Analog In-Memory Computing on Non-Volatile Crossbar Arrays

by

Justin M. Correll

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2021

Doctoral Committee:

Professor Michael P. Flynn, Chair
Professor John P. Hayes
Professor Wei D. Lu
Professor Zhengya Zhang

Justin M. Correll

correllj@umich.edu

ORCID iD: 0000-0000-0192-8129

Acknowledgements

I would like to express my gratitude and appreciation to those who have supported me and made this work possible. First and foremost, I thank Professor Michael P. Flynn for his patience and guidance throughout my graduate studies. It has been the greatest privilege to be a member of the Flynn Research Group and to contribute to the domain of Analog Computing. I thank my committee, Professor John D. Hayes, Professor Wei D. Lu, and Professor Zhengya Zhang, for reviewing my proposal and thesis, and for providing valuable feedback. I especially thank Professor Wei D. Lu and Professor Zhengya Zhang for our many discussions on ReRAM accelerator design.

My work wouldn't have been possible without the selfless dedication and efforts from the collaborators mentioned in this thesis. First, I thank Yong Lim, Vishishtha Bothra, Chester Liu, Seung Hwan Lee, and Fuxi Cai for their efforts on our design – the first memristor computer. I also thank DARPA UPSIDE for funding the project. Second, I thank Seungheun Song, Seungjong Lee, Lu Jie, Junkang Zhu, Wei Tang, Luke Wormald, and Jack Erhardt for their efforts on our design funded by Applied Materials. I also thank Applied Materials for selecting our university to build the first BEOL integrated multi-level cell ReRAM accelerator. From Applied Materials, I thank Christophe Chevallier, Deepak Kamalanathan, Siddarth Krishnan, and Nicolas Breil for their guidance on ReRAM bitcells.

Next, I thank the many members of the Flynn Research Group that helped me along my path of learning integrated circuit design and test. I thank Fred Buhler, Peter Brown, John Bell, Boyi Zheng, Rundao Lu, Daniel Weyer, Sunmin Jang, Chunyang Zhai, Matthew Belz, Christine

Weston, and those already mentioned above. I am honored to have fought this fight along with all of you. In graduate school, friendship and comradery are crucial for success. As such, I thank Aramis Alvarez, Adam Mendrela, Gopal Nataraj, Matthew Schwendeman, Jonathan Slater, Frank Salas, and Laura Andre. Whether they realize it or not, each of them came into my life when I needed it most.

Finally, I thank my family for their never-ending love and support. From a young age, my parents, James and Cheryl Correll, planted a seed in me that grew into a work ethic for which I am forever grateful. I thank my little brother, Matthew Correll – his college graduation inspired me to quit my electrician career and to pursue a college education. At last, I thank Katia Tkachenko for her endless love and support. She taught me the importance of addressing my mental health and with her help, I gained confidence in myself to finish this work.

Table of Contents

List of Tables

List of Figures

Abstract

Analog compute-in-memory with resistive random-access memory (ReRAM) devices promises to overcome the data movement bottleneck in data-intensive AI and machine learning. ReRAM crossbar arrays improve the efficiency of vector-matrix multiplications (VMM), which is a vital operation in these applications.

First, we introduce analog compute-in-memory on non-volatile crossbar arrays. Significant challenges exist towards building a complete system to achieve efficient analog VMM. We discuss the challenges at the device, mixed-signal circuit, and system levels. We provide an overview of non-volatile memory technology suited for analog compute-in-memory applications. Finally, we contrast digital and analog crossbar implementations.

Second, we demonstrate the first complete, fully integrated analog-ReRAM CMOS coprocessor. A passive 54x108 ReRAM crossbar array performs VMM in the analog domain. Specialized mixed-signal circuits stimulate and read the outputs of the ReRAM crossbar. The single-chip CMOS prototype includes a RISC processor interfaced to a memory-mapped mixed-signal core. In the mixed-signal core, ADCs and DACs interface to the passive ReRAM crossbar. The RISC processor controls the mixed-signal circuits and the algorithm data path. The system is fully programmable and supports forward and backward propagation. As proof of concept, a fully integrated 0.18µm CMOS prototype with a post-processed ReRAM array demonstrates several key functions of machine learning including online learning. The mixed-signal core consumes 64mW at an operating frequency of 148MHz. The total system power consumption, considering the mixed-signal circuitry, the digital processor, and the passive ReRAM array is 307mW. The maximum theoretical throughput is 2.6 GOPS at an efficiency of 8.5 GOPS/W.

Finally, we demonstrate a 65nm CMOS RISC-V-based DNN accelerator SoC with four integrated ReRAM CIM macros. We tackle the current state of the art on multiple fronts including 8-bit ADC-assisted bit-serial processing for high throughput and energy efficiency, and CIM macro mixed-signal circuit design for full characterization of foundry integrated low-current multi-level ReRAM. For demonstration, we train and map the LeNet 1 DNN architecture onto ReRAM devices on a single chip and run inference. This SoC achieves 14.63 TOPS/W maximum efficiency and 96.8% classification accuracy on the MNIST dataset.

Chapter 1 Introduction

1.1 The von-Neumann Bottleneck

The explosion in AI and machine learning is driving the need for efficient matrix operations. Convolutional neural networks (CNNs) also known as deep neural networks (DNN) depend on large-scale vector-matrix multiplication (VMM). GPUs are an excellent choice for this task because they are much more efficient than CPUs. Nevertheless, GPU systems are still energy-intensive and, therefore, prohibitive for energy-constrained applications. As AI becomes a more prominent component of computing, it is vital to improve its energy efficiency. Energy efficiency is particularly important for energy-constrained edge devices. The slowdown in scaling and the end of Moore's Law makes it urgent to find new approaches.

One of the problems of conventional approaches is that they are based on the von-Neumann architecture and therefore suffer from the von-Neumann memory bottleneck. Even if the compute is very efficient, moving multiplication coefficients from memory to the compute units itself consumes significant energy. This coefficient movement also takes time and therefore introduces latency. Compute-in-memory systems such as those based on resistive crossbars are attractive because they avoid the von-Neumann bottleneck.

1.2 Computing on the Edge

Machine learning can be divided into training and inference. Accurate learning is critical, and therefore training requires high numerical accuracy making it expensive in terms of hardware resources. However, in most cases, training can be performed in advance so that the requirements for inference dominate in practical applications. Performing inference at edge devices close to the sensor interface is essential to reduce communication needs and to improve security. In speech recognition and image recognition, coefficients are learned in advance and deployed to the edge devices. These edge devices are often highly energy-constrained, so it is vital to reduce the energy needed for inference. Fortuitously, we can exploit the lower accuracy needed for inference to reduce power consumption significantly. This low accuracy requirement favors analog computation and has driven a resurgence in analog computing.

1.3 Thesis Contributions

Analog computing in memory (aCIM) on non-volatile crossbar arrays is necessary to break the von Neumann "memory wall" in data-intensive AI applications. Parallel vector-matrix multiplication (VMM) with physical quantities and mixed-signal circuits increases compute throughput and energy efficiency. Especially for low-power edge applications, aCIM with multi-level cell (MLC) ReRAM is superior to digital SRAM-based solutions. Thanks to the non-volatile MLC capability of modern ReRAM, neural network parameters are stored in crossbar arrays with superior density which both alleviates the need for energy-intensive weight retrieval during VMM and supports massively parallel column-wise multiply and accumulate (MAC) operations. Parallel crossbar array operation increases throughput and maximizes energy amortization and breaks the "memory wall".

Despite recent progress in the field, significant challenges remain for practical ReRAM-based aCIM. First, although 1-bit ReRAM is readily available, the MLC ReRAM device stack is not yet fully mature and remains under development. Materials engineering for better ReRAM stack performance and lab-to-foundry transfer are key bottlenecks. This has limited most of the work in the ReRAM CIM field to binary input and binary ReRAM device operation [1-7]. Binary operation offers efficient VMM for certain neural network tasks but is not scalable to larger networks. Multi-bit input and multi-bit ReRAM are crucial for practical applications.

Second, fully parallel MAC operation on multiple crossbar columns is critical to maximize throughput and energy amortization. An advantage of aCIM on crossbar arrays is that the input is applied to multiple crossbar rows simultaneously. Since each row connects to multiple columns, the input is applied to multiple columns in parallel as well. The column-wise MAC and thus entire crossbar array VMM are computed at once in 10's to 100's of clock cycles. Clearly, the number of array rows depends upon the number of processing elements (PE) in a column and their conductance. Too many PEs in a column could lead to saturation and MAC error. Current works [2,3,5-8] are quite limited (<<128 rows) in the number of active rows due to individual ReRAM conductance and column summation circuitry. Therefore, crossbar array design, ReRAM bitcell design, and crossbar column termination circuitry remain open areas of research.

Finally, system integration remains a great challenge. For maximum throughput, crossbar rows and columns should have dedicated hardware. Crossbar arrays are highly dense structures thanks to ReRAM bitcell feature size. This places a critical constraint on the mixed-signal circuitry layout pitch that interfaces to the rows and columns of the crossbar. The interface circuitry contains multiple analog multiplexors, low-voltage read hardware, high-voltage ReRAM programming hardware, and requires extensive layout which must be channelized. Ideally, the pitch of the row and column circuits must equal the pitch of rows and columns of the crossbar. Due to the inherent challenge, recent work [4-6,10] has been quite limited in offering a practical pitch-matched architecture that is massively scalable.

This thesis presents two prototype ICs that tackle these issues and advances the state of the art on multiple fronts. These prototypes demonstrate new circuit and system techniques, that make

ReRAM CIM practical and that avoid the limitations of current CIM techniques. In Chapter 3[1], we introduce parallel charge-domain crossbar array operation for high throughput VMM and linear MAC operation. The prototype IC is comprised of an on-chip RISC host processor and dedicated channelized mixed-signal circuitry to interface to a passive ReRAM crossbar array post-processed on the top of the die. The SoC supports ReRAM characterization, write-verify programming, and parallel matrix operations. The prototype IC achieves fully parallel CIM with 6-bit input applied at each row and 13-bit ADC output at each column. We demonstrate basic neural network functions such as on-chip training and simple classification tasks on a 54x108 ReRAM crossbar array. The mixed-signal core consumes 64mW at an operating frequency of 148MHz. The total system power consumption, considering the mixed-signal circuitry, the digital processor, and the passive ReRAM array is 307mW. The maximum theoretical throughput is 2.6 GOPS at an efficiency of 8.5 GOPS/W.

Chapter 4[2] introduces the second prototype which improves upon the first in the following areas. We introduce ADC-assisted bit-serial processing for increased throughput and energy efficiency. With bit-serial processing, we compute an 8-bit VMM from an 8-bit input vector with minimal latency. We employ a total of nine ADC clock cycles per VMM period – eight for 8-bit MAC and one fast asynchronous ADC conversion. A compact lower-power binary-weighted ADC enables one ADC per bit line pair. A pitch-matched switch architecture seamlessly supports high and low-voltage operations on foundry integrated, multi-bit 1T1R ReRAM crossbar arrays. On-chip circuits support write-verify programming and accurate ReRAM characterization. A RISC-V host processor controls device programming and MAC operations. The prototype IC achieves fully

[1] This work was in calibration with Vishishtha Bothra, Fuxi Cai, Seung Hwan Lee, and Yong Lim
[2] This work was in collaboration with Jack Erhardt, Lu Jie, Seungjong Lee, Seung Hwan Lee, Seungheun Song, Wei Tang, Luke Wormald, and Junkang Zhu.

parallel CIM (multiplex by 2 for bit line column hardware sharing) with 8-bit input applied at each row (256) and 8-bit output at each pair of columns (32). For demonstration, LeNet1 is fully mapped to 2 bits per cell (BPC) ReRAM on three 256x64 CIM macros on the same IC. This SoC achieves a record 14.63 TOPS/W maximum efficiency and 96.8% classification accuracy on the MNIST dataset.

Chapter 2 Analog Computing on Crossbar Arrays

2.1 Introduction

Analog compute-in-memory on crossbar arrays is necessary to break the von-Neumann memory wall. We exploit the physical crossbar structure and emerging materials to perform parallel energy-efficient vector-matrix multiplication with O(1) time complexity. Low-power mixed-signal circuits perform column-wise multiply-and-accumulate operations and parallelism amortizes the power consumption over the entire crossbar. Matrix coefficients are stored in the crossbar in new multi-level non-volatile memory bitcells. This chapter introduces analog crossbar computation and discusses the challenges associated with building an analog compute-in-memory coprocessor. Finally, we discuss CNN weight mapping onto multi-level cell ReRAM crossbar arrays.

2.2 Analog Crossbar Computation

Analog crossbar arrays are biologically inspired structures that model the massive connectivity of the human brain and show promise for energy efficient VMM. Each of the crossbar array columns mimics a single neuron with many parallel synaptic connections and forms the basis for neural-network matrix multiplication. In the conventional Von Neumann architecture, matrix multiplication is a serial process and requires significant time and energy to move the matrix coefficients to and from external memory. Analog crossbar computation with ReRAM and other non-volatile emerging devices promises to break the computing bottleneck in machine-learning applications.

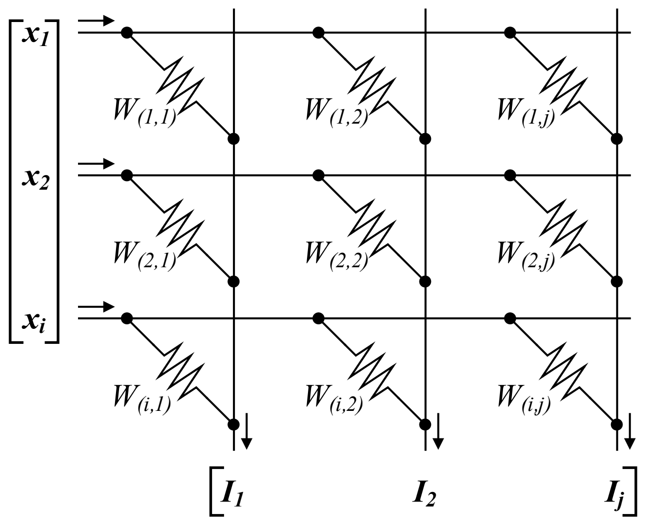


Figure 2-1. Analog crossbar matrix.

