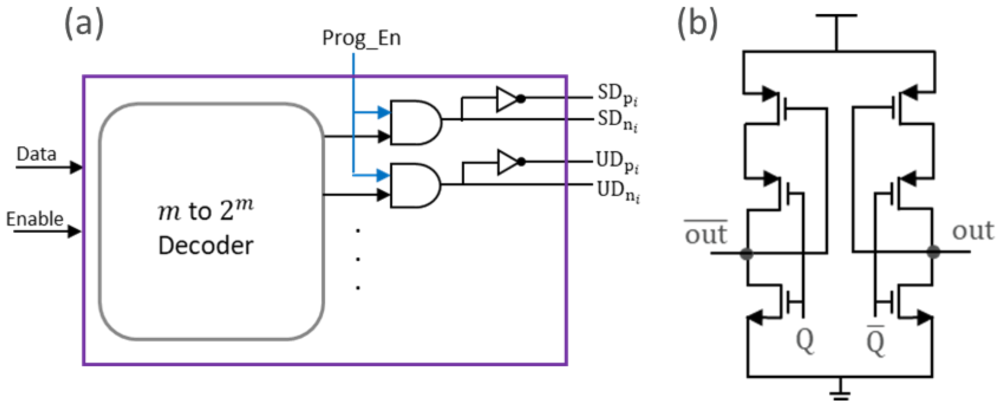


Fig. 39: (a) The custom decoder (with 1.8 V supply) and (b) the level shifter circuit using thick-oxide transistors with 3.3 V supply.

Table. III: Logic of the custom decoder.

Mode	Enable	Prog_En	Re	Selected Electrode				Unselected Electrode				V_R
				$SD_n = \overline{SD_p}$	V_{SD}	$UD_n = \overline{UD_p}$	V_{UD}	$SD_n = \overline{SD_p}$	V_{SD}	$UD_n = \overline{UD_p}$	V_{UD}	
Forming	1	1	0	1	$V_{SD\_WL} = \mathbf{V_{pulse}}$ $V_{SD\_BL} = \mathbf{0}$	0	X	0	X	1	$V_{UD\_WL} = \mathbf{V_{pulse}}/2$ $V_{UD\_BL} = \mathbf{V_{pulse}}/2$	X
Set	1	1	0	1	$V_{SD\_WL} = \mathbf{V_{pulse}}$ $V_{SD\_BL} = \mathbf{0}$	0	X	0	X	1	$V_{UD\_WL} = \mathbf{V_{pulse}}/2$ $V_{UD\_BL} = \mathbf{V_{pulse}}/2$	X
Reset					$V_{SD\_WL} = \mathbf{0}$ $V_{SD\_BL} = \mathbf{V_{pulse}}$							
Read	1	1	0	1	$V_{SD\_WL} = \mathbf{V_{pulse}}$ V_{SD_BL} to current measurement	0	0	0	X	1	0	X
Inference	X	0	1	0	X	0	X	0	X	0	X	V_{R_WL} from input generator V_{R_BL} to sensing circuit
Monitoring Input voltage generator	1	1	1	1	V_{SD_WL} to voltage measurement	0	X	0	X	0	X	V_{R_WL} from input generator
Monitoring Sensing circuit					V_{SD_BL} to apply current	0	X	0	X	0	X	V_{R_BL} to sensing circuit

Let us now discuss how each mode of operation is implemented and executed in the circuit.

a) Forming and Set operations

Fig. 40 indicates the process of forming/set of devices using tunning circuits. Let us clarify that V_{SD} and V_{UD} signals that are connected to bit-lines (or top electrodes) are denoted by V_{SD_BL} and V_{UD_BL}, respectively. These signals are directly routed to analog multiplexers placed on the PCB adapter (i.e., off-chip). Depending on the operation mode, V_{SD_BL} and V_{UD_BL} are either driven by a fixed voltage, a pulse generator, current sensing, voltage sensing, etc. A similar idea also holds for word lines (bottom electrodes) signals as well.

For forming and set operations, a word-line (WL) and a bit-line (BL) are selected by the decoding circuit. The off-chip switch matrix circuit is configured such that a voltage pulse V_{pulse} is applied to $V_{\text{SD\_WL}}$, $V_{\text{SD\_BL}}$ is grounded, and $V_{\text{UD\_BL}}$ and $V_{\text{UD\_WL}}$ are driven by $V_{\text{pulse}}/2$. Such topology essentially implements the $V/2$ scheme to prevent half-select problems [96] discussed in chapter 3. The applied voltage to form a device is relatively high (2.5 – 5 V), while the set voltage is typically smaller than 2 V. In the case of forming in large crossbars, we may leave the unselected electrodes from both top and bottom electrodes floating to minimize the current required for forming.

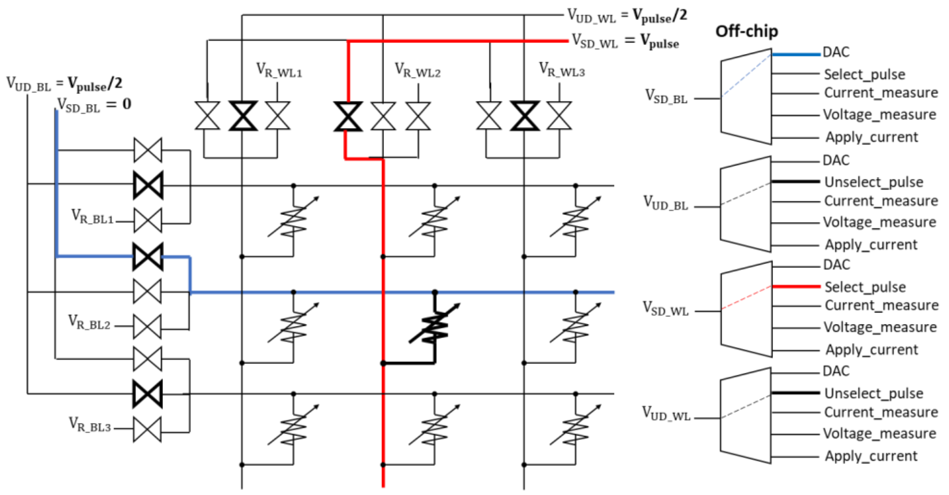


Fig. 40: Set and Forming Modes. The selected device and “on” pass gates are highlighted for clarity.

b) Reset operation

The process of resetting a device is very similar to set, only with the oppositive polarity. Here, a voltage pulse V_{pulse} usually smaller than 2 V is applied to $V_{\text{SD\_BL}}$, and $V_{\text{SD\_WL}}$ is grounded. Fig. 41 indicates the reset scheme in the X-block. Note that $V_{\text{pulse}}/2$ is also applied to $V_{\text{UD\_BL}}$ and $V_{\text{UD\_WL}}$ to prevent half-select problems.

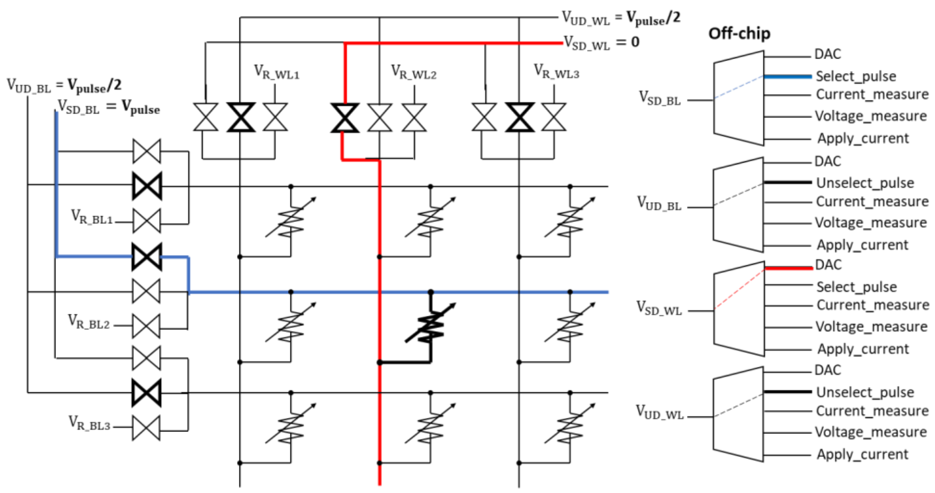


Fig. 41: Reset mode in X-block.

c) Read operation

The process of reading the state of a device (during the programming mode) is performed by providing a fixed small voltage on the selected WL port (V_{SD_WL}) and measuring the output current generated in the selected BL port. An off-chip current sensing circuit imposes virtual ground on the selected BL port (V_{SD_BL}) and senses its current. V_{UD_BL} and V_{UD_WL} are also grounded to zero out the current from other devices. Fig. 42 shows the reading process.

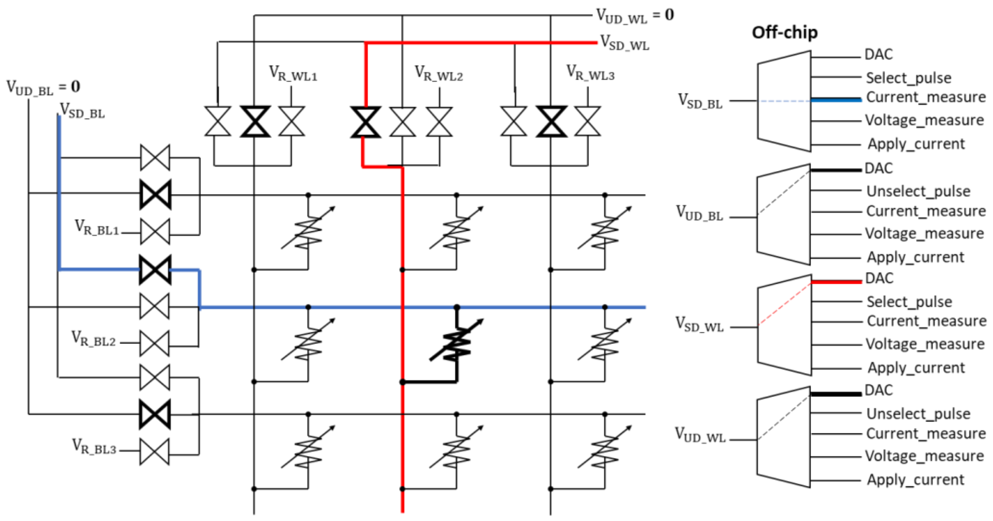


Fig. 42: Read mode in X-block.

d) Inference operation

Fig. 43 indicates the inference process (the normal operation of the crossbar). Typically, all devices in the crossbar are involved in this mode of operation. So, the inference switches of all BLs and WLs are turned on (via R_E signals). A voltage from an on-chip input voltage generator is applied to all devices from the WLs, and the output currents (the summation of computation results) are sensed in parallel from all BLs using sensing circuits.