Crossbar operation exploits simple physics for very efficient computation. Further, the in-memory nature of this computation removes the memory access bottleneck. As shown in Fig. 2-1, a crossbar is an array of row and column conductors with a resistor connecting the row and column at every row-column intersection. The conductances of these resistors form the multiplicand matrix. The crossbar exploits Ohm's Law and Kirchhoff's Current Law for multiply-and-accumulate matrix multiplication operations. We consider the case where the input vector is a set of voltages applied to the rows while the output is current from the columns. Virtual grounds connected to each column collect the column currents. The output current of each column is the vector-vector dot-product of the input voltages and the crossbar conductances for the column, as shown in Fig. 2-2a.

If a voltage vector x_i is applied to the rows, then the resulting current flowing through the conductance $w_{i,j}$ at intersection i, j is:

$$I_{ij} = x_i w_{ij} \tag{1}$$

The total column current, I_j represents the vector-vector dot product of the input vector with the crossbar column, W_j:

$$I_j = \sum_i x_i W_{ij} = x^T W_j \tag{2}$$

By collecting currents from the columns in parallel, the VMM output is obtained in a single step. Additionally, a crossbar can perform the transpose operation, or backpropagation, by simply applying an input vector to the columns and measuring the current flowing from the rows to the virtual ground (Fig. 2-2b). In this case, the total row current I_i represents the dot product of voltage input vector x_j with the crossbar row W_i:

$$I_j = \sum_j W_{ij} x_j = W_i x^T \tag{3}$$

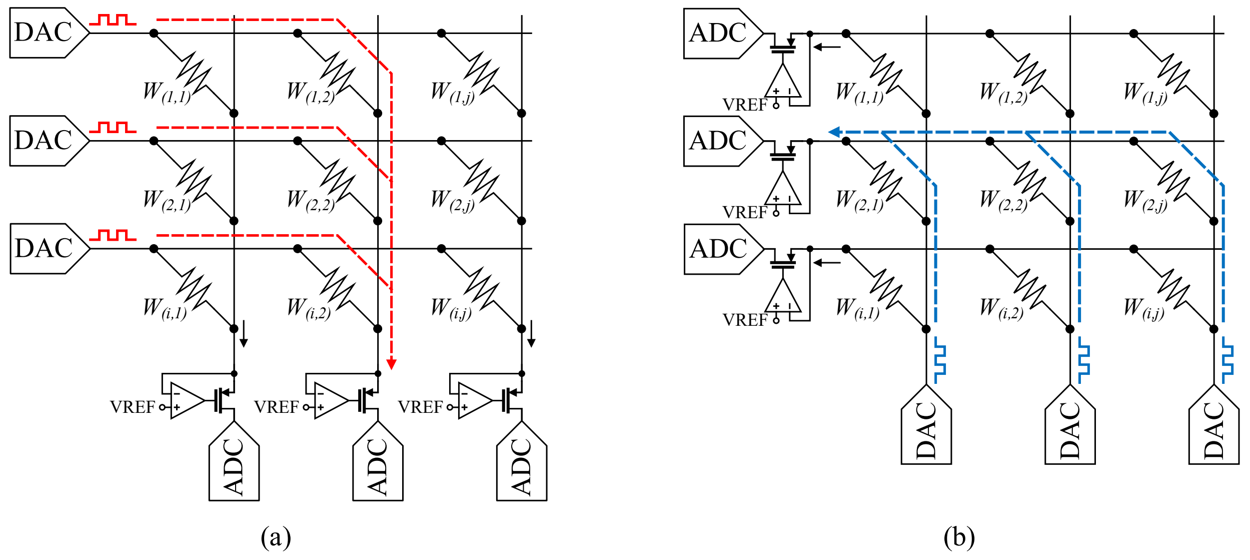


Figure 2-2. Analog vector-matrix multiplication (a), and matrix-vector multiplication with mixed-signal circuits (b).

2.3 Challenges of Crossbar Operation

2.3.1 Device Non-linearity

Before discussing some of the different devices that can be used for crossbar operation, we first review some of the challenges of crossbar operation. The first challenge is that practical crossbar devices, themselves, tend to be highly non-linear. In other words, doubling the voltage across the device does not lead to doubling the current. Two-point operation is an effective way to circumvent this problem. In this approach, the analog input is encoded as a pulse-width-modulated (PWM) signal, and either a fixed read voltage or 0 V is applied across the device. The column output of the crossbar is integrated over the maximum period of the PWM signal. An advantage of this approach is that it enables excellent multiplication linearity even with non-linear crossbar devices. However, there are significant challenges. The PWM nature means that operation takes much longer, in the simplest case 2^N clock cycles, where N is the bit-width of the input vector. Secondly, integration of the column output current is required to determine the analog multiplicand.

2.3.2 Peripheral Circuitry

Another challenge is that effective analog crossbar operation requires extensive analog support circuitry (Fig. 2-3). As discussed, PWM operation enables linear multiplication with non-linear memory devices. The generation of the PWM waveforms is relatively straightforward, as it can be based on synchronous digital circuitry. A drawback of the PWM operation is that the output current must be integrated, and this may require a relatively large capacitance. The requirement for a virtual ground is also a challenge since low-impedance wide-bandwidth virtual ground circuits (e.g., a Transimpedance Amplifier or TIA) can consume significant power. Digitization is also a challenge, as maximum crossbar throughput requires a dedicated ADC per column. For many applications, a relatively low-resolution ADC is sufficient (e.g., 5-8 bits); nevertheless, even low-resolution ADCs can be challenging to place in the narrow memory pitch. Finally, many memory devices require a high voltage for programming – this necessitates the use of high voltage FETs and level-shifters, both of which consume a large area.

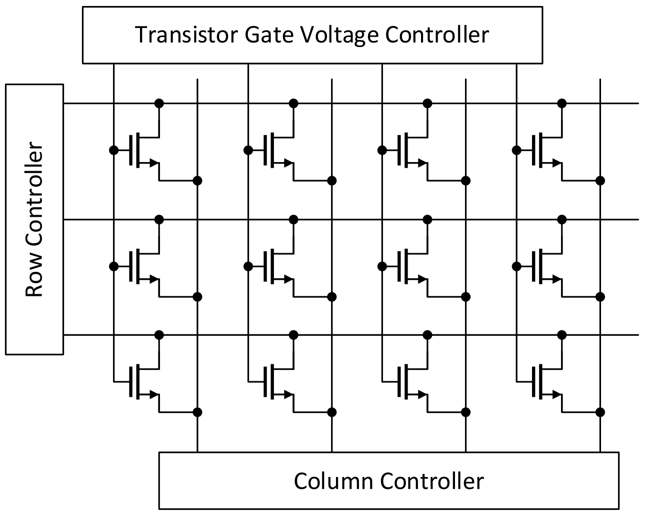


Figure 2-3.Analog crossbar array peripheral circuitry.

2.4 Non-volatile Crossbar Synapses

2.4.1 Flash

Flash memory is one of the best-established non-volatile memory technologies and can be effective for building a crossbar (Fig. 2-4). The threshold voltage of the flash device is programmed to set a crossbar weight. In one approach [11], the flash device functions as a programmable current source. A PWM signal drives the gate, and the duration of this PWM signal represents the input. The output current flows at this weight value during the on-time of the PWM signal. The integrated output current represents the product of the weight and the PWM duration. This current-output operation assumes that the MOSFET is operating in the saturation region.

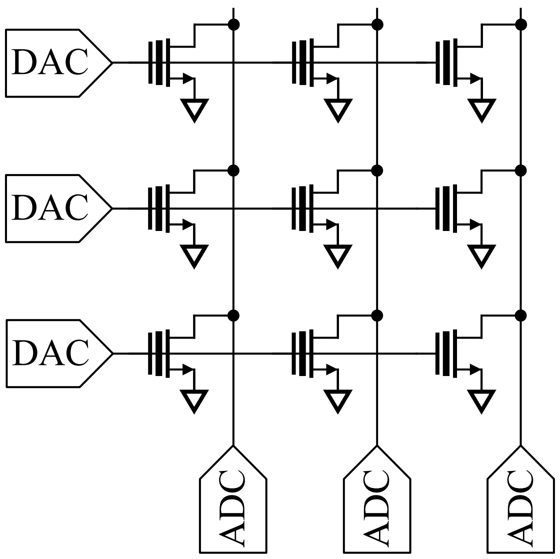


Figure 2-4. Flash crossbar array.

The output currents from the entire column can be collected by a virtual ground and then integrated and digitized. The authors in [11] avoid the complexity and energy cost of the virtual ground by integrating the current on the capacitance of the column line. This column capacitance is initially pre-charged and then discharged by the currents of the flash transistors. A low-resolution voltage-mode ADC digitizes the final column voltage. The non-linear capacitances and the finite output resistance of the memory devices limit the accuracy; nevertheless, this approach is efficient for low accuracy operation.

2.4.2 Emerging Devices

Although flash is a well-established memory technology, flash devices are relatively large. Furthermore, flash transistors require a large programming voltage, limiting the adoption of flash in the most advanced technologies nodes. For practical applications, the crossbar resistance should be both very compact and programmable. These size and programmability requirements have prompted research into emerging memory technologies such as Phase Change Memory (PCM), spin-transfer torque RAM (STT-RAM), and Resistive RAM (ReRAM). Fig. 2-5 compares different technologies.

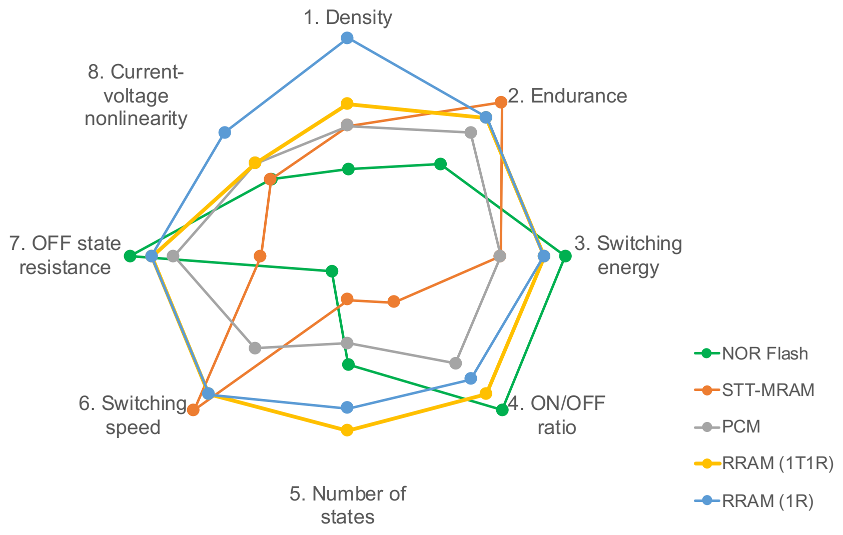


Figure 2-5. Comparison of key performance metrics for different non-volatile crossbar technologies in VMM operations[3].

2.4.3 Resistive Random-Access Memory

Filamentary ReRAM is a metal-insulator-metal structure that has been widely studied for application in crossbar arrays. The memristive synapse is formed by precisely controlling the defects in metal-oxide materials such as HfO_x, WO_x, and TiO_x. Such analog ReRAM devices are CMOS compatible and easily fabricated in scalable, highly dense crossbar arrays that support ultra-low-power compute operations. Crossbar arrays are either passive or active, depending on the bitcell composition, and each type of bitcell presents a unique set of benefits and tradeoffs.

[3]Figure by Seung Hwan Lee

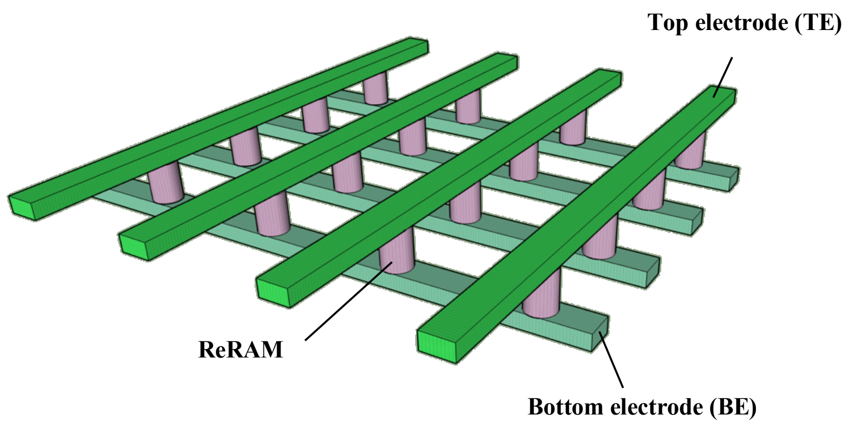


Figure 2-6. Passive ReRAM crossbar array[4].

Passive 1R crossbar arrays are formed with an ReRAM synapse at each array crosspoint (Fig. 2-6). In this approach, the physical size of the array is determined by the effective ReRAM area of $4F^2$, where F is the minimum feature size. The main advantage of passive arrays is their superior compute density. Furthermore, 3D integration is possible and shows promise as a path towards implementing large-scale neural networks with several million parameters. Another advantage is that the passive array structure naturally supports VMM and backpropagation, as discussed in Section 2.2.

A key disadvantage of passive arrays is the sneak-path currents that arise from the inherent parallel current paths present in the array. Multiple current paths cause erroneous currents that affect both weight storage and matrix multiplication accuracy during both the programming and the read operations, respectively. Both conditions ultimately lead to a degradation in network performance or classification accuracy. Common approaches to mitigate the sneak-path problem include half-voltage write-protect programming schemes [12, 13] and intentionally fabricating ReRAM devices with enhanced nonlinearity [14].

[4] [Figure adapted from [15] by Seung Hwan Lee]

A circuit-level solution to the sneak-path problem adds a selector device in series with the ReRAM bitcell. This 1T1R bitcell solves the sneak-path problem by gating the current through the ReRAM with an enable signal. This solution, of course, increases the complexity of the periphery circuitry shown in Fig. 2-7 and increases the area of the bitcell, which in turn drastically reduces the compute density. Generally, the selector device is implemented with a high-voltage IO transistor to support the high voltages and currents associated with programming.