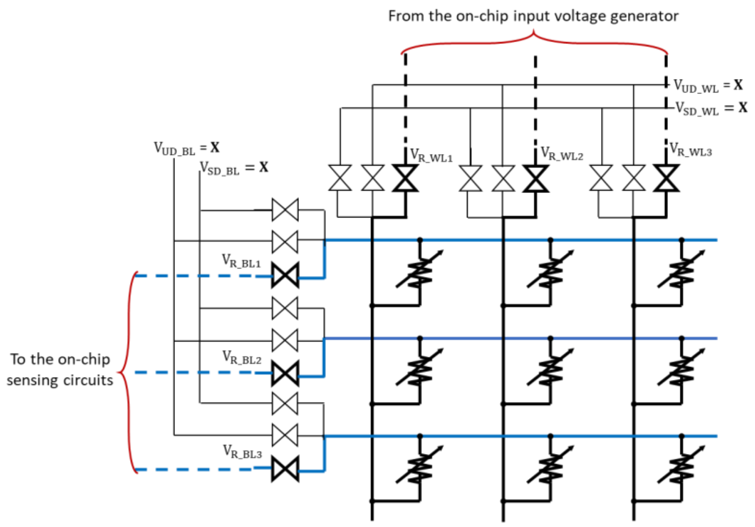


Fig. 43: Inference mode in X-block.

e) Monitoring operation

The circuit operation for studying the functionality of input generators and sensing circuits are shown in Fig. 44a and Fig. 44b, respectively. For example, to check the input generator's characteristics, the inference switches for WLs are turned on. The selected input generator drives the selected WL port. Using the V_{SD_WL} and tuning switches, the output voltage of the input generator can be measured using an off-chip voltage measurement (Fig. 44b). In this mode, the devices are not involved. The sensing circuit (Fig. 44a) characteristics are measured similarly.

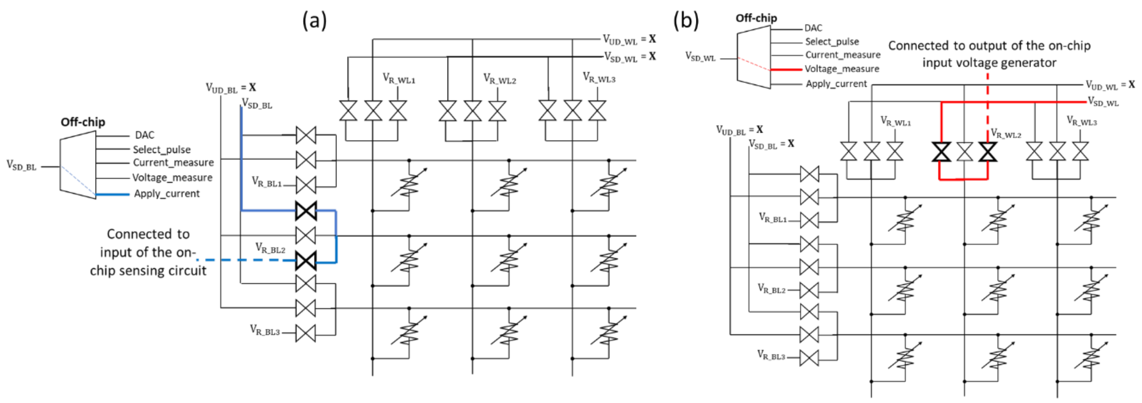


Fig. 44: The monitoring mode for measuring (a) the sensing circuits' characteristics and (b) input voltage generator's characteristics.

4.1.2 Fabricated Circuit Demos

We ran two wafer-scale tapeouts to integrate 180 nm CMOS circuits and 250 nm RRAM technologies. Fig. 45 shows the cartoon of how a crossbar is connected to underlying CMOS circuits. The top and bottom electrodes are specified in orange and green colors, respectively. RRAM devices are built at the intersection of the top and bottom electrodes (red circles). The via PADs are used for connecting memristor crossbars to CMOS circuitries.

Two scenarios are considered for BEOL integration. In the first scenario, both electrodes (top and bottom) are implemented in the BEOL process (using additional metal layers grown in UCSB's fab as a part of the integration). In contrast, in the second scenario, the M5 layer (from the CMOS stack) is used as the bottom electrode, and the top electrode is implemented at the BEOL process. In both scenarios, CMOS M4 and M2 layers are employed to bootstrap the crossbar electrodes to reduce the effective resistance (of top and bottom lines). The bootstrapping technique is suggested in [34] as a solution to improve the accuracy of the memristive VMM concerning IR drop across crossbar lines.

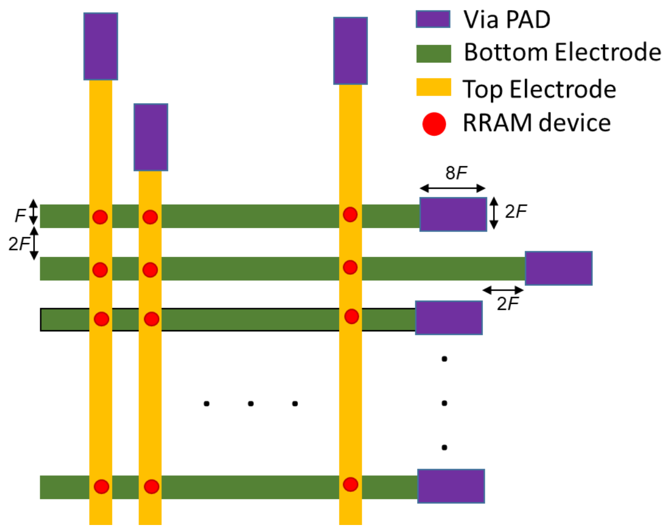


Fig. 45: The layout of a portion of the crossbar. *F* is the minimum feature size (250 nm in our design).

The designs are fabricated on 15 ×12 mm^2 reticles. The layout and snapshot of the wafer for the first demo are shown in Fig. 46a and 46c, respectively. This design consists of different flavors (sizes and types) of VMM circuits, an MLP network, a PUF circuit [97, 98], and RRAM macros (X-blocks). The second tapeout contains Sirius PUF [99, 100], a random number generator, and an optimization circuit (Fig. 46b and 46d) [101-103]. In the following section, we will provide more details about each demo.

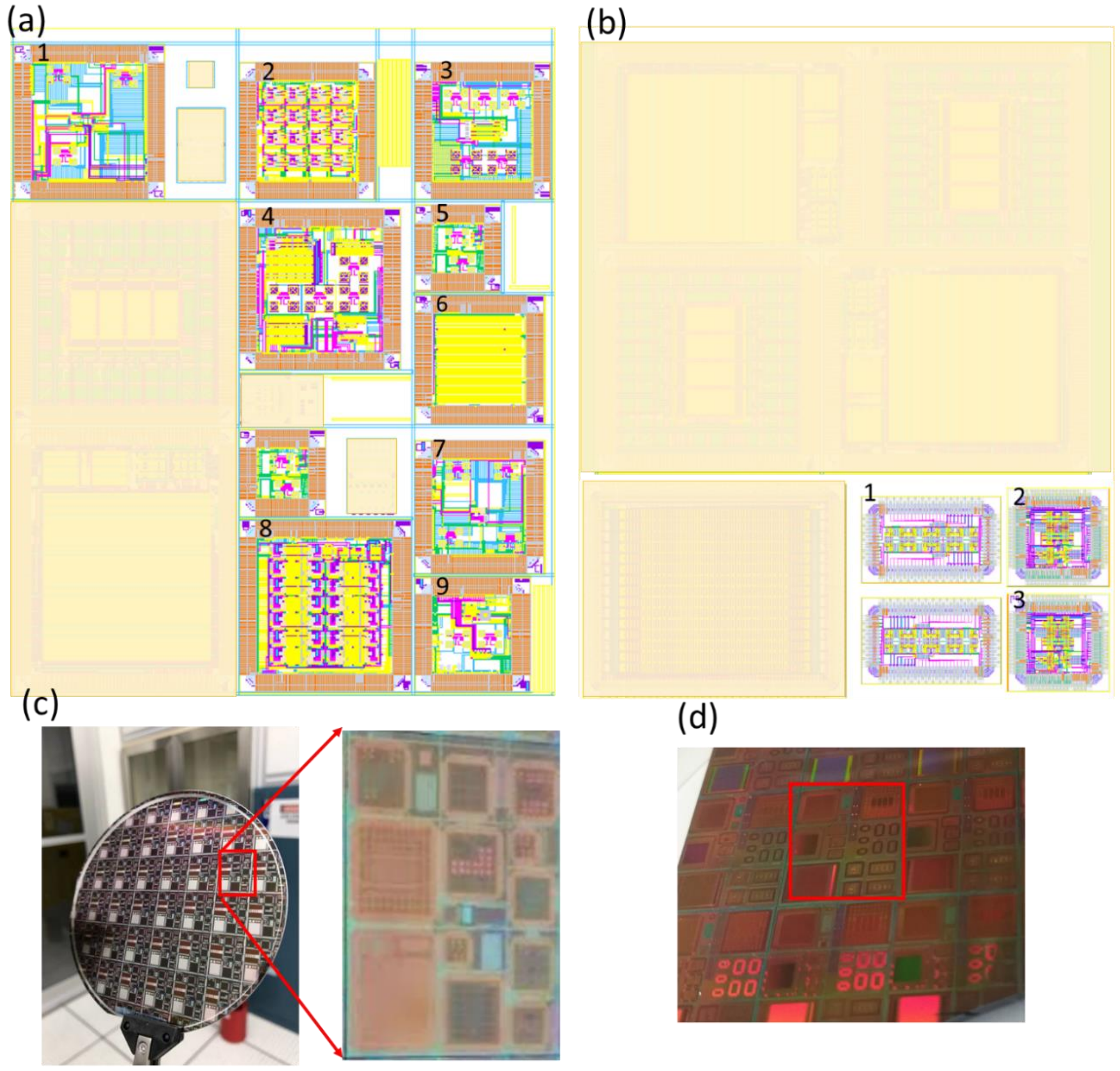


Fig. 46: (a) The first and (b) second runs of the passive RRAM and CMOS integration tapeouts. The blurred regions are designs from collaborators. Wafer pictures of (c) the first and (d) the second demo.

a) Fully analog VMM chip

The main block in scientific computing circuits and the most frequent operation in neural networks is the VMM operation (Fig. 47a). The design of fast, energy-efficient, and compact medium-precision VMM circuits is essential. RRAM devices can be used to implement a very efficient VMM block since they can store the neural network weights as the conductance of the devices and perform the computation simultaneously using Ohm’s and Kirchhoff's law. In

Fig. 47a, the VMM operation is defined by $X = WY$ in which Y is the input vector, X is the output vector, and W is the weight matrix. Fig. 47b shows the analog implementation of the VMM operation with memristive arrays. At the intersection of each column and row, a memristive device with the conductance of G_i is implemented. The predetermined weight vector (W) is encoded to the conductance of the devices (in order to implement the differential weights [35], in practice, each synapse of the network is mapped to two RRAM devices). Each device generates current proportional to the amplitude of the input signal (V) times its conductance (G). As a result, the multiplication operations are performed in parallel using Ohm's law ($I = GV$). The currents in all columns (bit lines) are summed up according to Kirchhoff's law. Hence, the sensed current vector (I) is the current-encoded version of the response vector (i.e., X).