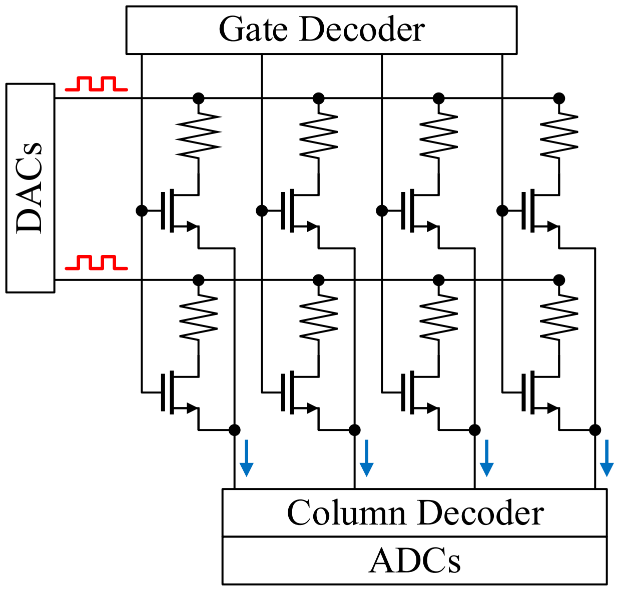


Figure 2-7. Active 1T1R crossbar array.

For metal-oxide ReRAM, the choice of 1R or 1T1R has a critical impact on device-to-device I-V variation. Filament formation in ReRAM is an abrupt process and difficult to control. In passive arrays, "forming-free" ReRAM devices are used, and the device variation is controlled in the fabrication process. The authors in [16] discuss the challenges and paths forward to implementing passive arrays with state-of-the-art device variation and tuning accuracy. In active 1T1R arrays, the virgin ReRAM devices experience a forming step to initiate the oxide filament. A risk in the forming step is that the abrupt formation of the filament can lead to a high conductance state that is unresponsive to device potentiation or "stuck". These high current cells significantly impact matrix multiplication accuracy. The selector transistor ensures current compliance, preventing excessive currents that can damage the ReRAM. The I-V characteristics of the selector transistor limit the current through the ReRAM device. Moreover, the transistor gate is used to precisely control the current flow during programming to overcome device variation during synaptic weight storage.

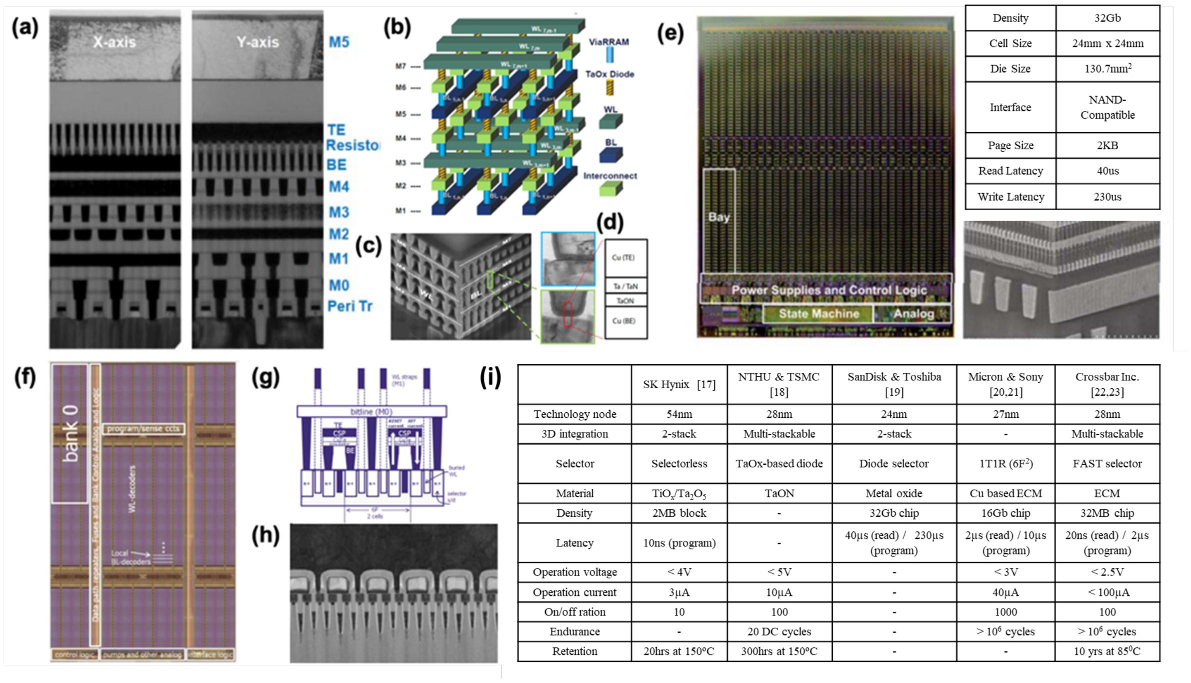
2.5 Digital ReRAM Crossbar

ReRAM is an up-and-coming digital non-volatile memory technology. The very compact bit-cell size makes ReRAM ideal for storing the digital weights for digital VMM. An ReRAM array holds the weights in the simplest case, while conventional CMOS logic performs the vector multiplication itself. Typically, the ReRAM memory is configured as a crossbar structure. A single row is activated to access the memory, and then sense amplifiers connected to the bit lines determine the digital output. The small area is a significant advantage and allows the weight memory to be close to the multipliers. Digital ReRAM is a relatively mature technology and is now commercially available. The table in Fig. 2-8 summarizes a few representative examples from different providers [15].

2.5.1 Analog Operation with Digital ReRAM Cells

A hybrid analog-digital crossbar structure applies single-bit ReRAM elements in an analog fashion. In this mode, the ReRAM is programmed to be either high resistance or low resistance. The two-state ReRAM operation is relatively easy to implement and considered to be more reliable [24].

In the simplest case, there are single-bit weights and a vector of single-bit voltage inputs. Simple digital circuits can generate the single-bit row voltages. A low-impedance virtual ground terminates the bit lines. An ADC digitizes the current flowing into the virtual ground to provide the digital output of the vector multiplication. With the single bit weights and the single-bit inputs, the ADC can be low resolution.



Density	32Gb
Cell Size	24mm x 24mm
Die Size	130.7mm^2
Interface	NAND-Compatible
Page Size	2KB
Read Latency	40us
Write Latency	230us

	SK Hynix [17]	NTHU & TSMC [18]	SanDisk & Toshiba [19]	Micron & Sony [20,21]	Crossbar Inc. [22,23]
Technology node	54nm	28nm	24nm	27nm	28nm
3D integration	2-stack	Multi-stackable	2-stack	-	Multi-stackable
Selector	Selectorless	TaOx-based diode	Diode selector	1T1R (6F^2)	FAST selector
Material	TiO_x/Ta_2O_5	TaON	Metal oxide	Cu based ECM	ECM
Density	2MB block	-	32Gb chip	16Gb chip	32MB chip
Latency	10ns (program)	-	40µs (read) / 230µs (program)	2µs (read) / 10µs (program)	20ns (read) / 2µs (program)
Operation voltage	< 4V	< 5V	-	< 3V	< 2.5V
Operation current	3µA	10µA	-	40µA	< 100µA
On/off ration	10	100	-	1000	100
Endurance	-	20 DC cycles	-	> 10^6 cycles	> 10^6 cycles
Retention	20hrs at 150°C	300hrs at 150°C	-	-	10 yrs at 85^{0}C

Figure 2-8. Summary *of commercially available digital ReRAM technologies [17-23]*[5].

The use of multiple digital ReRAM cells can facilitate multiple-valued weights. One approach uses a separate bit-line for each bit of the weight resolution. For example, in Fig. 2-9, two parallel bit lines are combined for two-bit synapse weights. The current outputs of these different bit lines are weighted and summed. This weighting and summing can be in the digital domain [24] or in the analog domain [2]. Higher input resolutions are enabled by processing the digital input words, serially, one bit at a time and weighting the bit line outputs. Because of its complexity, this approach is limited to small weight and input resolutions, typically two bits.

[5] Details on (a) – (i) found in reference [15]

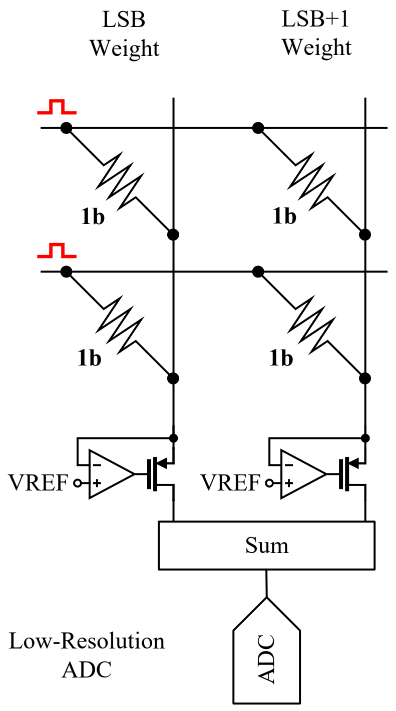


Figure 2-9. Digital ReRAM pseudo-two-bit synapses.

2.6 Analog ReRAM Crossbar

An analog crossbar structure utilizes multi-bit ReRAM at every crosspoint. The ReRAM is programmed to multi-level conductance states. Each ReRAM bitcell is analogous to one weight value in a weight matrix. This allows for a more direct mapping of the VMM or MVM weight matrix onto the analog crossbar (Fig. 2-2). The compute occurs in place and in parallel, thus improving the compute density over the digital ReRAM approach.

Full analog operation requires specialized mixed-signal hardware for programming and matrix multiplication. The degree of parallelization is dependent upon the area of mixed-signal DACs, ADCs, and column multiplexors. Furthermore, many challenges exist in the commercial fabrication and integration of reliable multi-bit ReRAM.

2.6.1 Analog Operation with Analog ReRAM Cells

Analog ReRAM is programmed to multi-level conductance states with dedicated high-voltage DACs. Fixed-amplitude positive and negative pulse trains applied to the ReRAM bitcell increase and decrease the conductance, as shown in Fig. 2-10. After each programming step, the conductance value is determined by reading the selected bitcell.

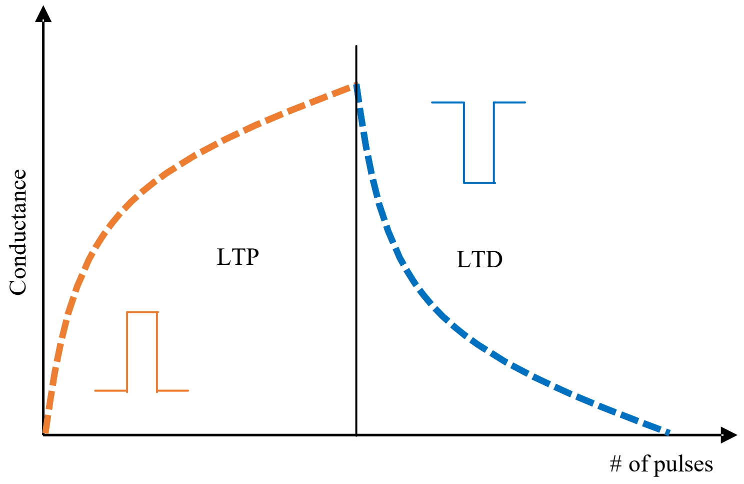


Figure 2-10. ReRAM long-term potentiation (LTP) and long-term depression (LTD)[6].

To perform multiplication, DACs apply voltage inputs to the analog crossbar rows. The resulting column currents are collected at low-impedance virtual grounds and digitized with moderate-to-high resolution ADCs. Multiplication and accumulation are performed along each column of the weight matrix in parallel, without the need for extra crossbar columns or processing steps.

2.7 Neural Network Mapping on Fully Parallel Crossbar Arrays

Accelerating vector-matrix multiplication in convolutional neural networks is a promising application of analog computing on crossbar arrays. This application is considered "inference-

[6] Figure adapted from [15] by Seung Hwan Lee

only" acceleration. Neural network weights are trained off-line meaning that the gradient calculation during weight update is performed at full precision in software simulation. Finally, the weights are mapped and programmed into CIM crossbar arrays for fast and efficient VMM.

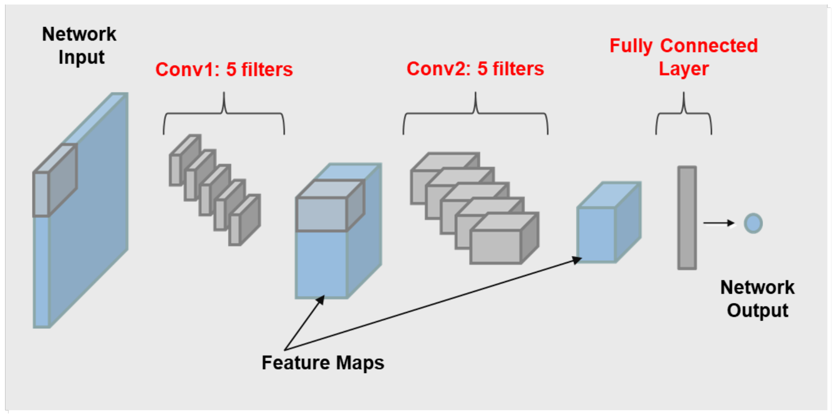


Figure 2-11. Example three-layer convolutional neural network.

Fig. 2-11 shows an example of a three-layer convolutional neural network with 5 filters (or kernels) in the first and second layer and a fully connected layer at the output. Once, the network architecture is defined, the weights are learned through supervised training. After training, the learned parameters are mapped to devices on actual hardware. The goal for neural network mapping is to represent the software-learned parameters with non-volatile multi-level devices.

Conceptually, the mapping process proceeds as follows: (1) unroll each filter into a column vector and (2) store each column vector in a crossbar array column. Fig. 2-12 shows an example of mapping five 5x1 filters to five crossbar columns.

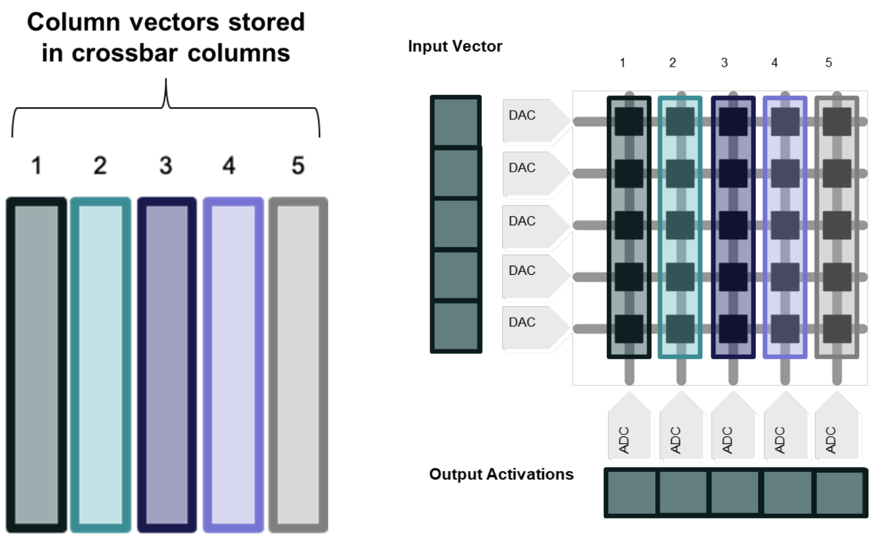


Figure 2-12. Convolutional filters mapped to crossbar columns.