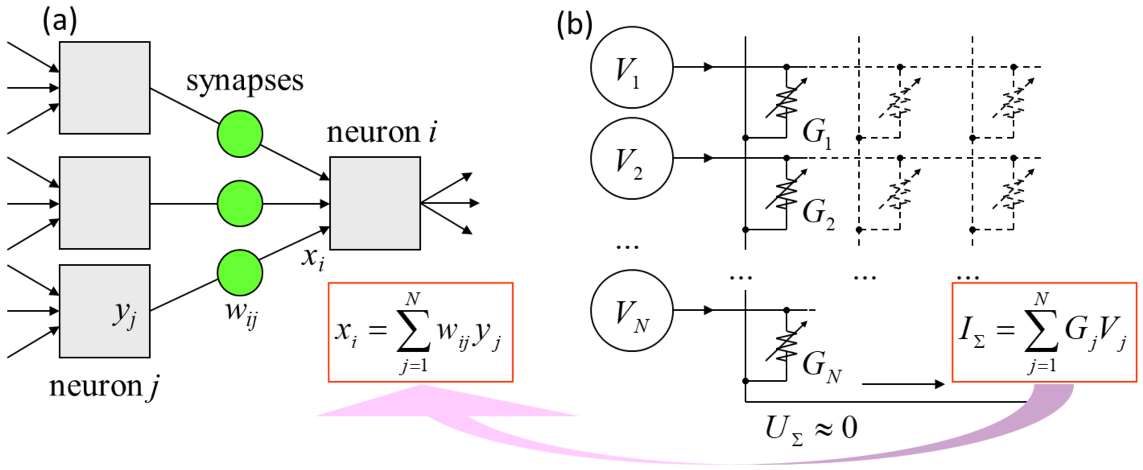
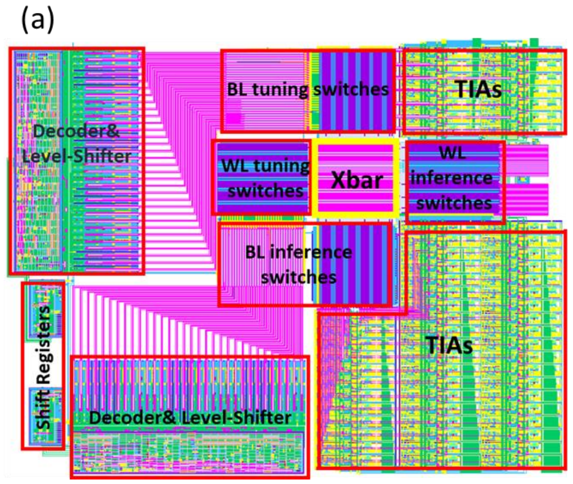


Fig. 47: (a) VMM operation as the basic neuromorphic operation and (b) analog implementation of VMM with memristive arrays.

In a fully analog VMM, direct analog inputs are applied using off-chip high-resolution DACs, and on-chip sensing circuits (transimpedance amplifiers) are designed to read the generated currents. The outputs (voltages in this case) are then measured using off-chip ADCs. In other words, a fully analog VMM includes an X-block coupled with sensing circuits. Fig.

$46a_1$ and $46a_7$ show two chips that prototype 64 × 64, and 32×32, 16×16, 8×8, and 4×4 fully analog VMMs. The goal of these designs is to test the impact of VMM size on its accuracy and performance. The layout of a 64 × 64 fully analog VMM is provided in Fig. 48a. transimpedance amplifiers (TIAs) are employed as peripheral (sensing) circuits to sense the output current in analog VMMs and are the most energy and area-consuming components.

Fig.48 shows the schematic of a TIA connected to the BL electrode of a memristor crossbar (note that tuning and inference switches are not shown). The maximum input current sensed by a TIA depends on the number of devices (N), maximum input voltage presented on WLs (V_{max}), and the average conductance of the devices (G_{avg}). We used $I_{\max}$ =$(N/2)G_{avg}\,V_{max}$ to obtain the maximum required current and assumed V_{max}=0.1 V and G_{avg}=50 µS. Note that the factor of 0.5 is also added to account for the fact that, in most normal VMM cases, not all inputs are always at V_{max}. Regardless, the exact value for $I_{\max}$ is typically determined by the targeted network and is obtained after the training via simulations. The feedback resistor is designed by dividing the dynamic range of output voltage by maximum input current, $R_f = V_{out\_d}/I_{max}$. In addition, the gain of the amplifier is designed such that the input resistance of the TIA ($\sim R_{F/}(1+A)$) is less than 1% of the total average resistance seen from the crossbar (R_{BL}), leading to $A \sim 100R_f\,NG_{avg}$. This will ensure the error due to the voltage drift on the inverting terminal of the amplifier is minimized.



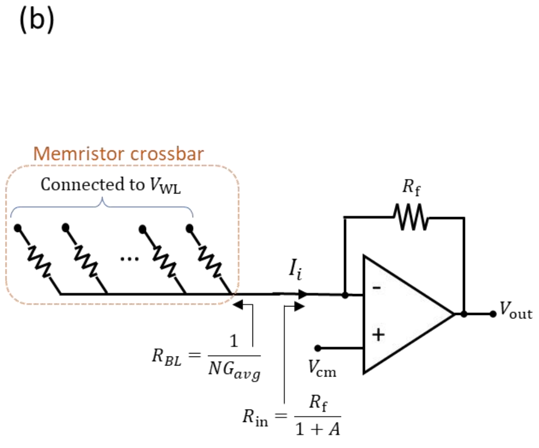


Fig. 48: (a) The layout of a 64×64 fully analog VMM (b) the TIA used for current sensing in fully analog VMMs.

A two-stage amplifier shown in Fig. 49 is considered to implement the amplifier in the TIA. The recycling folded cascode [104] is used as the first stage and a class-AB buffer as the second stage. The key point in recycling topology is that it can provide up to 4× better slew rate and gain-bandwidth product for the same power and area in comparison with the regular folded cascode (see [104] for more details). The design of the amplifiers started with a 60 µA power budget and gain requirements which were discussed above. We allow ~100 mV input-referred offset (this will be compensated using the memory array as shown in the experimental section) and stabilize the amplifier across all corners with ~60 closed-loop phase margin. Fig. 50a shows the distributions of phase margin obtained from running 500 Monte Carlo simulations across all corners. Fig. 50b also illustrates stable transient responses in different corners. In all simulations, the TIA is loaded with a 20 fF capacitance load.

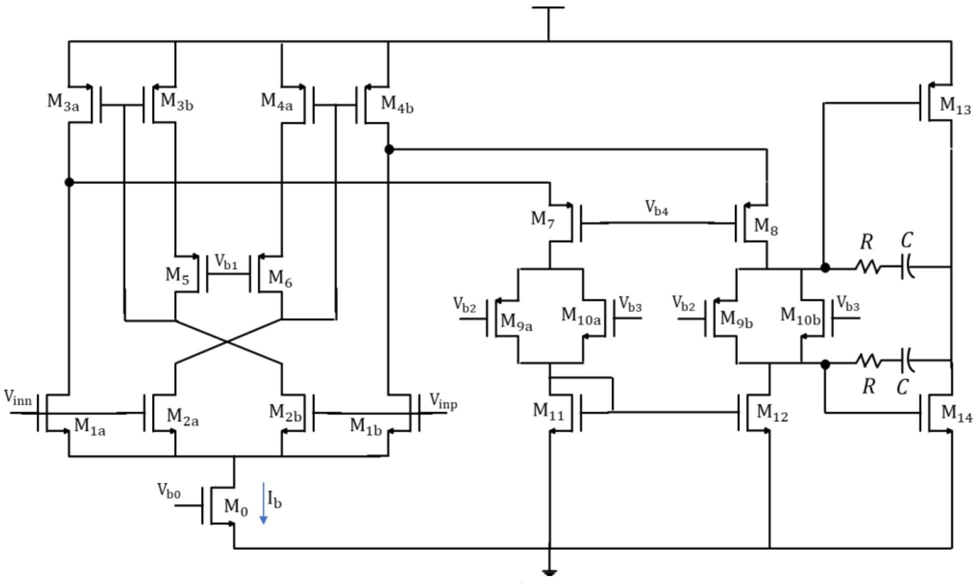


Fig. 49: The two-stage amplifier used in the TIA.

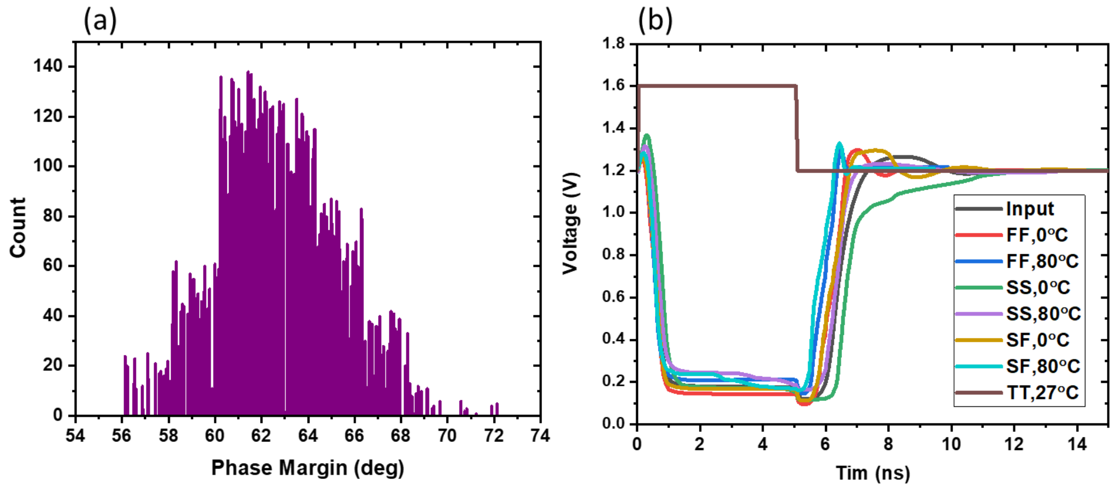


Fig. 50: (a) distributions of phase margin and (b) stable transient responses of the amplifier in different corners.

Fig. 51 shows the simulation results of an analog VMM. In the simulation, the total power consumption includes the power consumed in TIAs and the average power consumed in memory arrays, $P_{\text{array}} = (N/2) \times (V_{\max.}. G_{\text{avg}}) \times V_{\text{dd}}$ in which $V_{\max}$ and G_{avg} are considered 0.1 V and 50 μS respectively, and N is the VMM size. The speed of the VMM is essentially 1% response settling time which is dominated by the TIA slewing and linear settling. The area consumption is the VMM active area (excluding unused silicon and IO pads). In all VMMs, we used a fixed power budget for the TIAs to make a fair comparison. The settling time slightly gets better for larger VMMs (because of smaller feedback resistors in TIAs). Fig. 51a indicates the energy efficiency of a VMM, which changes linearly with VMM size because the number of operations grows quadratically, but energy increases almost linearly. As shown in Fig. 51b, throughput varies quadratically with VMM size as expected. As shown in Fig. 51c, area efficiency grows almost linearly with VMM size because the dominant factors (TIAs, switches, etc.) are linearly proportional to the VMM size.

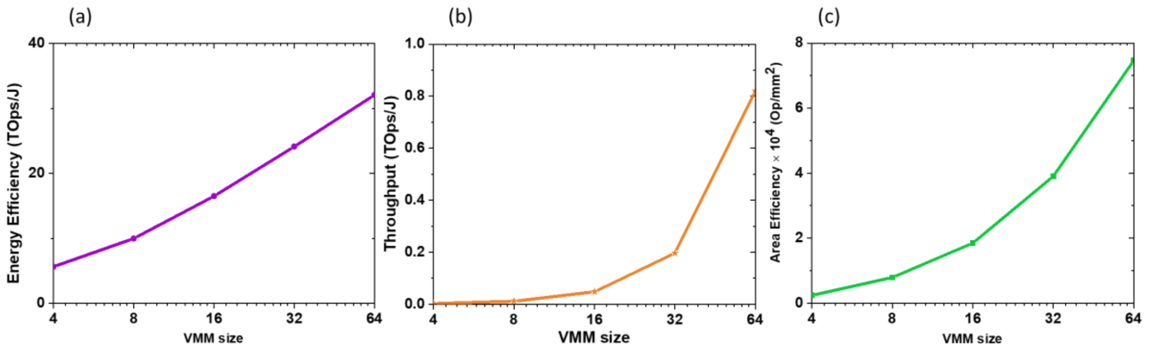


Fig. 51: (a) Energy efficiency, (b) throughput, and (c) area efficiency of analog VMMs versus size.

b) Mixed-signal VMM chip

Chip #4 in Fig. 46a includes different sizes (4×4, 8×8, 16×16, 32×32, 64×64) of mixed-signal VMM circuits, designed to aid the study of the impact of VMM size on the accuracy and performance of mixed-signal VMMs. In mixed-signal VMMs, the inputs and outputs of the VMM circuit are digitally applied/sensed. Hence, DACs are used to drive input WLs and the output currents sensed by TIAs ADCs, as shown in Fig. 52a. For this purpose, we used common data converter architectures such as 5-bit buffered current switching DACs [105] and 5-bit Flash ADCs [106]. The layout of a 64×64 mixed-signal VMM, including DACs, an X-block, TIAs, and ADCs, is provided in Fig. 52b. DACs feature 5-bit resolution, 280 mV full-scale voltage, 361 μW average power consumption, 62.5 MSps conversion rate. The

ADCs also feature 5-bit resolution, 1.6 mW average power consumption, and a 500 MSps conversion rate. Due to the high power consumption of our memory arrays, we focused on maximizing the speed of our peripheral circuits and throughput. Such methodology is especially desirable for small-scale single-shot chips, which typically implement fixed network structures.

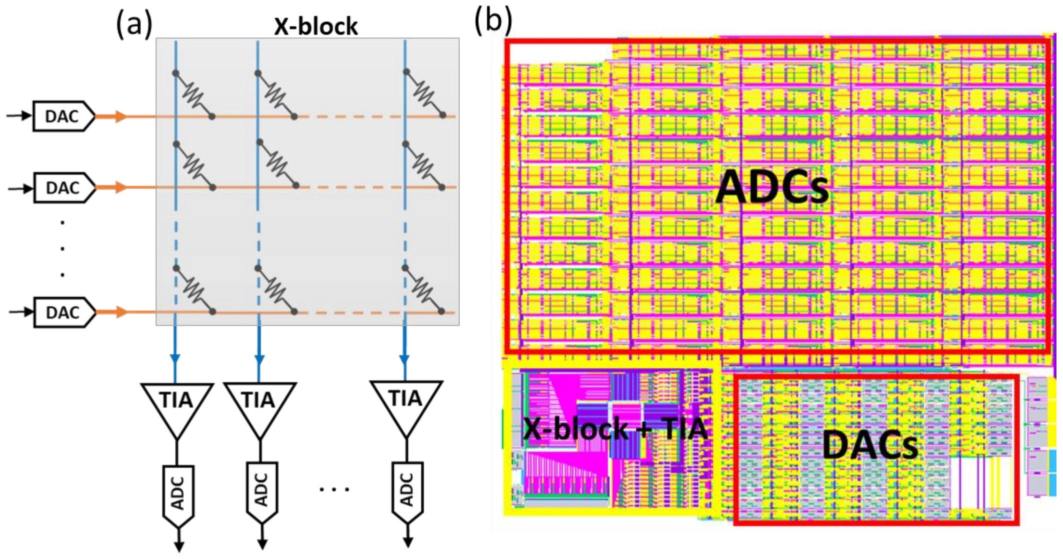


Fig. 52: (a) The schematic of a mixed-signal VMM and (b) the layout of a 64×64 mixed-signal VMM.

Fig. 53 shows the simulation results of the performance of mixed-signal VMMs. Note that the power consumption, area, and settling time include the contributions from TIAs, DACs, ADCs, and the X-block. The trends are similar to the fully analog VMMs. Energy efficiency, throughput, and area efficiency are reduced in comparison with the fully analog approach because of the overhead of the data converters.

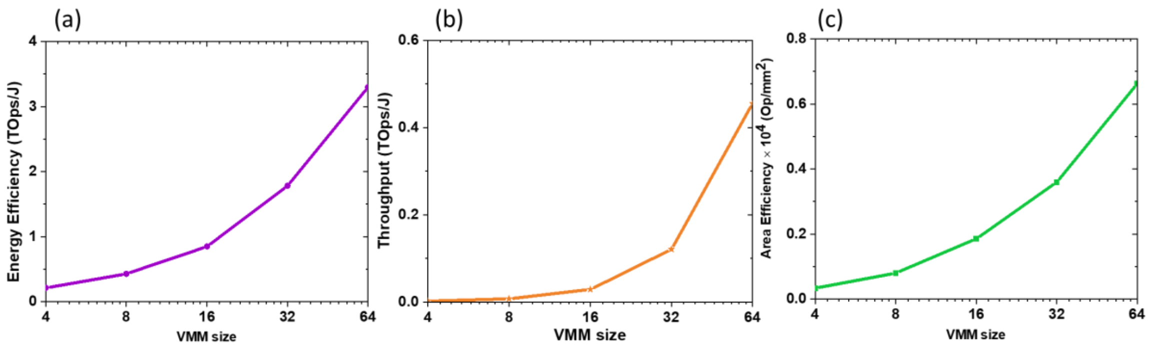


Fig. 53: Simulated (a) energy efficiency, (b) throughput, and (c) area efficiency of mixed-signal VMMs versus VMM size.

c) MLP chip

The third design includes a 3-layer neuromorphic circuit for MNIST handwritten recognition dataset. The design MLP network is implemented with 784 inputs, 64 hidden layer neurons with Relu activation functions, and 10 output neurons (Fig. 46a8). Fig. 54a shows the general structure of the MLP network. The implemented network includes 50,890 synapses (i.e., we need 101,780 synaptic devices) and 74 neurons.

We need a (784+1)×64 crossbar to implement the first layer and a (64+1)×10 RRAM crossbar to realize the second layer. The giant size of the VMM needed in the first layer creates some design problems. The first issue is that increasing the crossbar size (particularly > 64×64) results in the degradation of device uniformity, as discussed in chapter 3. Such an issue increases the spread of switching threshold distributions. Increasing the variation of the switching thresholds and increasing the crossbar size make the half-select problem (see chapter 3) more significant, resulting in a dramatic impact on the tunning accuracy (and hence, larger computation error). Using large crossbars also creates a large parasitic at the input of neurons and increases the IR drop on BLs. To address these issues, we broke the first layer into (2×13) 64×64 mixed-signal VMMs. The factor of 2 is because of differential implementation. Fig. 54b shows the overall designed architecture of the MLP network.

During inference mode of the network, the image is down-sampled to 28×28 5-bit grayscale pixel. The input pattern bits are buffered and shifted serially into 785 DACs, which are responsible for converting digital pixels to analog voltages ready to apply to the first layer. The result of each VMM in the first layer should be summed up before entering the next layer. In order to prevent the IR drop in the path of adding 13 VMM blocks and decreasing the computation error created by the voltage variation on the BLs, the output of each VMM is buffered using a TIA-based local sensing circuit (Fig. 55a). The outputs of the local sensing circuits are then summed up and subtracted (for the differential operation) at the neuron side before applying the activation function and generating the output voltage. The effective current sensed at the jth neuron is given by $I_j = {I_j}^+ - {I_j}^- = \frac{R_{\mathrm{f,LS}}}{R_{\mathrm{x,LS}}} \sum_{i=1}^{785} v_i . (G_{ij}^{+,1} - G_{ij}^{-,1})$ where $R_{\mathrm{f,LS}}$ and $R_{\mathrm{x,LS}}$ are shown in Fig. 55. $G_{ij}^{+,1}$ and $G_{ij}^{-,1}$ are differential conductances in the first layer. The output voltage of the first layer is then given by ${V_o}_j^{l1} = f(R_{\mathrm{f,GS}} I_j)$, where f(.) is the activation function and $R_{\mathrm{f,GS}}$ is its gain. Fig. 55b shows the schematic of the global neuron consisting of a current inverter and a subtractor (to subtract the differential currents) that performs the activation function as well. The second layer is implemented using two (64 + 1) ×10 mixed-signal VMMs. An output neuron (shown in Fig. 55c) senses ${I_{2j}}^+$ and ${I_{2j}}^-$and subtracts them to generate the final differential value. It can be easily shown that $I_{2j} = {I_{2j}}^+ - {I_{2j}}^- = \sum_{i=1}^{65} {V_o}_j^{l1} . (G_{ij}^{+,2} - G_{ij}^{-,2})$ and ${V_o}_j^{l2} = R_{\mathrm{f,2}} I_{2j}$ (which $R_{\mathrm{f,2}}$ is the gain of output neuron) are the output current and voltage of the second layer. Note that for simplicity, we merged the network bias weights with neurons.

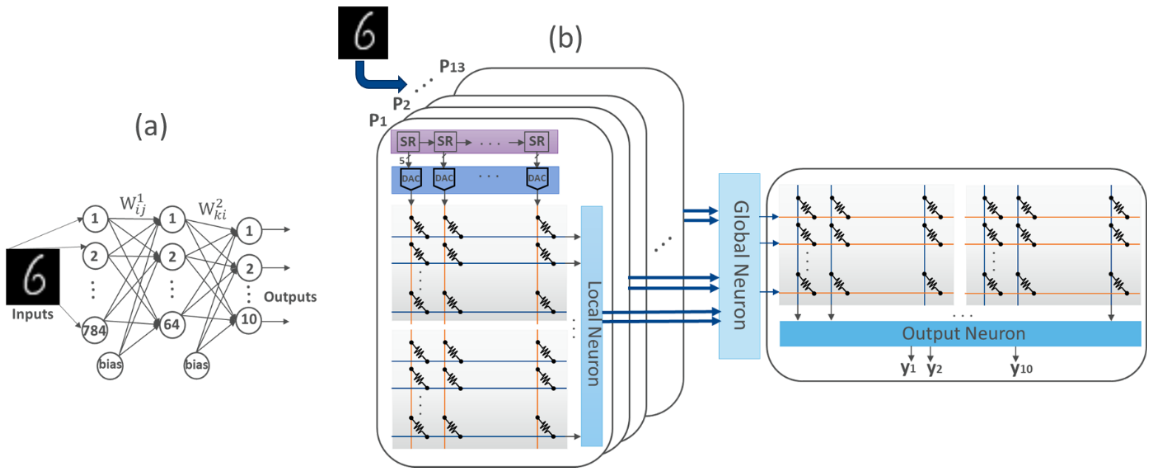


Fig. 54: (a) The general structure of the MLP network and (b) the implemented architecture based on RRAM crossbars.