2.7.1 Mapping Challenges

Neural network mapping with non-volatile devices presents two particular challenges. First, typically the parameters learned during neural network training contain negative values. Since analog crossbar arrays operate with currents and voltages it is not straightforward to implement negative weights. Second, traditional training methods do not account for the multi-level cell capability of modern non-volatile memory technology. For example, if a network is trained for 8-bit precision, then the weights are represented in 2 bits per cell (BPC) ReRAM, the network accuracy will significantly degrade due to weight clipping. This process is referred to as post-training quantization.

2.7.2 Quantization Aware Training and Dual Array Mapping

A new approach solves both challenges. First, the network is trained with quantization-aware training (QAT) [25,26]. In this method, quantization is applied in the forward pass during training, while backpropagation is computed in high precision. In this way, the target quantized precision of the non-volatile weights is inferred during training to recover classification accuracy.

The output of QAT is the quantized trained parameters that are programmed onto hardware. A dual array method is used to represent negative parameters in a crossbar array [27,28]. The dual array method maps positive weights onto one array and negative weights onto a separate array. After a MAC operation, digital processing subtracts the two array results before the application of the column activation function. Dual array mapping compensates for low BPC NVM. Since each weight is represented by 2 MLC NVM bitcells, the effective weight resolution is 1-bit higher than the NVM BPC. Accordingly, QAT is performed at 1-bit higher than the NVM BPC resolution. Fig. 2-13 shows an example of dual array weight mapping for a filter in the first convolutional layer of LeNet1 [29].

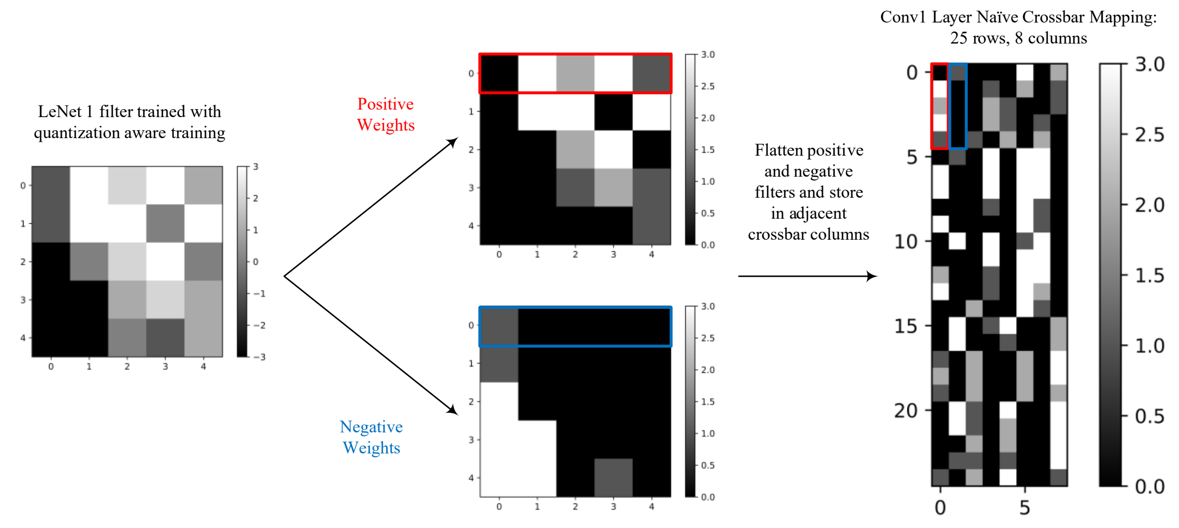

LeNet 1 filter trained with quantization aware training
Positive Weights
Negative Weights
Flatten positive and negative filters and store in adjacent crossbar columns
Conv1 Layer Naïve Crossbar Mapping: 25 rows, 8 columns

Figure 2-13. Example of naive dual array mapping to 2BPC ReRAM devices.

Chapter 3 Memristor-CMOS In-Memory Compute Coprocessor

3.1 Introduction

This chapter presents a complete SoC architecture for parallel aCIM on a passive MLC 1R ReRAM crossbar array[7]. This chapter focuses on the CMOS circuitry that supports RRAM computation and complements [30], which introduces the entire system but focuses on the devices and algorithms. We introduce parallel charge-domain crossbar array operation for high throughput VMM and linear MAC operation. Dedicated row and column hardware compute VMM on an entire post-processed crossbar array with minimal latency. Full parallel operation maximizes throughput and energy amortization. An on-chip RISC processor directs ReRAM characterization, bitcell programming, and MAC operations. This prototype IC achieves fully parallel CIM with 6-bit input and 13-bit output on a 4-bit MLC 54x108 ReRAM crossbar array. We demonstrate basic neural network functions such as on-chip training and simple classification tasks. At an operating frequency of 148 MHz, the maximum efficiency is 8.5 GOPS/W.

[7] This work was in collaboration with Vishishtha Bothra, Fuxi Cai, Seung Hwan Lee, and Yong Lim. The authors key contributions were: Vishishtha Bothra: openRISC modification, verification, and implementation and top-level place and route; Fuxi Cai: software test platform and neural network demonstration; Justin Correll: mixed-signal controller, array switch interface, and DAC design, SOC verification, layout, test hardware, and silicon bring-up; Seung Hwan Lee: memristor device and array design, device integration, and neural network demonstration; Yong Lim: ADC and mixed-signal channel design and layout

3.2 System Architecture

3.2.1 ReRAM Coprocessor

Our self-contained fully programmable coprocessor flexibly maps neuromorphic algorithms onto an integrated 54x108 ReRAM crossbar. The ReRAM crossbar performs VMM operations and is post-processed directly onto the CMOS die. A mixed-signal core with specialized DACs and ADCs applies the input vector to excite the crossbar and digitizes the output vector. The prototype coprocessor, shown in Fig. 3-1, includes a RISC CPU for control and configuration as well as complete mixed-signal circuitry for writing to the ReRAM bitcells and VMM operation. Compute is in place and in parallel, thus maximizing VMM throughput. Parallel operation is supported by dedicated mixed-signal peripheral hardware for each row and column of the crossbar.

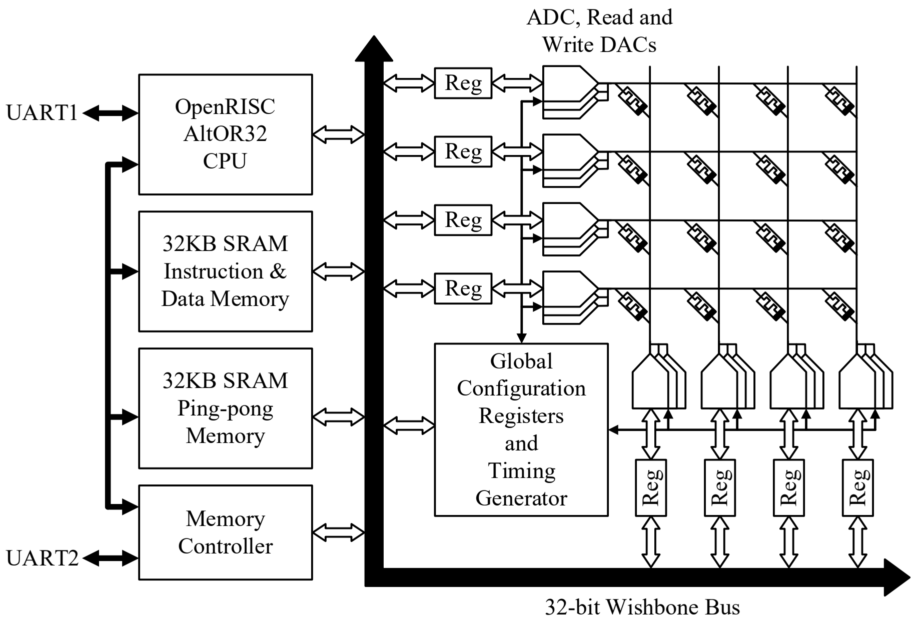


Figure 3-1. ReRAM coprocessor.

Dedicated hardware for each row and column supports read and write operations. Write hardware programs the ReRAM bitcells to the appropriate weight value. The read hardware performs read-verify during write operations and performs VMM operations. During programming, separate row and column DACs apply timed high-voltage pulses to the selected bitcell. After each write step, the updated ReRAM bitcell conductance is read-verified using a row read DAC and a column ADC. Once fully programmed, to perform a VMM, read DACs in each row (column) apply input pulses to excite the ReRAM crossbar and column (row) ADCs integrate and digitize the charge collected at each column. With a full set of mixed-signal circuitry at each row and column, the system supports forward pass (inner product) and backward pass (transpose inner product) operations.

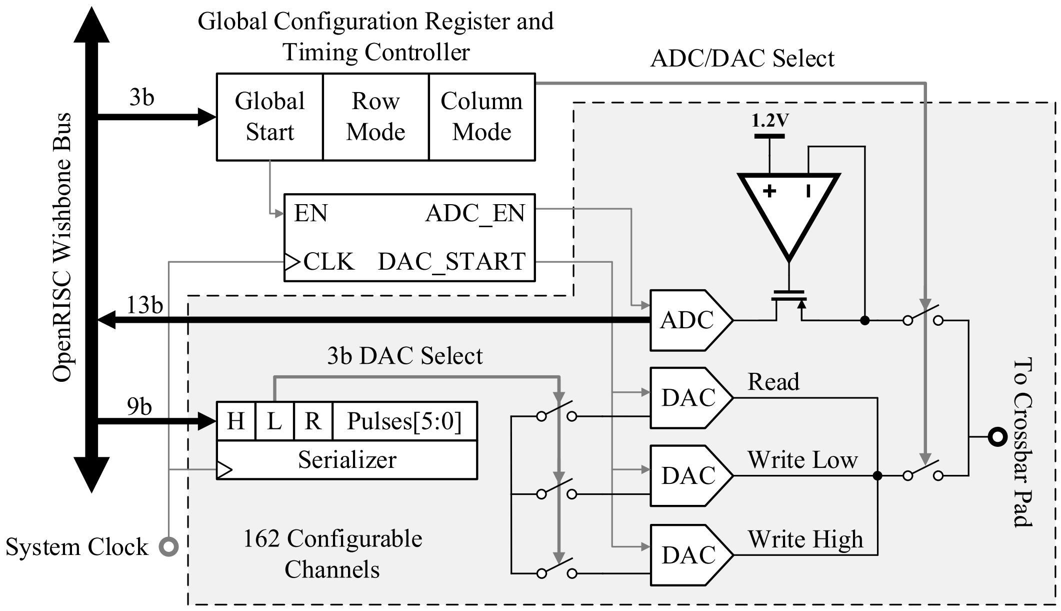


Figure 3-2. Mixed-signal interface.

For easy operation and flexibility, the entire mixed-signal interface (Fig. 3-2), comprised of DACs, ADCs, global timing controller, and configuration registers, is memory-mapped to a custom on-chip RISC CPU. Processor instructions written in standard C code configure the mixed-signal core for different modes and implement ReRAM-crossbar-mapped algorithms. The complete coprocessor facilitates convolutional or fully connected network layers. It handles different activation functions and learning rules_and supports both spiking and non-spiking networks.

3.2.2 ReRAM Crossbar

The 54x108 passive crossbar array is comprised of WO_x-based ReRAM bitcells without selector devices. An ReRAM bitcell is a two-terminal device whose resistance is modulated by the history of external stimulation. ReRAM resistance is both non-volatile and reversible. The bitcell is programmed by applying high-voltage positive and negative pulse trains to opposite sides of the device. Fig. 2-10 shows conceptually how the device conductance changes with the number of positive or negative pulses. Once the bitcell is programmed, a low-voltage pulse train reads the state of the device without perturbing the conductance.

The crossbar structure is formed by fabricating an ReRAM bitcell at the intersection of every row and column of the crossbar array. A passive array, without selector devices, increases the compute density and reduces circuit design complexity, but suffers from sneak-path currents and channel crosstalk [14]. Although selector devices such as those found in 1T1R crossbar arrays [31,32] help mitigate these issues, they increase the bitcell area and require considerable extra circuitry and address decoding.

We address the sneak-path and crosstalk at the bitcell and system levels. In the bitcell, we use devices that exhibit highly non-linear I-V characteristics to minimize parasitic leakage currents in neighboring cells. Fig. 3-4 shows typical I-V characteristics of a WO_x-based ReRAM bitcell. At the system level, during each operating mode, we use protective (write) or common-mode (read) voltage schemes, as described in the next section.

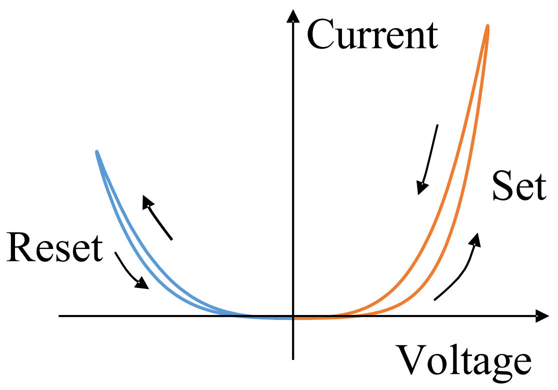


Figure 3-3. ReRAM current-voltage (I-V) characteristics[8].

3.2.3 Charge Domain Operation

Practical WO_x-based ReRAM devices are not suitable for voltage-to-current multiplication because they are highly non-linear. Doubling the voltage across an ReRAM bitcell does not lead to a doubling of the current. In some cases, increasing the device voltage could re-program the bitcell. Time-based operation with PWM DACs is an alternative, but a conventional PWM DAC introduces significant non-linearity due to the non-zero rise-times and fall-times. Finally, the column ADC must present a robust virtual ground to the column and accurately integrate and digitize the column charge.