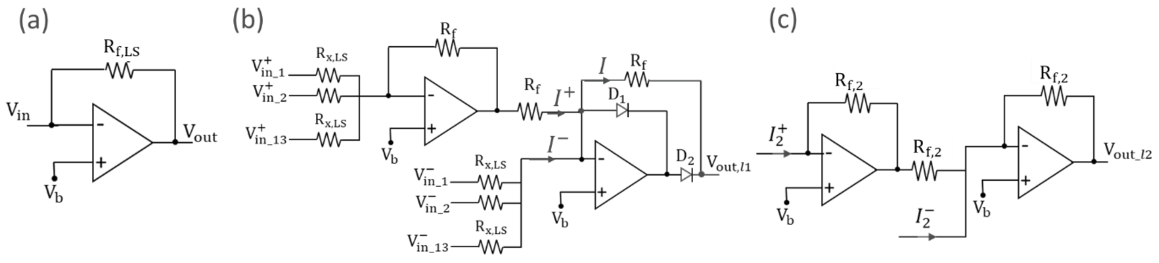


Fig. 55: The schematic of (a) local, (b) global, and (c) output neuron circuits.

Our simulation results indicate that the chip should achieve an energy efficiency of ~1582 TOp/J. Fig. 56b shows the power distribution among the chip blocks. As we observe, a tremendous amount of power is consumed by memory arrays in the first layer of the MLP network. One solution for making the network more energy efficient is to scale the size of RRAM devices, which results in a smaller device conductance and hence lower power. Fig. 56c demonstrates the distribution of the footprint area of various blocks. Regardless of routings and IOs, a considerable portion of the chip belongs to X-Blocks and DACs. Such an area can be reduced by using a more advanced process that provides more metal layers and denser circuits. The area efficiency for this network is ~15075 Op/mm2. We also calculated the worst-case speed of the network by adding the 1% settling time of DACs, local sensing, global neuron, and output neuron circuits, which roughly translates into ~ 66 ns. Hence, the network is expected to achieve a throughput of 1541 GOp/J.

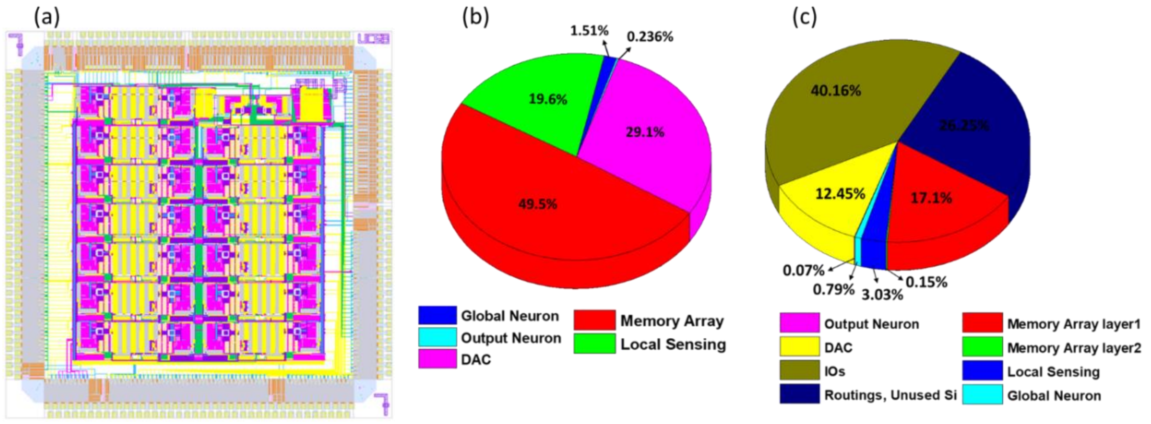


Fig. 56: (a) The layout of the MLP chip, (b) the simulated power distribution, and (c) the area consumption of the MLP network.

d) Other Chips

We have also fabricated multiple other demos, which are briefly described here. In the first run, Chip #9 (Fig. 46a) is a 64×64 fully integrated RRAM-based PUF that was previously prototyped using stand-alone memristors in Ref. [107]. In previous work, the CMOS circuitry was emulated with Agilent tools. The main difference is that the peripheral CMOS circuitries, i.e., row selectors, column selectors, and dynamic comparators, are now fully integrated with the X-block. The comparator block is based on the strongARM topology with a pre-amplifier [108]. Chip #2 and #6 (in Fig. 46a) include many multiplexed 64×64 and 4×4 X-blocks, respectively, designed to study the statistics of memristive devices and crossbars, including their switching threshold distributions, IR drop, *IV* nonlinearity, temperature dependency.

We designed a more advanced PUF topology in the second run: 16 32×32 integrated double-sided fully integrated RRAM-based PUF (see Ref. 125]). The chip has a capacity of generating 1.5×10^{14} CRPs per block at ~1.6 Gbps which is limited by IO speed and serialization of the input challenge. Chip #2 in Fig. 46b includes true random number generator circuits based on 32×32 X-blocks using a novel idea of harnessing the intrinsic noise of RRAM devices for generating dynamic entropy. The other die in Fig. 46d includes a neurooptimization circuit that supports versatile annealing techniques [109] with simulated <680 pS/update across all corners and solves problems up to 64 nodes. The fully dynamic design of the peripheral circuits leads to negligible static power and high energy efficiency.

4.1.3 Preliminary Experimental Results

In the process of CMOS integration, we fabricate the CMOS circuits and then build the RRAM devices at the top of CMOS circuitries on the same wafer in UCSB's nanofab. The integration process is not finished by the time we write this thesis. The second step, fabricating memristor devices, is a very challenging process and requires an enormous effort to adjust the passive memristor process and make reliable devices that can be formed easily. In this section, we discuss the details of some experimental results obtained from testing peripheral circuits.

The CMOS parts designed in the integration demos are fabricated in Silterra's foundry. Besides the previously discussed chips, we also designed a test chip (Fig. 46a_5) that includes only the CMOS peripheral circuits used in other chips. To test the functionality of the CMOS circuits (TIAs, decoders, shift registers, level shifters, etc.), we diced a wafer (with only the CMOS stack), packaged the test chips, and successfully tested the functionality of these circuits. We use a general measurement setup designed in our lab for chip characterization. The setup includes a personal computer connected to an FPGA that controls the analog circuits and systems, e.g., high-resolution DACs, ADCs, voltage/current measurement circuitry, pulse generators, voltage regulators, etc. The testing process and setup are controlled via a MATLAB program running on the personal computer. We design an adapter that connects the chip to the measurement setup and wrote the MATLAB program for performing the intended measurements to test every chip.

More importantly, we packaged the analog VMM chip (Fig. 46a1) to test the TIAs and demonstrate an RRAM-based offset cancelation technique. The monitoring mode (discussed in section 4.2.1e.) is used to test the functionality of the sensing circuit and input voltage generator. Here, we experimentally demonstrate the monitoring mode in the X-block and, for the first time, show offset compensation in TIAs performed with stand-alone memristor devices. Fig. 57 shows the adaptor designed to connect the analog VMM chip to the measurement setup.

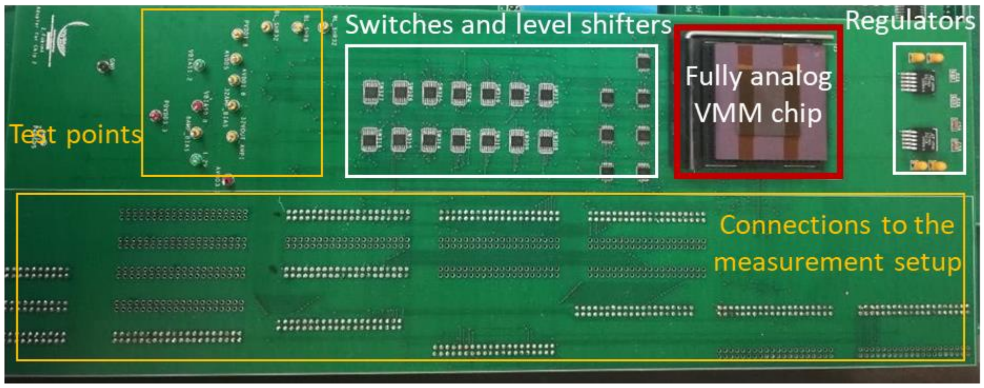

Switches and level shifters
Regulators
Fully analog VMM chip
Test points
Connections to the measurement setup

Fig. 57: The adaptor for testing analog VMM dies (chip #1).

First, let us discuss the dc and transient characteristics of on-chip TIAs. In this experiment, we enable the monitoring mode by activating the decoder and tuning switches according to Table. III. We measured the characteristics of each amplifier used in these VMMs by applying current to the TIA, effectively by changing the output voltage of a designated off-chip DAC. Fig. 58a shows the *IV* characteristic of a TIA in an 8×8 VMM design. The simplified schematic of the circuit used to generate the input current is shown in the inset of Fig. 58a. Here, the V_{cm} voltage is set to 1.2 V and R_f=18.4 kΩ. The DC response in Fig. 58a underlines the impact of the amplifier offset at the output. When the input applied current of the amplifier is zero, the output voltage should be the same as V_{cm} (1.2 V), but as we see, the output is set >1.2 V due to the amplifier input offset. When $I_{in} < 70$ µA, the amplifier is operating in the linear regime. When $I_{in} > 70$ µA, the output stage enters the triode region, the Opamp gain drops, and the output voltage saturates at ~0.1 V. Note that the TIA is designed for the output range of 0.1-1.2 V, which is also observed here. Fig. 58b shows the result of the transient response experiment. The green and yellow waveforms are input and output voltages (see the inset of Fig. 58a), respectively. When the input is low (at 1.2 V), the output voltage is close to 1.2 V (note the impact of offset), and when the input is high (at 1.8 V), maximum designed current enters the TIA, and the output voltage drops close to the saturation voltage of the amplifier.

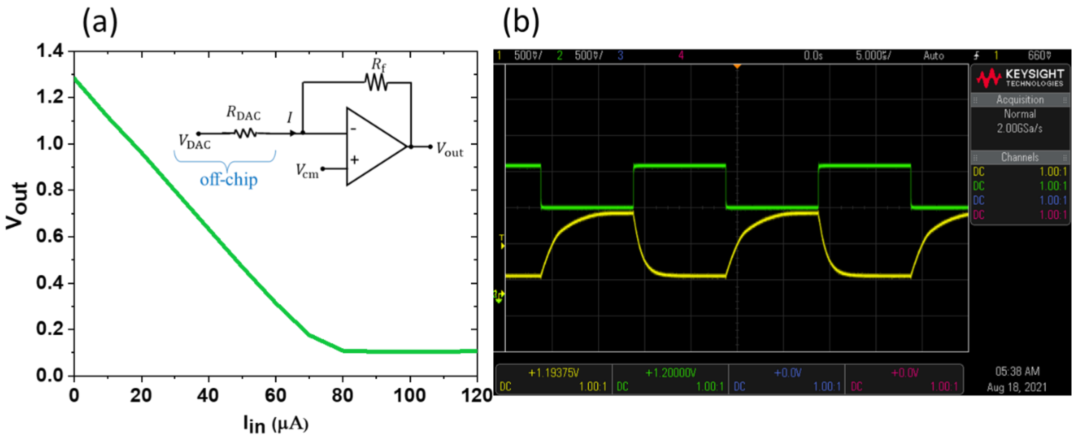


Fig. 58: (a) Measured *IV* characteristics of a TIA and (b) transient analysis of a TIA in the VMM chip.

The offset voltage is a critical parameter of the amplifier performance since it can limit the system's accuracy. This non-ideality originates from the mismatch between the components inside the amplifier. The key parameter to minimize the offset of an amplifier is the length of the driver transistors. By increasing the length of the transistor, we reduce the impact of mismatch between the driver and load transistors. However, this will reduce the overall speed of the amplifier. To avoid this issue, we deliberately allowed up to ~100 mV offset in the design of the TIAs. Meanwhile, we show that we can apply a simple trick to remove the offset from the amplifier. Here, for the first time, we experimentally demonstrate a low-cost offset compensation method that was initially suggested as an idea in Ref. [110] in the context of Flash-based VMMs.

The key idea is to use analog tunable RRAM devices in the array to sink or source a programmable current that cancels out the amplifier's offset. The idea is shown in Fig. 59. The offset is compensated by adding two additional columns, effectively two devices per TIA (G_{c1} and G_{c2} in Fig. 59). For simplicity, we assume that depending on the offset voltage sign, one of the devices (G_{c1} or G_{c2}) is programmed to the lowest conductance state, and the other is programmed to compensate for the offset.

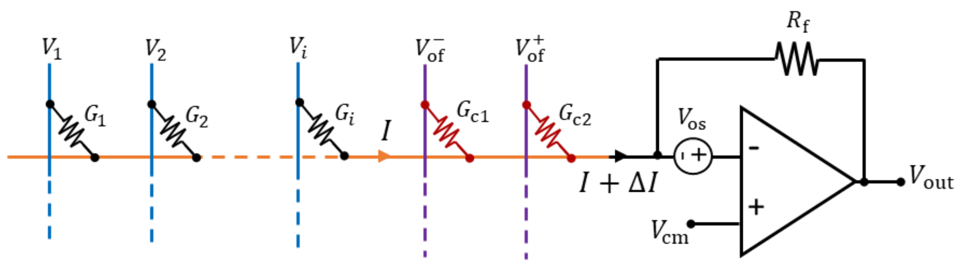


Fig. 59: Offset compensation using two extra RRAM devices.

First, the input current (I in Fig. 59) is set to zero to measure the offset voltage. Then, we assumed that the circuit works in inference mode. So the current that is sensed by the TIA (the VMM output current) is given by

$$I = \sum_{i=1}^{N} G_i \left(V_i - V_{cm} - V_{os}\right) \tag{1}$$

The currents that flow into the compensation devices (G_{c1} and G_{c2}) are:

$$I^+ = G_{c2}\left(V_{of}^+ - V_{cm} - V_{os}\right), \qquad I^- = G_{c1}\left(V_{cm} + V_{os} - V_{of}^-\right). \tag{2}$$

Now, the output voltage of the TIA can be obtained by

$$V_{out} = (V_{cm} + V_{os}) - R_f \, (I + \Delta I), \tag{3}$$

assuming $\Delta I = I^+ - I^-$. For simplicity, we consider $V_{of}^- = V_{cm} - \Delta V$ and $V_{of}^+ = V_{cm} + \Delta V$ in which ΔV is a constant voltage. In our experiments, we consider ΔV=400 mV. To cancel the impact of the input offset, we equalize the output voltage from Eq. 3 to the ideal output of the VMM $V_{\text{out\_ideal}} = \sum G_i \, (V_i - V_{cm})$ to find the proper values of the compensation conductances. Then, we may obtain the compensation conductances (G_{c1} or G_{c2}) using

$$(V_{cm} + V_{os}) + R_f \sum G_i \, V_{os} = R_f \, G_{c2} \, (\Delta V - V_{os}) - R_f \, G_{c1} \, (\Delta V + V_{os}). \tag{4}$$

Depending on the offset sign, we set one of the compensation conductances to the minimum possible value (~6 μS) and adjust the other. For example, if $V_{os} > 0, G_{c2}$ is programmed to 6 μS and G_{c1} is obtained from the following:

$$G_{c1} = \left(R_f (\Delta V + V_{os})\right)^{-1} \left[R_f \, G_{c2} \, (\Delta V - V_{os}) - (V_{cm} + V_{os}) - R_f \sum G_i \, V_{os}\right]. \tag{5}$$

The same process is done for the cases with $V_{os} < 0$ in which we set G_{c1} to 6 μS and find G_{c2}.

In order to demonstrate this technique experimentally, we measured the offset of 40 TIAs in 32 × 32 and 8 × 8 fully analog VMMs. Fig. 60a indicates the distribution of the measured offsets prior to performing the compensation. Then, we used stand-alone memristive crossbars [32] and connected them to the input of the TIA (using the adaptor in Fig. 57). This can be accomplished by operating the X-blocks in the monitoring mode in which we can access the input of any TIA by connecting it to $V_{\text{BL\_SD}}$. The shared electrode of two memristors is connected to $V_{\text{BL\_SD}}$. Note that we considered the general case, and for simplicity, we solved Eq. 4 for the case of $G_i = 0$. We compute the values of compensation conductances for each TIA, program the devices to the desired value, and then measure the offset again at the output.

The devices are programmed (set or reset) with 1% tuning accuracy. Fig. 61 shows how the resistance value of RRAM devices changes during the programming of one pair. Here, the target resistance values for offset compensation are ~41 kΩ and ~170 kΩ.

The distribution of offset voltages for 40 TIAs after applying the compensation method is shown in Fig. 60b. Obviously, the spread of offset has significantly reduced (<1 mV is achieved for all TIAs). The histogram of the effective offset current generated by the memristor devices is plotted in Fig. 60c.

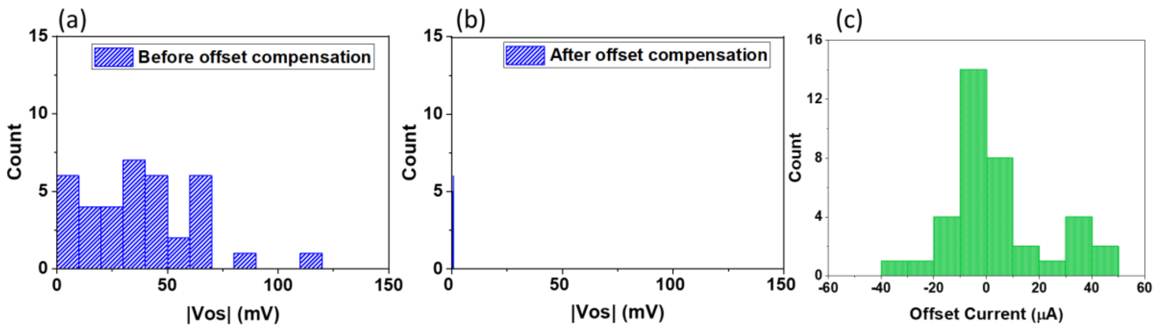


Fig. 60: The distribution of offset voltages for 40 TIAs (a) before and (b) after offset compensation. (c) The distribution of effective offset current generated by memristors.

The experimental results in this section are based on stand-alone memristors and fabricated CMOS. Our results show that we can effectively remove the input-referred offset of amplifiers by using memristors arrays. In future works, we will demonstrate this functionality using integrated RRAM devices.

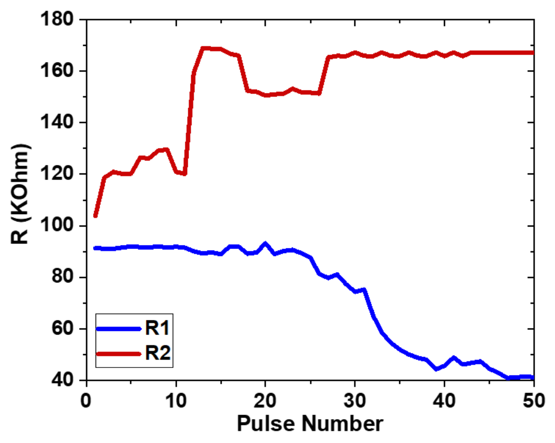


Fig. 61: Programing memristors to states which cancel out the offset of a TIA.

4.2. DNN Accelerator with 1T1R Memories

This section reviews the design and fabrication of a general-purpose DNN accelerator, Cerebro, in 65 nm 7-metal layer CMOS based on industrial 1T1R memories. The chip may be used to accelerate a wide range of neural network inference models. The training of the models is performed ex-situ, i.e., weights are precomputed on a server. In the tuning phase, we transfer the weights into the conductance of memristive devices and store them in 1T1R arrays. Then, the chip is ready to perform the inference task: input data are loaded to the main memory, the controller executes necessary operations to perform the inference task, and the inference results will be computed and stored on the chip.

The architecture of this chip is based on the aCortex design [32]. The inference is performed layer by layer by sequentially reading the input from the main memory, loading them into digital buffers using a router, activating proper VMM and neuron blocks to perform the target dot-product operation, and then the computed results are transferred back to the main memory via a router. The Cerebro chip includes two arrays of VMM blocks, a 32kb SRAM-based main memory, a chain of digital input buffers, data buses, a router, and output neuron blocks which all are designed on-chip. The controller and memory for storing instructions are implemented off-chip. The architecture of Cerebro is shown in Fig. 62. In this design, the number of rows and columns are 44 and 36(×2), respectively. In order to run the inference operation for an image, we first load the image into the SRAM memory bank. Then, we use the routers to buffer different portions of the image into digital shift registers, which drive VMM blocks. Different network layers are mapped to different groups of VMM blocks (see highlighted portions in Fig. 62a), and the inference operation is performed layer by layer. This means that the output of each layer is computed by the VMM, converted to digital by ADCs, and stored in the main memory. The controller that performs these operations is implemented off-chip in this design. In this thesis, we focus on the design of the mixed-signal VMM block shown in Fig. 62b. VMM block consists of a 130×64 crossbar, tuning circuits, DAC arrays, sensing circuits, and logic that enable and disable the operation of the circuit in different modes.