To address the accuracy and linearity challenges, we operate the crossbar in the time and charge domains instead of in the voltage and current domains. The pulse-mode read DACs deliver fixed-amplitude, return-to-zero (RTZ) pulse trains where the number of pulses is equal to the digital input code. An advantage of this approach is that the non-linearity associated with the rise-time and fall-time is proportional to the input and appears as a gain error that can be canceled in software. Pulsed charge-domain operation also relaxes the bandwidth requirement of the virtual-ground regulator. Fig. 3-5 shows example RTZ pulse trains with proportional gain errors. The

[8] Figure adapted from [15] by Seung Hwan Lee

voltage amplitude is selected based on the ReRAM dynamics. The fixed duration pulses modulate with the crossbar array producing "charge packets" that flow into the virtual ground of a new hybrid 13b charge-integrating ADC.

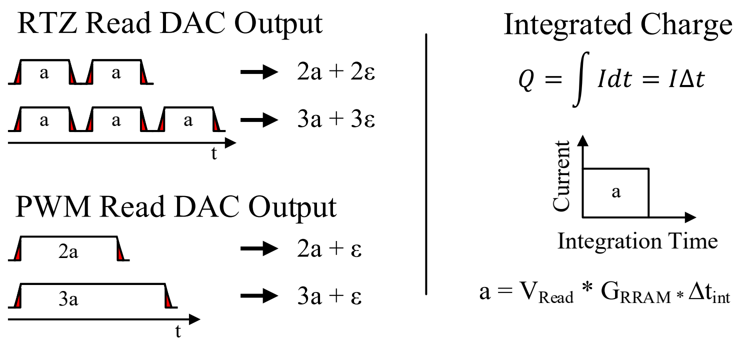


Figure 3-4. Charge-domain operation with pulse-mode RTZ DACs.

We choose a relatively high ADC resolution (13-bits) to accommodate a range of ReRAM parameters and to facilitate a wide variety of VMM applications. High ADC accuracy is also essential for verifying the write of a single bitcell. We decide on the nominal resolution by considering the extremes of operation. The highest integrated charge is for the largest input vector and the highest set of bitcell conductances. With a 6-bit input, 54 rows, and 4-bit ReRAM resolution, the theoretical dynamic range is around 16 bits. In practical applications, the output does not exceed 1/8 of the theoretical maximum so that 13 bits of resolution is sufficient. A 13-bit resolution is also sufficient to measure a single bitcell-conductance to more than 5-bits accuracy during write verify. In summary, 13-bit ADC resolution is a good tradeoff for both online learning and inference tasks, given the size of the crossbar.

3.2.4 Operating Modes

The coprocessor supports five different ReRAM crossbar operating modes – these modes cover write and read operations. The write operations include positive write, negative write, and read-verify and are used to program and verify ReRAM bitcells. Forward read and backward read operations compute the inner-product and the transpose inner-product. A combination of settings in a global configuration register and discrete settings in the row and column hardware determine each mode (Table 3-1). A global timing generator provides the mode timing.

Mode	Row Mode	Column Mode	Row Voltage	Column Voltage	ReRAM Voltage
Positive Write	1 = DAC	1 = DAC	1.9V	0.1V	1.8V
Negative Write	1 = DAC	1 = DAC	0.1V	1.9V	-1.8V
Read-Verify	1 = DAC	0 = ADC	1.8V	1.2V virt. GND	0.6V
Forward Read	1 = DAC	0 = ADC	1.8V	1.2V virt. GND	0.6V
Backward Read	0 = ADC	1 = DAC	1.2V virt. GND	1.8V	0.6V

Table 3-1. Global configuration register settings and crossbar voltage levels.

For the positive and negative write operations, two sets of 6-bit row and column write DACs are used in conjunction to increase or decrease the conductance state of a selected ReRAM bitcell. After each write step, the bitcell conductance is read using a 6-bit row read DAC and a 13-bit column ADC. Only the row and column addresses unique to the selected device are loaded with the input pulse-train data. All other rows and columns are set to a zero-input code so that DACs of the unselected rows and columns output a protective half-voltage to mitigate sneak-path currents.

We now consider the write process in more detail, beginning with a write to increase the conductance of a bitcell. The RISC processor loads the registers for one high-voltage row write DAC and one low-voltage column write DAC with the pulse-train input code equal to the number of desired pulses. The selected row high-voltage write DAC pulses the crossbar row between the 1V protective half-voltage and the 1.9V write high-voltage. At the same time, the selected column low-voltage write DAC pulses the crossbar column between the 1V protective voltage and the 0.1V write low-voltage. The unselected row and column DACs provide the 1V protective voltage for the duration of the pulse train. In this way, the selected device is pulsed with a 1.8V pulse train.

The same operation is performed but in the reverse direction to decrease the conductance of a bitcell, resulting in a -1.8V voltage pulse train. The resultant 1.8V voltage difference, either positive or negative, across the selected device is sufficient to potentiate the conductance of the bitcell, as shown in Fig. 3-3. As an example, Fig. 3-6 shows the crossbar configuration for forward write of the bitcell in row 2, column 2.

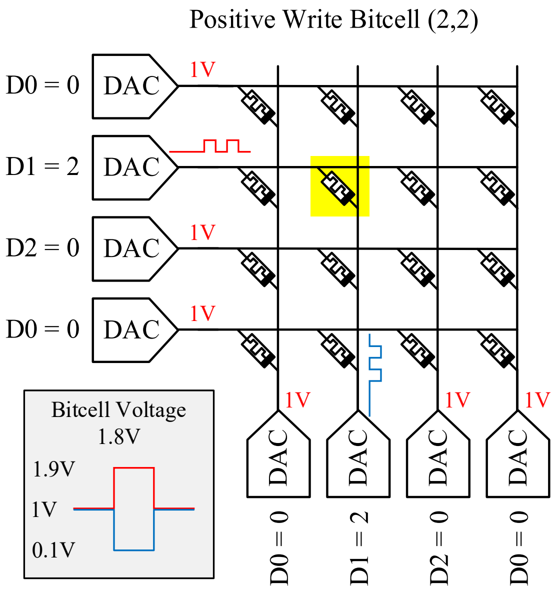


Figure 3-5. Positive write mode bitcell write for row 2, column 2.

For the read-verify step, the processor reconfigures the peripheral hardware to read out the conductance of the selected device. The row and column write DACs are tri-stated from the crossbar, and the row read DACs and column ADCs connected. The processor clears all of the read DAC input registers and loads only the address unique to the selected device is with the input pulse-train. The columns are connected to ADCs which present a strong 1.2V virtual ground. With a zero-input read DAC code, all of the unselected read DACs also output the 1.2V common-mode voltage. The addressed read DAC pulses the crossbar row between the 1.2V common-mode voltage and the 1.8V read voltage. This operation pulses the device with a 600mV pulse train, and the resulting column current flows into the 1.2V virtual ground of the ADC and is digitized. The processor converts the ADC code to a conductance value so that the write step described above can be verified. The write and verify operations repeat until the ReRAM bitcell is programmed to the desired weight value.

During read operations, the entire array operates in parallel for matrix multiplications. In the forward-read direction, the row read DACs are loaded with the input vector data and connected to the crossbar. The column ADCs provide a 1.2V common-mode virtual ground to the columns of the crossbar. The read DACs generate 1.2V-1.8V pulse trains in number equal to the DAC input codes from the input vector. The rows of the crossbar are excited with 600mV pulses, and the ADCs digitize each of the resulting integrated column currents forming the VMM output vector. For backward read, the process reverses, and row ADCs provide a 1.2V virtual ground to the rows, while column DACs apply pulse-trains to the columns. Fig. 3-7 shows the forward read crossbar configuration for VMM.

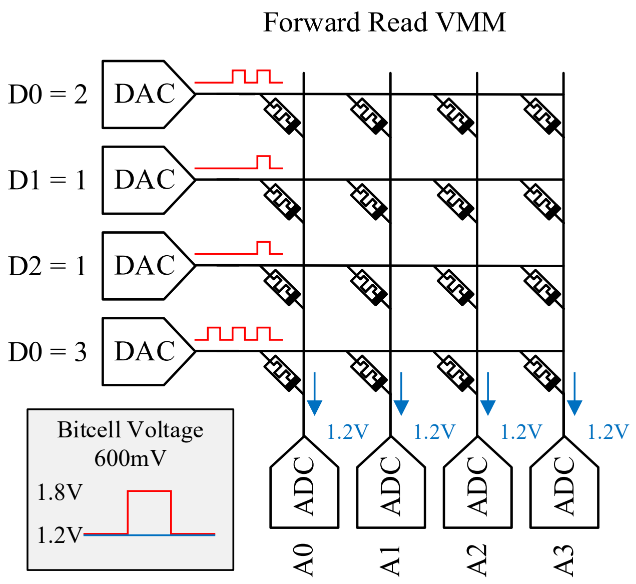


Figure 3-6. Forward read mode for VMM.

3.3 Mixed-Signal Architecture

3.3.1 Mixed-Signal Controller

The mixed-signal controller includes the global configuration register and a global timing controller that configure the array and drive the timing of the peripheral hardware. During operation, the processor sequentially loads all of the memory-mapped mixed-signal registers, then writes a global-start bit to trigger the mixed-signal timing. During the mixed-signal operation, the compute occurs in parallel in the mixed-signal time domain while the processor waits (NOP) a deterministic amount of clock cycles until finished. The control is handed back to the processor and sequentially proceeds to the next step.

The global configuration register (Fig. 3-2) is a one-hot, 3-bit register that controls the write/read mode of all the rows and columns and the start command for the global timing controller. The two lower bits set the row mode (bit 1) and column mode (bit 0). These are global control signals that select whether the DACs (1) or ADCs (0) are connected to the rows and columns of the crossbar, as shown in Fig. 3-2. For example, in forward-read mode, all 54 rows connect to read DACs, and all 108 columns connect to ADCs. In this case, the row-mode bit is set to 1, and the column-mode bit is set to 0. In the forward-write mode, the processor sets both the column-mode and row-mode bits to 1, connecting DACs to all the rows and columns. As explained below, the individual DAC type (i.e., read, write high, write low) is set locally as part of the DAC input code. The DAC output voltages are summarized in Table 3-2.

Mode		RTZ DAC Pulse Low and High Voltage	
Programming	Write High	1V	1.9V
	Write Low	1V	0.1V
	Read	1.2V	1.8V
VMM	Forward Read	1.2V	1.8V
	Backward Read	1.2V	1.8V

Table 3-2. Write and read DAC output voltages.

A global timing generator is a state machine triggered by the MSB of the global configuration register. The period of the mixed-signal timing is equal to the period of the system clock multiplied by the number of clock cycles needed for the 6-bit RTZ DAC operation. The timing generator outputs global control signals, DAC_START and ADC_EN, for the DACs and ADCs and a global stop signal to signal the end of the mixed-signal timing period. The global control signals are active during both write and read operations, but the ADC_EN signal is ignored during write operations. The ADC_EN signal starts and ends one clock cycle before and after the DAC_START signal to power cycle for the ADC.

During write operations, an asserted DAC_START signal enables the selected row and column DACs to output their respective pulse trains. During read operations, ADC_EN wakes up and turns off the power-hungry integrator of the ADC. In a single VMM operation, first, the ADC_EN signal wakes up the ADC, then the DAC_START signal both starts the read DAC pulse train and signals the beginning of the ADC integration period. The ADC integrates until DAC_START is de-asserted, then the collected charge is integrated. The ADC is powered down when ADC_EN is de-asserted.

3.3.2 Pulse-Domain Write and Read DACs

We address the need for a linear time-domain DAC with a 2-level RTZ pulse-mode scheme. A 9-bit input register controls the DACs for each mixed-signal channel. The 9-bit code comprises three configuration bits and the 6-bit DAC value (Fig. 3-2). The three one-hot configuration bits enable either the read (R), write high (H), or write low (L) DAC, while tri-stating the unused DACs. The 6-bit DAC input code is serialized into a 1-bit pulse train where the number of pulses is equal to the value of the input code. The serialized pulse train drives logic that controls two switches for each of the DACs, connected to three off-chip voltage references. The output of the DAC is connected to the crossbar row (or column) and drives the voltage between two off-chip references (determined by the control bits) under control of the DAC logic. The six CMOS switches are sized at 20μm width to minimize the voltage drop from the DAC references.

3.3.3 Active Virtual Ground and Integrating Charge-Domain ADC

An active virtual ground terminates each crossbar output column (or row) and provides a current signal to a dedicated charge-integrating ADC. A high-quality virtual ground is essential so that the row (column) input voltage and the ReRAM transconductance determine the current through each ReRAM device. As we operate the crossbar in the charge domain to avoid non-linearity in the DAC operation, the ADC passively integrates the current delivered through the virtual ground. The ADC digitizes the total charge integrated during the conversion period.

Our approach tackles the challenges of high ADC accuracy and small die area. We implement a programmable current attenuation and a high ADC resolution of 13 bits to support a wide range of algorithms and ReRAM structures. The active virtual ground incorporates programmable current attenuation so that we can accommodate different ReRAM configurations. A novel hybrid ADC architecture is efficient and dramatically reduces the size of the integration capacitor. The 13-bit ADC is a cascade of a 5-bit first stage first-order incremental ADC and a 9-bit second-stage SAR ADC for high resolution. Since directly integrating the current from the crossbar requires large capacitance, instead, the first-stage ADC integrates the charge multiple times with a smaller capacitor and, at the same time, performs coarse digitization. When the integration period ends, the second-stage fine SAR sub-ADC digitizes the residual charge and produces the 13-bit ADC output code.

3.3.4 Active Virtual Ground with Variable Gain Control[9]

Fig. 3-8 shows a simplified schematic of the active virtual-ground circuit. Since the current from the ReRAM crossbar is too large to integrate directly, we divide the current with a programmable attenuation ranging from 1/64 to 8/64. The current divider splits the current from the crossbar into two paths, a dump path (63/64 - 56/64), and a signal path (1/64 – 8/64). Only signal-path current is integrated and digitized by the ADC.

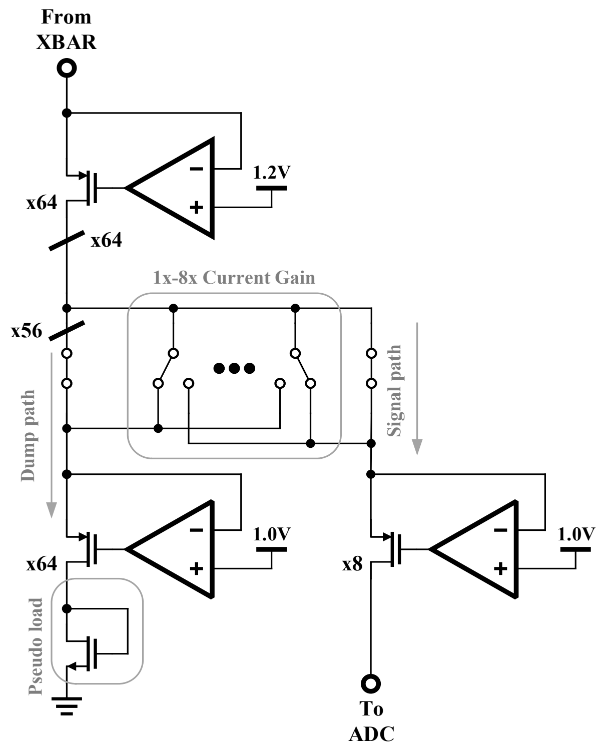


Figure 3-7. Two-stage, active virtual ground with gain control[10].

[9] Designed by Yong Lim
[10] Figure by Seungjong Lee

We introduce a two-stage regulation structure for accurate current division and high output-current linearity. The first stage is a regulated common-gate amplifier that presents a 1.2V virtual ground voltage to the crossbar. The second stage maintains the drain voltage of the first-stage regulator transistor at 1.0V. Only a subset of the second-stage regulated common-gate transistors connect to the ADC – the remainder connects to the dump output. Variable current gain is easily achieved by changing the current ratio using switches. The two-stage regulator provides much higher output impedance compared to a single-stage regulator. This higher impedance improves the linearity of the current supplied to the ADC input. To ensure accurate current division, we load the output of the second-stage dump regulator with a diode-connected pseudo load. Thanks to cascaded regulation and the pseudo load, the simulated current-division error is less than 0.4%, and the full-range current gain variation is less than 0.1%. Each regulation amplifier is an NMOS-input folded-cascode amplifier with an open-loop DC gain of 58 dB, GBW of 16 MHz, and current consumption of 45μA.

3.3.5 Charge Domain 13b Hybrid ADC[11]

The new 13-bit hybrid integrating ADC combines a 5-bit first-stage incremental ADC with a 9-bit second-stage SAR ADC (Fig. 3-9). The first stage is a first-order incremental ADC that performs coarse digitization while alternatively passively integrating the input current on two capacitors, C_1 and C_2. For high resolution, the input charge must be collected and transferred to incremental ADC without leakage. The passive ping-pong integrator consists of a continuous-time hysteresis comparator and three capacitors, C_D, C_1, and C_2. When the integration-enable signal goes high, the reset switches in Fig. 9 open, and capacitors C_1 and C_D begin to accumulate charge.

[11] Designed by Yong Lim

The continuous-time hysteresis comparator senses when the V_X exceeds a threshold, and when this occurs, C_1 disconnects from the input and connects to the active integrator of the incremental ADC. To prevent glitching and other non-ideal switching effects, we incorporate hysteresis in the continuous comparator decision. After C_1 disconnects, C_2 connects to the current input and begins accumulating charge. Non-overlapped clocking for the C_1 and C_2 switches is essential to prevent charge leakage. The extra capacitor, C_D, continues accumulating charge while C_1 and C_2 are disconnected during the non-overlap period, preventing V_X from going too high. When the clocked comparator determines that the output of the active integrator exceeds the threshold voltage, V_{thINT}, then the pre-charged capacitor, C_{CLSB}, connects and subtracts charge in the next integration phase. Since C_{CLSB} is half the size of C_{INT} and pre-charge voltage is V_{REFP}-V_{REFN}, the subtracted voltage V_{CLSB} is (V_{REFP}-V_{REFN})/2. The coarse digital code is the number of subtraction occurrences.

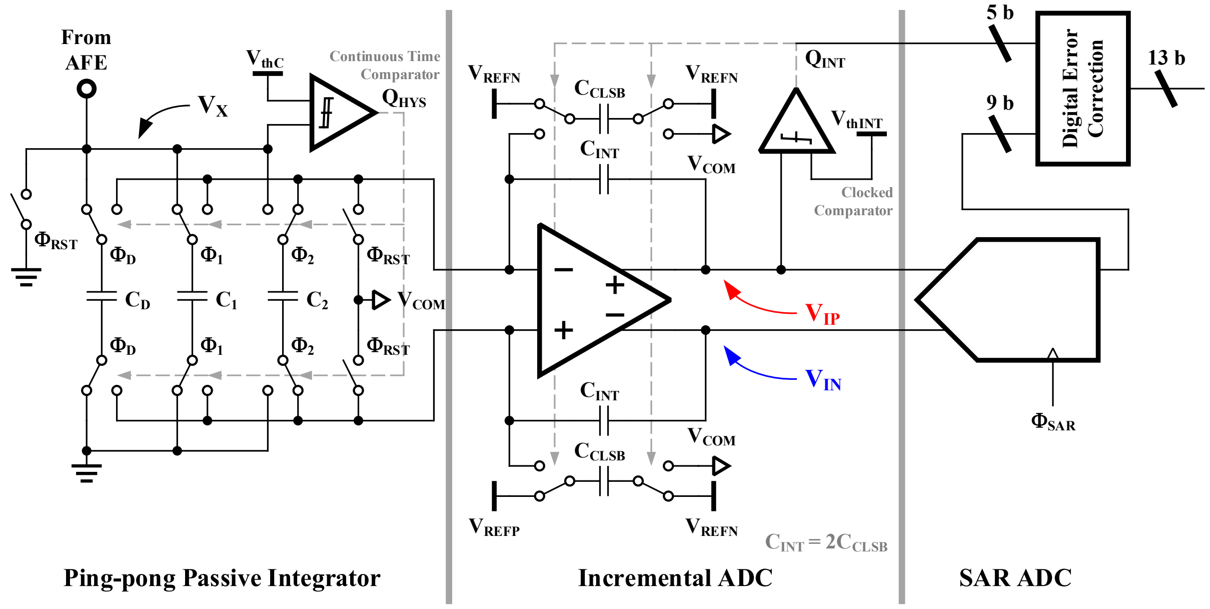


Figure 3-8. Schematic of the hybrid charge integrating ADC[12].

[12] Figure by Seungjong Lee

When the period integration ends, the fine charging capacitor and C_D connect to the active integrator and integrate the charge. Then, the integrator connects to second-stage SAR ADC, which starts fine conversion. The conversion range of the SAR ADC is twice the nominal output range of the active integrator, $2V_{CLSB}$, to provide redundancy. This redundancy accommodates comparator offset as well as any excess output voltage caused by the final residue integration. Considering this one bit of redundancy, the overall resolution of the two-stage ADC is 13-bits.

Lossless charge transfer from the passive integrator is vital for high-resolution ADC operation. The continuous-time comparator is relatively slow, and, therefore during the comparator decision, excessive charge integrates on C_1 (or C_2) and C_D. Furthermore, the input voltage strongly influences comparator speed. Our approach accurately transfers charge, despite inaccuracy in the comparator decision. Fig. 3-10 shows how V_X progresses during the integration phase. The total charge Q_{tot}, after N cycles of passive integration is:

$$\begin{aligned} Q_{tot} = (C_1 + C_D)(V_{thC} + \varepsilon_1) + C_D V_{D1} + (C_2 + C_D)(V_{thC} + \varepsilon_2 - V_{int.1}) + C_D V_{D2} + \cdots \\ + (C_X + C_D)(V_F - V_{int.N-1}) \end{aligned}$$

where ε is the voltage increase during comparator delay and V_D is voltage increase during the non-overlap phase. C_X is the connected capacitor at the final phase; V_F is the final voltage when the integration signal goes low. C_{int} is the initial voltage once a capacitor is connected. During the non-overlap phase, only C_D is connected to V_X. After C_2 is connected, charge redistribution occurs and the initial voltage for the next phase is:

$$V_{int.1} = \frac{C_D\left(V_{ref1} + \varepsilon_1 + V_{D1}\right)}{C_2 + C_D}.$$

Substituting for V_{int}, the total charge is:

$$Q_{tot} = C_1(V_{thC} + \varepsilon_1) + C_2(V_{thC} + \varepsilon_2) + \cdots + (C_X + C_D)V_F.$$

The equation shows that the total charge is transferred without any loss.

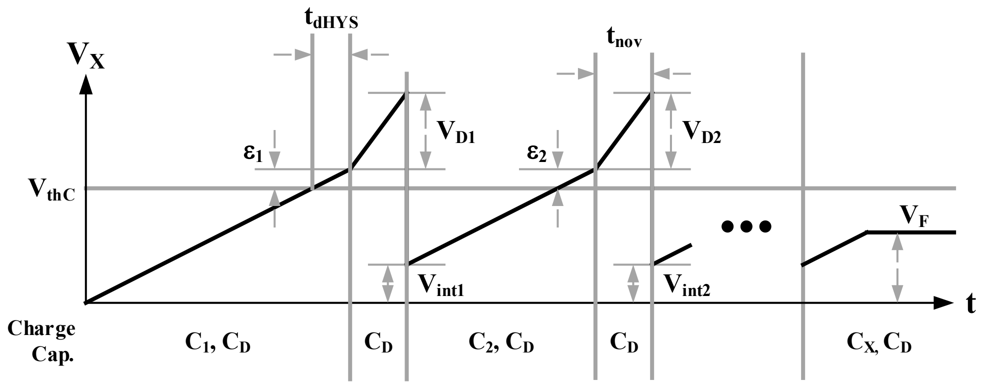


Figure 3-9. Detailed capacitor waveform for the passive integrator[13].

The active integrator uses a three-stage fully-differential ring amplifier [33] with an auxiliary first-stage auto-zero (similar to [34]) to improve energy efficiency. The clocked comparator is a double-tail latched comparator [35]. The SAR ADC employs bottom-plate sampling for accuracy and asynchronous logic for speed. When the integration signal goes low, the SAR ADC samples the output of the active integrator output and begins the SAR algorithm. The overall ADC code is the sum of the first-stage code and second-stage code. The ADC consumes 1.3mW at 19MS/s and occupies 0.066mm^2 (74µm x 890 µm).

3.4 Digital Architecture [14]

A custom OpenRISC processor with 64KB of dual-port SRAM supports and controls the mixed-signal core. The registers for the DACs, ADCs, and global configuration are memory mapped to the OpenRISC address space via the shared 32-bit Wishbone bus. The memory-mapped registers are accessible in a standard C code application. Since the core mainly performs register-level manipulations, a stripped-down version, the Alternative OpenRISC 1000, or AltOR32, was chosen to save area and power [36]. The AltOR32 removes the floating-point unit, hardware

[13] Figure by Seungjong Lee
[14] Processor implementation and digital design were by Vishishtha Bothra

multiplier, and pipeline delay slot instructions and hardware from the implementation. We developed a custom Verilog module that implements the mixed-signal core as a peripheral slave to the 32-bit Wishbone bus. This module places all of the mixed-signal registers in the address space of the processor. The module is treated just like any other OpenRISC peripheral.

The on-chip SRAM is divided into two blocks - 32KB for both processor instruction and data memory and 32KB of "ping-pong" memory. The ping-pong memory can be used for loading input data from off-chip during runtime for large-scale neural network applications. All 64KB of SRAM is dual-port and is mapped to both the OpenRISC 32-bit Wishbone bus and a custom memory controller external to the processor. The memory controller implements a backdoor UART port that provides access to all of the SRAM.

The processor instructions, written in standard C code, are compiled and run on-chip to control the mixed-signal core. A typical C program implements sequential instructions to set up the global configuration registers and peripheral hardware then starts the global timing controller to compute a VMM. The processor waits (NOP) the necessary time until the state machine is finished. The program then reads out the ADC results into memory. In [30] we demonstrated the mapping of complete algorithms on-chip.

The following pseudo code writes the bitcell in row 2, column 2 with two write pulses:

```
// Write bitcell(2,2) with pulse=2
row = 2;
pulses = 2;
row_mode = 1;                               // Select row DACs
col_mode = 1;                               // Select col DACs
for(int i=0;i<162;i++){
DAC[i] = 0;                         // Clear all DACs
}
DAC[row-1] = pulses;
DAC[col-1+54] = pulses;

//Start global timing generator
start();
//Read-verify bitcell
value = read_verify(row,col);
```

3.5 Prototype

3.5.1 ReRAM Crossbar

The W/WO_x/Pd ReRAM crossbar array was patterned using e-beam lithography, etching (for the W bottom electrode, BE), and liftoff (for the Pd top electrode, TE) processes. The WO_x switching layer was created through rapid thermal annealing of the exposed W BE surface with oxygen gas at 425°C for 60 seconds. A SiO_2 spacer structure was formed by PECVD followed by RIE etch-back before WO_x growth and TE deposition to allow for better step coverage of the TE at the cross points. Details on CMOS integration can be found in [30].

The WO_x devices do not require a forming step – a key requirement for passive crossbar arrays. Without selector devices, the high-voltage forming process can damage other already formed cells in the array. Switching behavior in WO_x devices is categorized as bulk-type. Conductance modulation is proportional to the effective conductive area change [37] instead of the opening or closing of the depletion gap found in filamentary (TaO_x, HfO_2) ReRAM devices. Bulk-type switching leads to forming-free behavior, but device retention is limited. For the prototype devices, the device retention is limited to minutes but is sufficient to demonstrate the algorithms in [30]. More details on the WO_x devices and their switching behavior can be found in [37] and [38].

3.5.2 Integrated Coprocessor[15]

The prototype IC is fabricated in 180nm CMOS and occupies 62mm^2 (Fig. 3-11). This includes 162 ADCs, 486 DACs, the OpenRISC processor, 64KB of SRAM, and the integrated

[15] Crossbar array design and post processing of ReRAM was by Seung Hwan Lee at University of Michigan's Lurie Nanofabrication Facility.

ReRAM crossbar fabricated on the surface of the die. A single channel is comprised of 3 DACs and 1 ADC and occupies 0.13 mm^2. Fig. 3-12 shows the post-processed crossbar fabricated on the IC.