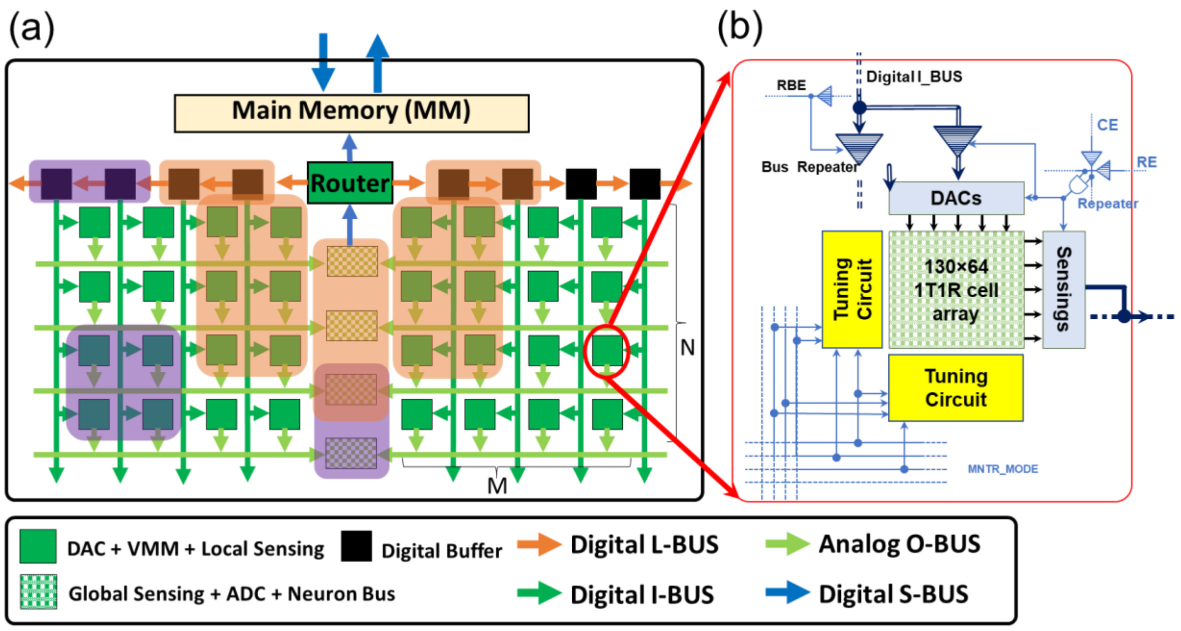


Fig. 62: (a) The top-level architecture of Cerbero and (b) The central computational core.

The entire layout of the VMM and the micrograph of the entire chip are provided in Fig. 63a and 70b, respectively. In future works, we plan to test this chip, and we will provide more details on different blocks and novelties of the design.

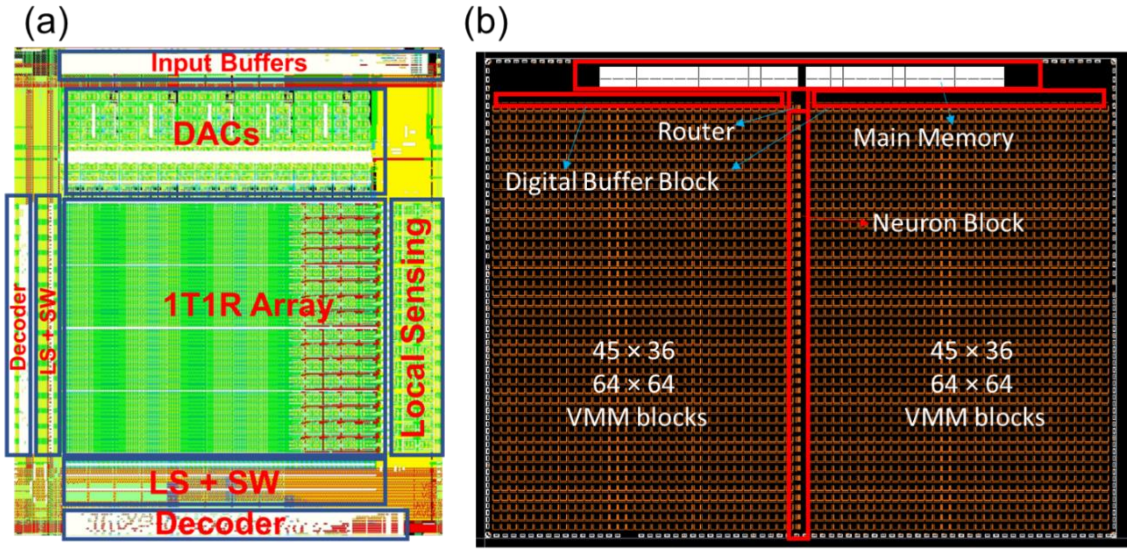


Fig. 63: (a) The layout of a 130×64 1T1R VMM block and (b) the micrograph of the fabricated chip.

5. Summary and Conclusion

Analog-grade memristive technology is a promising candidate for implementing analog in-memory computing systems for several reasons: first, the ability to adjust the conductance in analog fashion; second, the long-term retention of conductance; and third, their incredible scalability. These capabilities of memristor devices allow us to implement VMM operation, which is the main computing block in neural networks, using simple Ohm's and Kirchhoff's law, very efficient, dense, and fast.

In this thesis, we first investigated the nonidealities of RRAM technology and how these imperfections impact the performance of network implementations. We have studied several nonidealities such as temperature dependency, tuning error, and stuck-at fault. In chapter 2, we showed that device imperfections might lead to a dramatic reduction of network performance. We proposed a holistic approach by modifying the tuning procedure, circuit, and training phases of hardware development. Note that, in this approach, the training is still performed ex-situ with negligible hardware cost. The results show that the proposed method has improved the inference accuracy/performance of the network significantly, in particular, by allowing 2.5× to 9× improvements in the energy consumption of memory arrays during inference, sub-percent accuracy drop across 25–100 °C temperature range and increasing the defect tolerance by >100×.

In addition, the impact of device uniformity in passive memristive circuits was also fully studied in chapter 3. We conducted an in-depth analysis of this problem and studied the tradeoffs between computing accuracy, crossbar size, switching threshold variations, and target precision. The impact of crossbar uniformity was studied in two representative deep neural networks, and three solutions, including hardware-aware training, improved tuning algorithm, and switching threshold modification, were proposed to improve the performance. It is shown that these techniques allow us to implement advanced deep neural networks with almost no accuracy drop, using current state-of-the-art analog 0T1R technology.

In the road toward implementing neural networks with synaptic RRAM devices, many operations are inefficient to be implemented with RRAM devices, yet flexible CMOS circuits are promising. To address this issue, we designed and taped out several neuromorphic networks and prototype demos based on CMOS/memristor integration with memristive devices fabricated at the top of CMOS circuitry in BEOL or MEOL processes. Two large-scale wafers, including different sizes of VMMs, an MLP network, etc., are designed in 180 nm CMOS technology. We also tapped out a gigantic DNN accelerator in 65 nm integrated with industrial 1T1R memristor devices.

For future works, we would like to test the DNN accelerator and neuromorphic chips designed based on CMOS/memristor integration after fabricating RRAM crossbars on the top of the CMOS circuitry in UCSB's nanofabrication facility. Designing very compact chips is another interesting future goal that can be achieved by scaling either memristor or CMOS technology process. By scaling the design, we can also improve the energy efficiency of the neuromorphic circuits. Peripheral circuits are more power-hungry parts in the design. So, designing very efficient peripheral circuits using other topologies is also another method for improving efficiency. More advanced DNN benchmarks such as MobileNet, ResNet, etc., can be implemented with passive technology. Improving the RRAM yield is another important future work on the road toward commercializing these technologies.

References

[1] He, K., *et al*. "Deep residual learning for image recognition.", *IEEE conference on computer vision and pattern recognition*. 2016.

[2] Huang, G., *et al*. "Densely connected convolutional networks.", *IEEE conference on computer vision and pattern recognition*. 2017.

[3] Xiong, W., *et al*. "The Microsoft 2017 conversational speech recognition system.", *ICASSP*, 2018.

[4] Xu, X., *et al*. "Scaling for edge inference of deep neural networks.", *Nature Electronics*, 2018

[5] Silver, D., *et al.* "Mastering the game of Go with deep neural networks and tree search.", *Nature*, 2016.

[6] Akopyan, F., *et al*. "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip.", *IEEE transactions on computer-aided design of integrated circuits and systems*, 2015.

[7] Mahapatra, N. R., and Balakrishna, V. "The processor-memory bottleneck: problems and solutions.", *Crossroads*, 1999.

[8] Di Ventra, M., and Yuriy, V. P. "The parallel approach." *Nature Physics*, 2013.

[9] Borghetti, J., *et al*. "Memristive switches enable stateful logic operations via material implication." *Nature*, 2010.

[10] Cassinerio, M., *et al.* "Logic computation in phase change materials by threshold and memory switching." Advanced Materials 25.41 (2013):

[11] Jeong, D. S., *et al*. "Memristors for energy-efficient new computing paradigms.", *Advanced Electronic Materials*, 2016.

[12] Ney, A., *et al*. "Programmable computing with a single magnetoresistive element.", *Nature*, 2003.

[13] Chakrabartty, S., and Gert, C. "Sub-microwatt analog VLSI trainable pattern classifier.", *JSSC*, 2007.

[14] Ramakrishnan, S., and Jennifer, H. "Vector-matrix multiply and winner-take-all as an analog classifier.", *VLSI*, 2013.

[15] Merrikh-Bayat, F., *et al*. "High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays.", *IEEE transactions on neural networks and learning systems*, 2017.

[16] Eryilmaz, S. B., *et al*. "Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array.", *Frontiers in neuroscience*, 2014.

[17] Burr, G. W., *et al*. "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element.", *TED*, 2015.

[18] Boybat, I., *et al*. "Neuromorphic computing with multi-memristive synapses.", *Nature Communications*, 2018.

[19] Kaneko, Y., et al. "Ferroelectric artificial synapses for recognition of a multishaded image.", *TED*, 2014.

[20] Boyn, S., *et al*. "Learning through ferroelectric domain dynamics in solid-state synapses.", *Nature Communications*, 2017.

[21] Romera, M., *et al*. "Vowel recognition with four coupled spin-torque nano-oscillators.", *Nature*, 2018.

[22] Milano, G., *et al*. "Self-limited single nanowire systems combining all-in-one memristive and neuromorphic functionalities.", *Nature Communications*, 2018.

[23] Choi, S., *et al*. "SiGe epitaxial memory for neuromorphic computing with reproducible high performance based on engineered dislocations.", *Nature Materials*, (2018.

[24] Kim, K., *et al*. "A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications.", *Nano Letters*, 2012.

[25] Yeon, H., *et al*. "Alloying conducting channels for reliable neuromorphic computing.", *Nature Nanotechnology*, 2020.

[26] Fuller, E. J., *et al*. "Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing.", *Science*, 2019.

[27] Goswami, S., *et al*. "Robust resistive memory devices using solution-processable metal-coordinated azo aromatics.", *Nature Materials*, 2017.

[28] Indiveri, G., *et al*. "Integration of nanoscale memristor synapses in neuromorphic computing architectures." *Nanotechnology*, 2013.

[29] Prezioso, M., *et al*. "Training and operation of an integrated neuromorphic network based on metal-oxide memristors.", *Nature*, 2015.

[30] Ambrogio, S., *et al*. "Neuromorphic learning and recognition with one-transistor-one-resistor synapses and bistable metal oxide RRAM.", *TED*, 2016.