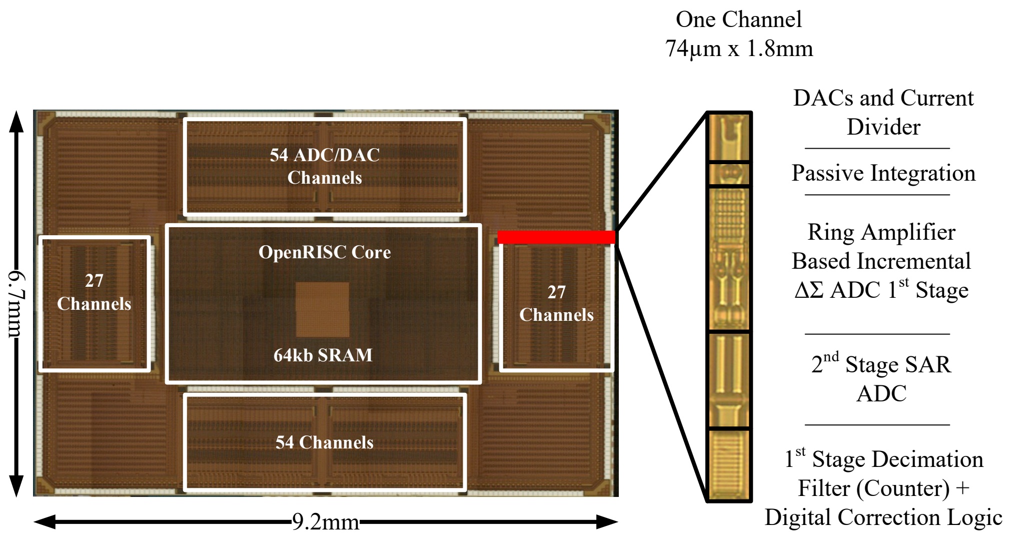


Figure 3-10. Die photograph of the 180-nm prototype.

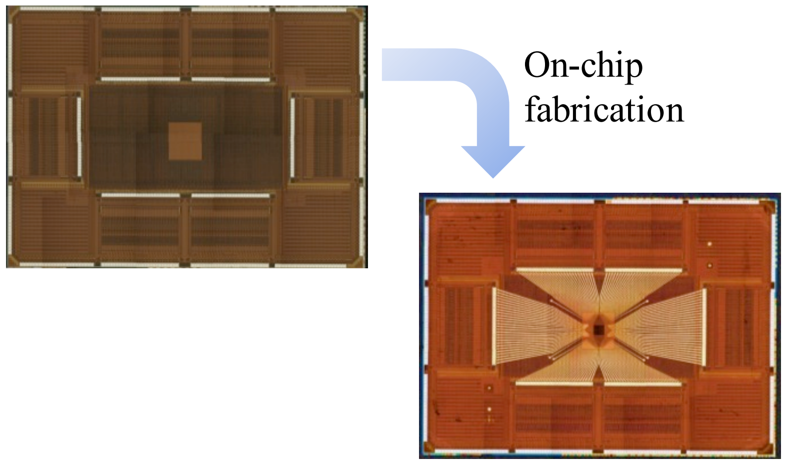


Figure 3-11. Die photograph with on-chip ReRAM crossbar.

3.6 Measurement Results

3.6.1 Measurement Setup

The prototype ReRAM coprocessor is wire bonded to a 391-pin PGA package for testing and measurement. A custom printed circuit board that provides analog and digital supplies, as well as the off-chip biasing for the mixed-signal peripheral hardware to the coprocessor, is shown in Fig. 3-13. The instructions for the coprocessor are written in standard C and compiled into binary machine code on a host PC using the or1knd toolchain [36]. The C program contains the high-level instructions for algorithm implementation as well as the low-level instructions supporting the on-chip hardware configuration settings.

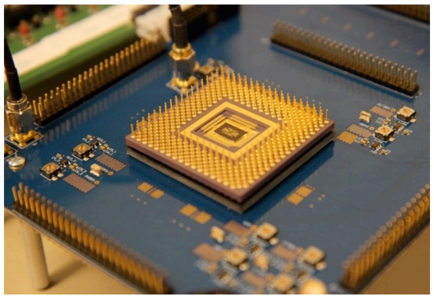


Figure 3-12. Pin grid array package on measurement printed circuit board.

Access to the coprocessor dual-port SRAM is provided by an on-chip “back-door” memory controller and the UART port of the OpenRISC. A custom C bootloader is compiled and loaded into the SRAM using the memory controller and Opal Kelly XEM7001 FPGA. The bootloader loads neuromorphic applications directly into the SRAM instruction memory address space from a PC through a USB-to-UART translator to the UART port peripheral of the OpenRISC processor.

3.6.2 Charge Domain Multiplication

The pulse-mode DAC and charge-domain ADC linearity are tested by replacing the ReRAM connection between DAC and ADC with a selection of discrete resistors and applying 6b input pulse trains. The integrated charge per step is the multiplication of the input value and the conductance – or one “op” in a VMM. Fig. 3-13 shows the integrated charge versus DAC input code for five different input currents. The integrated charge error is the difference between the expected charge and the charge measured by the ADC. Since the input is applied as RTZ pulses, any non-idealities or errors show up as a gain error, as shown in Fig. 3-13.

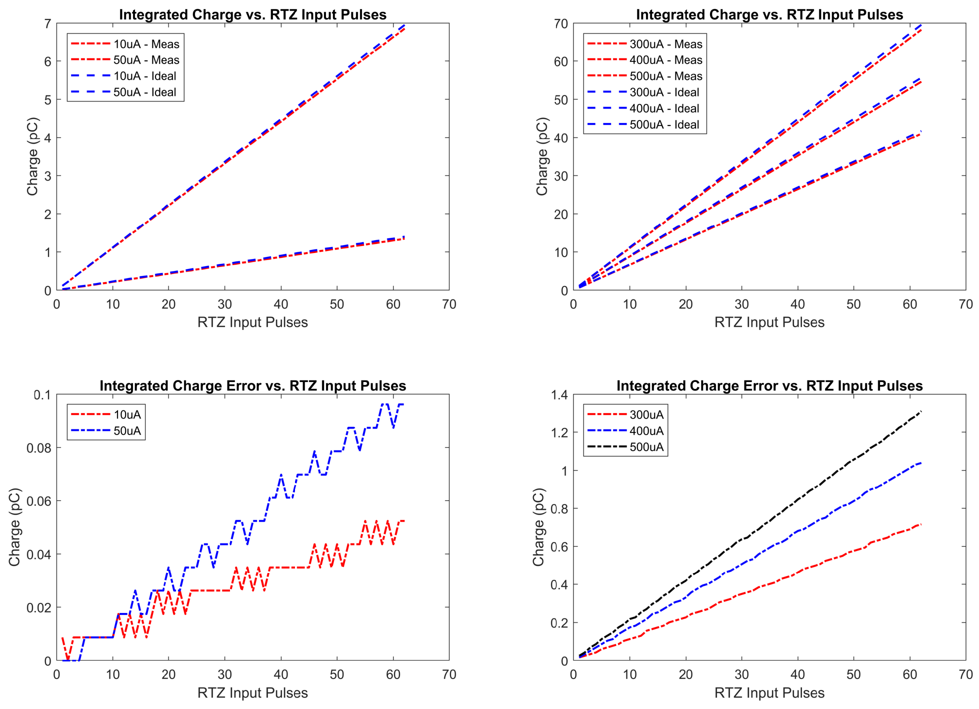


Figure 3-13. Integrated charge (top row) and error (bottom row) different input currents measured at 48 MHz system clock frequency.

3.6.3 System Performance

The system performance is determined from the power consumption and throughput of the entire system at the 148 MHz maximum operating frequency (Table 3-3). The mixed-signal core, including the ADCs and DACs supporting the 54x108 crossbar, consumes 64.4mW. The mixed-signal energy efficiency is 144 nJ/VMM or 25 pJ/Op. The OpenRISC core and passive crossbar array consume 235.3mW and 7mW, respectively. The total system power is 307mW and the maximum theoretical throughput is 2.6 GOPS at 8.5 GOPS/W.

Crossbar	ReRAM Crossbar	Passive 1R
	ReRAM Bitcell	WO_x
	Array Dimension	54x108
	Minimum Resistance (kOhm)	300
	Maximum Resistance (kOhm)	600
	Levels	16
	Variation	4.2%
	Power (mW)	7
Mixed-Signal Interface	# of ADCs	162
	ADC Resolution (bits)	13
	# of DACs	486
	DAC Resolution (bits)	6
	Power (mW)	64
CPU	Power (mW)	235
System Performance	System Clock (MHz)	148
	Input (bits)	6
	VMM/S	448 K
	OP/S	2.6 G
	OPS/W	8.5 G
Mixed-Signal Efficiency (6 bit inputs)	Energy/VMM (nJ)	144
	Energy/Op (pJ)	25

Table 3-3. System summary.

3.7 Conclusion

Analog ReRAM promises very efficient vector-matrix multiplication but requires extensive circuit support. This work presents mixed-signal circuitry for efficient and flexible analog VMM and describes the first fully integrated single-chip analog ReRAM system. We introduce specialized mixed-signal circuits to stimulate and read the ReRAM crossbar efficiently. Charge-domain operation with pulse-mode DACs ensures high linearity. Two-stage regulation provides a robust virtual ground and programmable attenuation. Hybrid two-stage ADCs integrate and digitize crossbar output current. A RISC processor interfaced to a memory-mapped mixed-signal core controls crossbar operation. The integrated, reconfigurable coprocessor implements unsupervised and supervised online learning, feature extraction, and classification in a multilayer network.

Chapter 4 An 8-bit 14.63 TOPS/W Multi-Macro Multi-Level-Cell ReRAM Compute-In-Memory RISC-V SoC[16]

4.1 Introduction

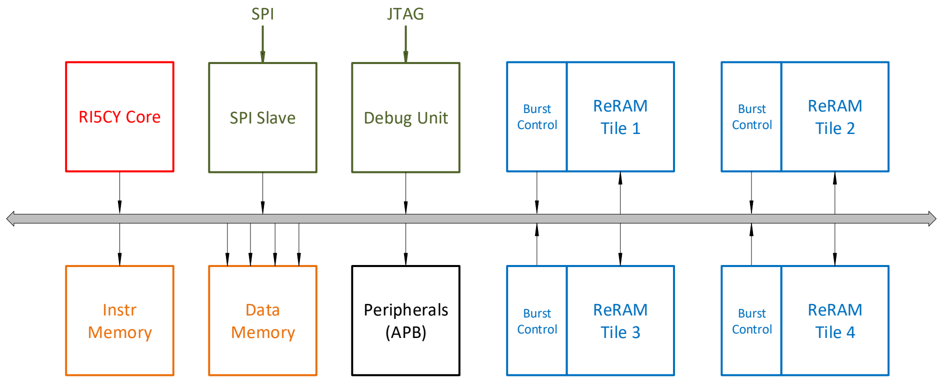


Figure 4-1. Diagram of PULPino RISC-V processor and AXI interconnect bus peripherals[17].

Despite significant progress [1-10,24,30,39,40], the mixed-signal and digital circuits remain power and area bottlenecks for aCIM. In addition, the design in the last chapter highlights several key problems. The design in Chapter 3 applies a 1b DAC input pulse train to solve the bit-cell linearity problem, but a fundamental drawback is that this causes the VMM period to scale exponentially with the input precision. This especially increases VMM latency for higher resolution inputs. Second, the post-processed ReRAM crossbar array (Fig 3-12) is not suitable for practical application. For maximum integration and compute density, the crossbar array should be

[16] This work was in collaboration with Jack Erhardt, Lu Jie, Seungjong Lee, Seung Hwan Lee, Seungheun Song, Wei Tang, Luke Wormald, and Junkang Zhu. The authors main contributions are: Jack Erhardt: firmware development; Justin Correll: mixed-signal controller and switch interface design, SOC verification, SOC bring-up, neural network demonstration; Lu Jie: ADC concept, design, and layout; Seungjong Lee: bit-serial concept and mixed-signal integration and layout; Seung Hwan Lee: System simulations; Seungheun Song: TIA concept, design and layout and 9-bit programming DAC concept, design, and layout; Wei Tang: RISC-V modification, verification, and implementation and top-level place and route ; Luke Wormald: PCB design and firmware development; Junkang Zhu: RISC-V modification, verification, and implementation and top-level place and route.
[17] Figure by Junkang Zhu

foundry integrated in the back of the line (BEOL). Further, the crossbar array circuitry should be pitch-matched to the crossbar array rows and columns. Third, a high-resolution 13-bit ADC is not necessary for inference and wastes energy in edge CIM applications. Instead, an 8-bit ADC is sufficient and will save energy thus increasing performance. Fourth, the passive 1R crossbar array structure suffers from sneak-path currents which cause errors both during programming and inference. Foundry integrated 1T1R ReRAM in the BEOL is a better option as a crossbar array processing element (PE). The transistor in the bitcell both addresses the sneak path problem and also provides current compliance during programming. Finally, the array size (54x108) is too small. Such an array size does not provide enough parameters to implement a practical DNN. For example, one of the simplest networks, LeNet1 contains ~3K parameters [29].

This work improves upon recent progress and the prototype from the previous chapter in the following areas: (1) We introduce ADC-assisted bit-serial processing for high throughput and energy efficiency; (2) A compact low-power binary-weighted 8-bit ADC enables one ADC per bit line pair; (3) A pitch-matched switch architecture seamlessly supports high and low voltage operations; (4) On-chip circuits support write-verify programming and accurate ReRAM characterization; (5) A RISC-V host processor controls device programming and MAC operations and exercises macro-level control by establishing data streaming between data memory and CIM tiles; and (6) A simple local controller offloads MAC operations from the host. We present a complete RISC-V SoC prototype comprised of 4 ReRAM-based CIM macros that support instruction-programmable end-to-end DNN inference (Fig. 4-1). For demonstration, we train and map the LeNet 1 DNN architecture onto ReRAM devices on a single chip and run inference. The SoC achieves a maximum efficiency of 14.63 TOPS/W. The classification accuracy for the MNIST dataset is 96.8%.