[31] Chen, A., "A review of emerging non-volatile memory (NVM) technologies and applications." *Solid-State Electronics*, 2016.

[32] Kim, H., *et al*. "4K-memristor analog-grade passive crossbar circuit.", *Nature Communications*, 2021.

[33] Bavandpour, M., *et al*. "Mixed-signal vector-by-matrix multiplier circuits based on 3D-NAND memories for neurocomputing.", *DATE*, 2020.

[34] Mahmoodi, M. R., *et al*. "Intrinsic bounds for computing precision in memristor-based vector-by-matrix multipliers.", *IEEE Transactions on Nanotechnology*, 2020.

[35] Fahimi, Z., *et al*. "Mitigating Imperfections in Mixed-Signal Neuromorphic Circuits.", *ArXiv*, 2021.

[36] Klachko, M., *et al*. "Improving noise tolerance of mixed-signal neural networks.", *IJCNN*, 2019.

[37] Mahmoodi, M. R., *et al*. "Versatile stochastic dot product circuits based on nonvolatile memories for high-performance neurocomputing and neurooptimization.", *Nature Communications*, 2019.

[38] Kataeva, I., *et al*. "Towards the development of analog neuromorphic chip prototype with 2.4 M integrated memristors.", *ISCAS*, 2019.

[39] Prezioso, M. *et al*. "Modelling and implementation of firing-rate neuromorphic-network classifiers with bilayer Pt/Al2O3/TiO2− x/Pt memristors", *IEDM*, 2015.

[40] Prezioso, M., *et al*. "Training and operation of an integrated neuromorphic network based on metal-oxide memristors.", *Nature*, 2015.

[41] Adam, G. *et al*. "3-D memristor crossbars for analog and neuromorphic computing applications.", *TED*, 2016.

[42] Sheridan, P. M., *et al*. "Sparse coding with memristor networks.", *Nature Nanotechnology*, 2017.

[43] Choi, S., *et al*. "Experimental demonstration of feature extraction and dimensionality reduction using memristor networks.", *Nano Letters*, 2017.

[44] Yao, P., *et al*. "Face classification using electronic synapses.", *Nature Communication*, 2017.

[45] Shin, J., *et al.* "Hardware acceleration of simulated annealing of spin glass by RRAM crossbar array.", *IEDM*, 2018.

[46] Jeong Y., *et al*. "K-means data clustering with memristor networks.", *Nano Letters*, 2018.

[47] Merrikh Bayat, F., *et al*., "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits.", *Nature Communication*, 2018.

[48] Li, C., *et al*. "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks.", *Nature Communication*, 2018.

[49] Hu, M., *et al*. "Memristor-based analog computation and neural network classification with a dot product engine.", Advanced Materials, 2018.

[50] Moon, J., *et al*. "Temporal data classification and forecasting using a memristor-based reservoir computing system.", *Nature Electronics*, 2019.

[51] Cai, F., *et al*. "A fully integrated reprogrammable memristor–CMOS system for efficient multiply-accumulate operations.", *Nature Electronics*, 2019.

[52] Yao, P., *et al*. "Fully hardware-implemented memristor convolutional neural network.", *Nature*, 2020.

[53] Kirton, M. J., and Uren, M. J. "Noise in solid-state microstructures: A new perspective on individual defects, interface states and low-frequency ($1/f$) noise.", *Advances in Physics*, 1989.

[54] Lee, S. R., *et al*. "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage-class memory.", *VLSIT*, 2012.

[55] Chen, A. "A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics.", *TED*, 2013.

[56] Liu, B., *et al*. "Vortex: Variation-aware training for memristor X-bar.", *DAC*. 2015.

[57] Hu, M., *et al*. "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication.", *DAC*, 2016.

[58] Agarwal, S., *et al*. "Compensating for parasitic voltage drops in resistive memory arrays.", *IMW*, 2017.

[59] Xia, L., *et al*. "Stuck-at fault tolerance in RRAM computing systems.", *ICAS*, 2017.

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

[61] Liu, C., *et al*. "Rescuing memristor-based neuromorphic design with high defects.", *DAC*, 2017.

[62] Chai, Z., *et al*. "Impact of RTN on pattern recognition accuracy of RRAM-based synaptic neural network.", *EDL*, 2018.

[63] Joksas, D., *et al*. "Committee machines—a universal method to deal with nonidealities in memristor-based neural networks.", *Nature Communications*, 2020.

[64] Mammone, R. J., ed. "Artificial neural networks for speech and vision.", Vol. 4. *Kluwer Academic Publishers*, 1994.

[65] Mehonic, A., *et al*. "Simulation of inference accuracy using realistic RRAM devices.", *Frontiers in neuroscience*, 2019.

[66] Joshi, V., *et al*. "Accurate deep neural network inference using computational phase-change memory.", *Nature Communications*, 2020.

[67] Chen, L., *et al*. "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar.", *DATE*, 2017.

[68] Gokmen, T., *et al*. "The marriage of training and inference for scaled deep learning analog hardware.", *IEDM*, 2019.

[69] Bayat, F. M., *et al*. "Memristor-based perceptron classifier: Increasing complexity and coping with imperfect hardware.", *ICCAD*, 2017.

[70] Guo, X., *et al*. "Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm NOR flash memory cells.", *CICC*, 2018.

[71] Chen, S., *et al*. "Wafer-scale integration of two-dimensional materials in high-density memristive crossbar arrays for artificial neural networks.", *Nature Electronics*, 2020.

[72] Chandrasekaran, S., *et al*. "Improving linearity by introducing Al in HfO2 as a memristor synapse device.", *Nanotechnology*, 2019.

[73] Torres-Huitzil, C., and B. Girau. "Fault and error tolerance in neural networks: A review.", *IEEE Access 5*, 2017.

[74] Merolla, P., *et al*., "Deep neural networks are robust to weight binarization and other non-linear distortions.", *ArXiv*, 2016.

[75] Darabi, S., *et al*., "BNN+: Improved binary network training.", *ArXiv:1812.11800*, 2018.

[76] Furber, S. "To build a brain.", *IEEE Spectrum*, 2012.

[77] Ambrogio, S., *et al*. "Equivalent-accuracy accelerated neural-network training using analog memory.", *Nature*, 2018.

[78] Petropoulos, A., *et al*. "Accurate Emulation of Memristive Crossbar Arrays for In-Memory Computing.", *ArXiv*, 2020.

[79] Zhang, L., and Sitte, J. "Hardware-in-the-loop training of analog neural network chip.", *International Symposium on Neural Networks*, Springer, Berlin, 2006.

[80] Moon, S., *et al*. "Enhancing reliability of analog neural network processors.", *VLSI*, 2019.

[81] Miyashita, D., *et al*. "A neuromorphic chip optimized for deep learning and CMOS technology with time-domain analog and digital mixed-signal processing.", *JSSC*, 2017.

[82] Bavandpour, M., *et al*. "aCortex: An Energy-Efficient Multipurpose Mixed-Signal Inference Accelerator.", *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 6, 2020.

[83] Lecun, Y., *et al*. "Gradient-based learning applied to document recognition.", *Proceedings of the IEEE*, 1998.

[84] He, K., *et al*. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.", *ICCV*, 2015.

[85] Bavandpour, M., *et al*. "Mixed-signal neuromorphic inference accelerators: Recent results and future prospects.", *IEDM*, 2018.

[86] Guo, X., *et al*. "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology.", *IEDM*, 2017.

[87] Iddir, H., *et al*. "Diffusion mechanisms of native point defects in rutile TiO2: Ab initio total-energy calculations.", Physical Review, 2007.

[88] Ninomiya, T., *et al*. "Conductive filament scaling of TaO x bipolar ReRAM for long retention with low current operation.", *VLSIT*, 2012.

[89] Mahmoodi, M. R., *et al*. "A Strong Physically Unclonable Function With> 2^{80} CRPs and< 1.4% BER Using Passive ReRAM Technology.", *ISSCCL*, 2020.

[90] Alibart, F., *et al*., "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm", *Nanotechnology*, 2012.

[91] Nili, H., *et al.*, "Hardware-intrinsic security primitives enabled by analog state and nonlinear conductance variations in integrated memristors.", *Nature Electronics*, 2018.

[92] Wang, Z., *et al.*, "Reinforcement learning with analog memristor arrays.", *Nature Electronics*, 2019.

[93] Wang, Q., *et al.*, "A Deep Neural Network Accelerator Based on Tiled RRAM Architecture", *IEDM*, 2019.

[94] Peng, X., *et al.*, "DNN+NeuroSim V2.0: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators for On-chip Training.", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[95] Young-Fisher, K. G., *et al.*, "Leakage current-forming voltage relation and oxygen gettering in HfO x RRAM devices.", *EDL*, 2013.

[96] Fahimi, Z., *et al*. "The Impact of Device Uniformity on Functionality of Analog Passively-Integrated Memristive Circuits.", TCAS-I, 2021.

[97] Mahmoodi, M. R., *et al*. "A Strong Physically Unclonable Function With> 2^{80} CRPs and< 1.4% BER Using Passive ReRAM Technology.", *ISSCL*, 2020.

[98] Strukov, D., *et al*. "Reconfigurable physically unclonable functions based on analog non-volatile memories." *U.S. Patent No. 10,812,084*, 2020.

[99] Mahmoodi, M. R., *et al*. "Ultra-low-power physical unclonable function with nonlinear fixed-resistance crossbar circuits.", *IEDM*, 2019.

[100] Mahmoodi, M. R., *et al*. "ChipSecure: A reconfigurable analog eFlash-based PUF with machine learning attack resiliency in 55nm CMOS.", *DAC*, 2019.

[101] Fahimi, Z., *et al.* "Combinatorial optimization by weight annealing in memristive Hopfield networks." *Scientific Reports*, 2021.

[102] Mahmoodi, M. R., *et al.* "An analog neuro-optimizer with adaptable annealing based on 64× 64 0T1R crossbar circuit.", *IEDM*, 2019.

[103] Mahmoodi, M. R., *et al.* "Versatile stochastic dot product circuits based on nonvolatile memories for high-performance neurocomputing and neurooptimization.", *Nature Communications,* 2019.

[104] Assaad, Rida S., and Jose Silva-Martinez. "The recycling folded cascode: A general enhancement of the folded cascode amplifier.", JSSC, 2009.

[105] Razavi, B. "The current-steering DAC [a circuit for all seasons].", *SSC-M*, 2018.

[106] Stojcevski, A., *et al*. "Flash ADC architecture.", *Electronics Letters*, 2003.

[107] Mahmoodi, M. R., *et al.* "Ultra-low-power physical unclonable function with nonlinear fixed-resistance crossbar circuits." *IEDM*, 2019.

[108] Razavi, B., "The StrongARM latch [a circuit for all seasons]", *SSC-M*, 2015.

[109] Mahmoodi, M. R. *et al*., "An Analog Neuro-Optimizer with Adaptable Annealing Based on 64×64 0T1R Crossbar Circuit.", IEDM, 2019.

[110] Mahmoodi, M. R., and Strukov, D. B., "An ultra-low energy internally analog, externally digital vector-matrix multiplier based on NOR flash memory technology", DAC, 2018.