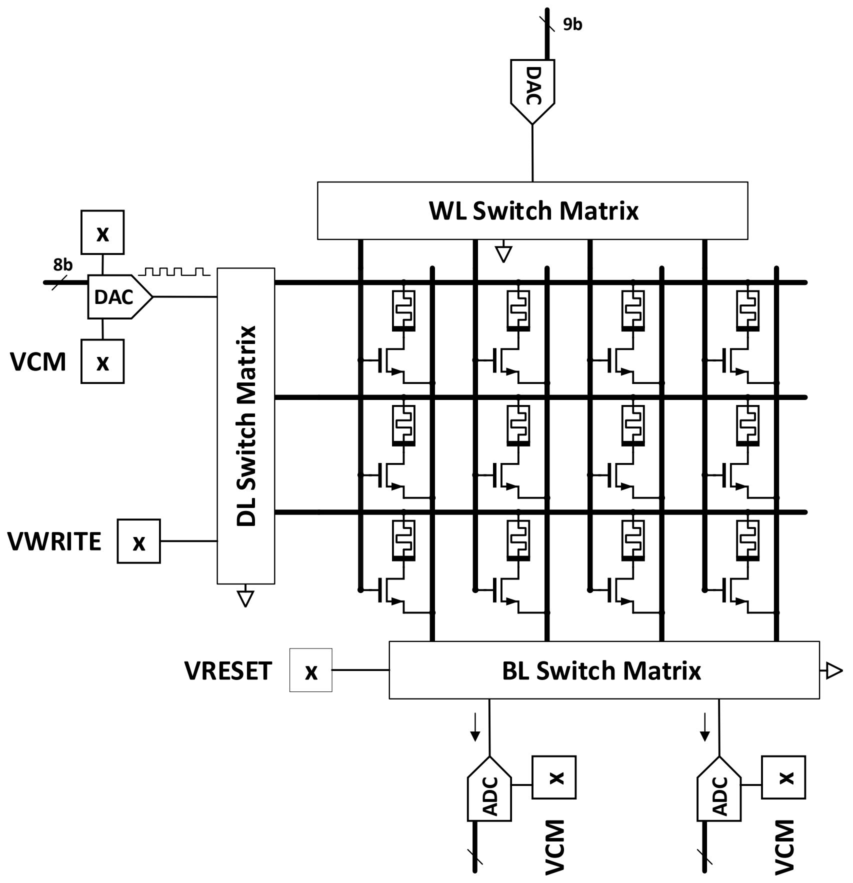


Figure 4-2. Diagram of mixed-signal hardware in the compute-in-memory macro. The DL switch matrix connects mixed-signal row circuits to the rows of the crossbar. The BL switch matrix connects the mixed-signal columns circuits to the columns of the crossbar. The WL switch matrix connects the bitcell selector (1T) to the programming DAC or 3.3V reference.

4.2 System Architecture

Fig. 4-2 shows one of the four self-contained CIM tiles in the prototype. The 256x64 crossbar array is comprised of 1T1R filamentary ReRAM devices. Connected through the bit-line switch matrix, 32 8-bit ADCs and TIAs support 64 crossbar array bit-lines. A 9-bit word-line DAC provides the gate voltage for the 1T1R bitcells. The word-line switch matrix connects the 9-bit DAC to even or odd columns (MAC) or just a single column (write or forming). 256 DACs apply the 8-bit input to the drivelines. The driveline, bit-line, and word-line switch matrices support MAC, form, set, reset, and single device measurement. Memory-mapped registers interface to the hose processor through fast DMA. A local controller runs MAC operations and alerts the host processor once complete.

Dedicated hardware for each row and column supports necessary array operations. Write hardware selects individual ReRAM bitcells for high-voltage, bi-directional potentiation as well as provides current compliance during ReRAM filament formation and fine-grained tuning during programming. Read hardware performs individual bitcell read-verify and parallel VMM at low-voltage. Specialized analog switch matrices enable high-voltage operation while protecting low-voltage hardware.

4.3 Design Considerations

A critical contribution of this work is ADC-assisted bitwise crossbar operation for high throughput, high accuracy, and low power. PWM or pulsed-DAC operation overcomes bitcell nonlinearity, but the required time grows exponentially with input resolution [30] causing slow throughput, especially for > 6-bit MAC inputs. Another challenge is that PWM operation involves integration in the bit-line or mixed-signal circuitry. In our approach, the drive-line DACs apply N pulses corresponding to the N bits of the input. A Transimpedance Amplifier (TIA) converts the bit-line output to voltage and drives the ADC input. A new binary-weighted multi-cycle sampling (BWMCS) ADC combines and weights the result from each bitwise operation before digitization.

Crossbar arrays of 1T1R bitcells require extensive support circuitry for practical application. For example, ReRAM programming voltages (~4V) exceed nominal digital and analog references, while MAC or VMM operates at nominal voltage levels. Array row and column terminations are shared for programming and VMM this must accommodate high and low voltage modes. In addition, the support circuitry must match the pitch of the crossbar array for maximum throughput and compute density. To address this, we present a minimal switching interface that achieves high voltages while protecting low voltage circuitry. The interface contains pitch-matched specialized switches, level-shifters, and address decoding to accommodate all operating modes of the crossbar.

The total MAC column current depends on DNN algorithm workload and individual bitcell conductance. The column termination must be capable of resolving large MAC currents and performing write-verify on single bitcells during weight storage. Rather than design for the worst-case maximum column current (maximum bitcell current * number of rows), we consider the workload for the MNIST handwritten digits [42] classification task on the LeNet1 architecture [29]. Given a bitcell specification of 300nA to 3μA, we determined a maximum column current of 45μA for inference. Finally, we chose 8-bit resolution as a good tradeoff for both weight storage accuracy and MAC output quantization.

4.4 Mixed-Signal Compute-In-Memory Macro

We now introduce the mixed-signal compute-in-memory (CIM) macro which includes drive line (DL), bit line (BL), and word line (WL) mixed-signal peripheral circuits as well as the mixed-signal digital interface (Fig. 4-3). The mixed-signal digital interface consists of memory-mapped registers and state machines to provide timing to drive ReRAM programming operations and VMM in the mixed-signal time domain. During operation, the processor sequentially loads configuration registers, then triggers mixed-signal timing. Pending mixed-signal operation, the processor waits (NOP) or is scheduled for compute on one of the 3 other CIM macros. All 4 of the macros are designed for independent operation. This section describes the hardware components of the CIM macro.

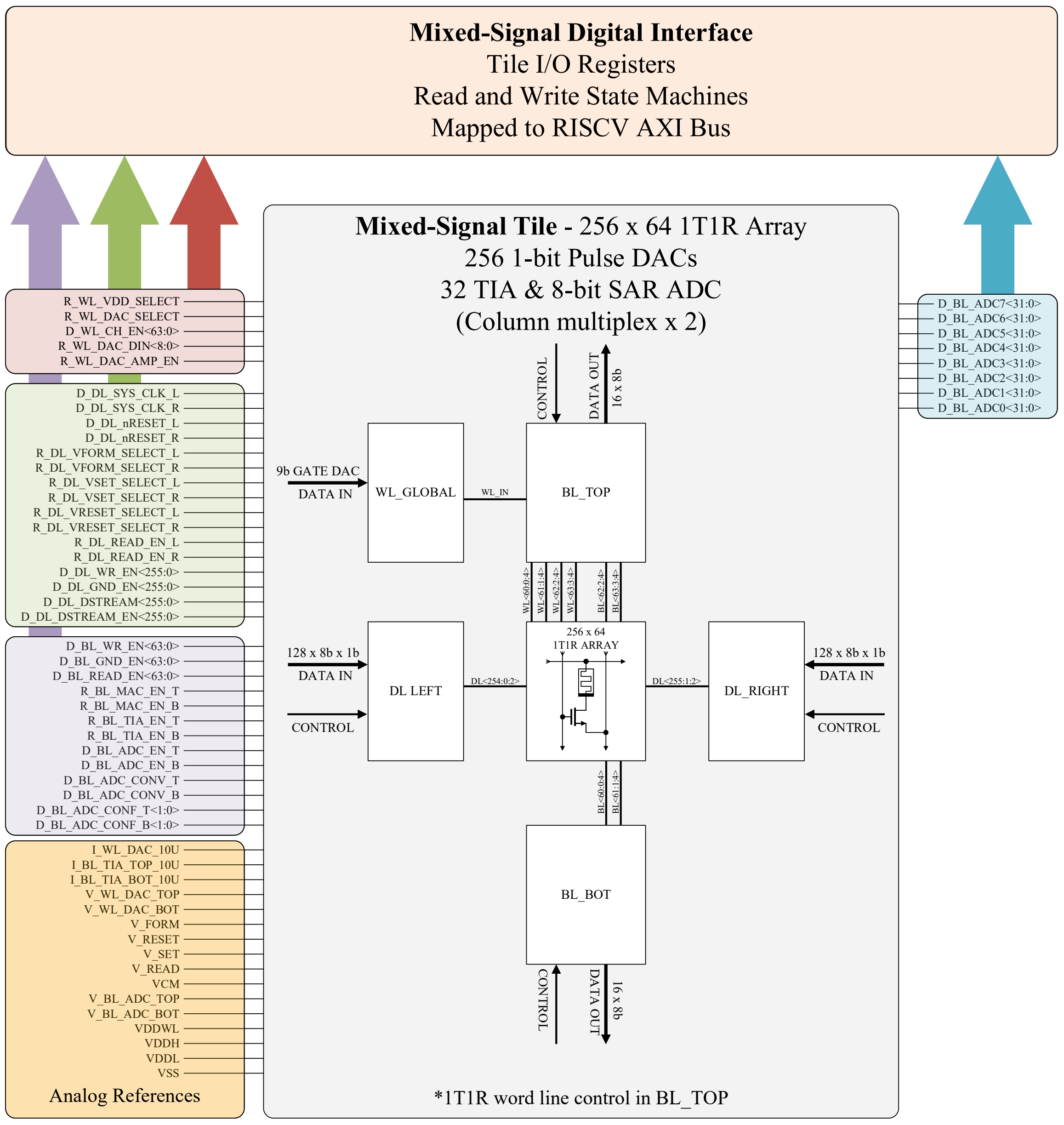


Figure 4-3. Diagram of full compute-in-memory macro including mixed-signal tile and digital interface.

4.4.1 Drive Line 1-bit DAC

Dedicated 1-bit drive line DACs deliver input stimulus to the rows of the crossbar for write-verify and MAC operations. In MAC mode, high-resolution MAC input is applied to the CIM array in parallel and with minimal latency. For each active row, an 8-bit digital input word is serialized into an 8-pulse bitstream which drives the DAC to output corresponding fixed-amplitude voltage pulses. For write-verify, the input is applied to only one active row.

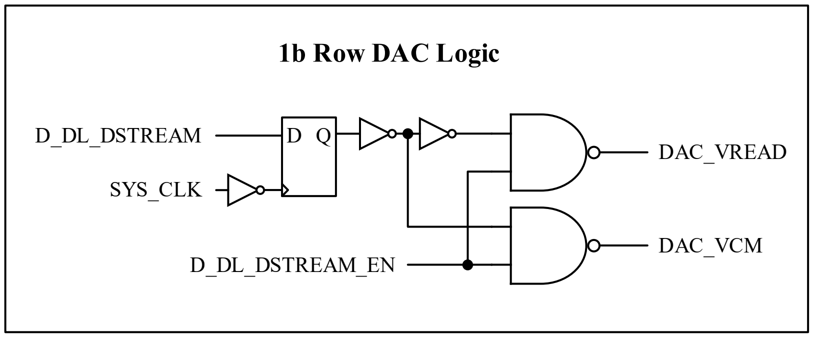


Figure 4-4. Drive line pulse DAC.

The logic of the drive line 1-bit pulse DAC (Fig. 4-5) drives two 1.2V PMOS switches between 900mV VCM and 1.2V VREAD. Since the source of the 1T1R bitcell is connected to the column VCM during read mode, the 1-bit pulse DAC toggles the row with the appropriate 300mV read voltage amplitude (VREAD-VCM). The pulse DAC input, D_DL_DSTREAM, is a 1-bit serialized bit stream generated in the mixed-signal digital interface block from an 8-bit input word. The DFF with negative CLK latches the serialized bitstream into the tile clock domain with a ¼ clock cycle delay.

4.4.2 Bit Line ADC[18],[19]

We introduce the binary-weighted multi-cycle sampling (BWMCS) ADC which efficiently weights the partial bitwise output and digitized the overall MAC result. Fig. 4-5 depicts the conceptual operations of BWMCS ADC. During the bitwise MAC operation, an adder sums the ADC input (i.e. from the TIA) with half of the existing voltage on the sampler. Next, the sampler updates by sampling the adder output at the positive edge of the sampler clock. This process repeats

[18] Seungjong Lee and Seng Hwan Lee introduce the bitwise ADC operation
[19] Lu Jie created the BWMCS ADC

for multiple sampler clocks, sampling and summing with binary weighting. Finally, the quantizer digitizes the sampler output:

$$D_{OUT} = 2^{-1}A_{IN}[n] + 2^{-2}A_{IN}[n-1] + 2^{-3}A_{IN}[n-2] + \cdots + 2^{-n}A_{IN}[1],$$

where D_{OUT} is the digital output and $A_{IN}[n]$ is the nth sampled analog input.

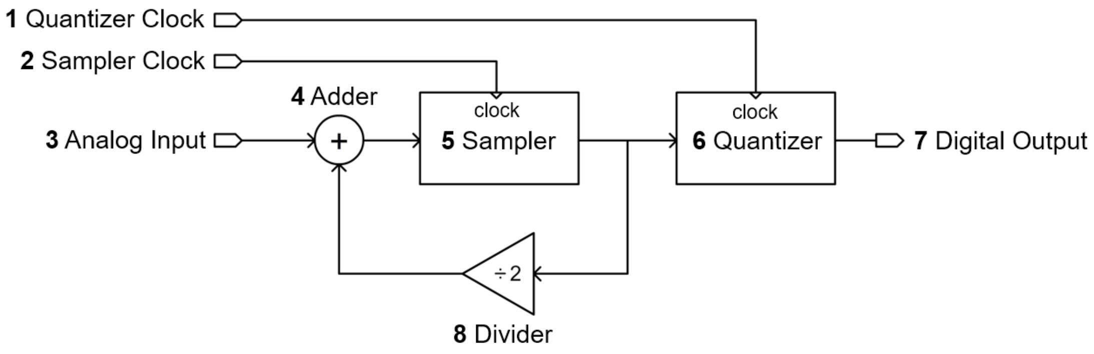


Figure 4-5. Multiplying ADC system level diagram[20].

Our very compact and highly efficient BWMCS ADC exploits the efficiency of the SAR ADC architecture. Unlike a SAR ADC, the BWMCS ADC has a separate sampling capacitor in addition to the binary-weighted CDAC array. The sampling capacitance has the same capacitance as the entire CDAC. After the sampling capacitor samples the input signal, charge sharing with the CDAC adds half the sampling capacitor voltage to half the CDAC voltage – this process implements the adder and divider in Fig. 4-6. The quantizer clock initiates successive approximation. The conversion switch connects the CDAC capacitors to the comparator input. The digital output updates and a reset switch clears the voltage on sampling and CDAC capacitors at the end of successive approximation. Asynchronous SAR operation eliminates the need for a special comparison clock. As a result, 8-bit SAR digitization completes within one system clock cycle. Table 4-1 shows the BWMCS ADC performance.

[20] Figure by Lu Jie

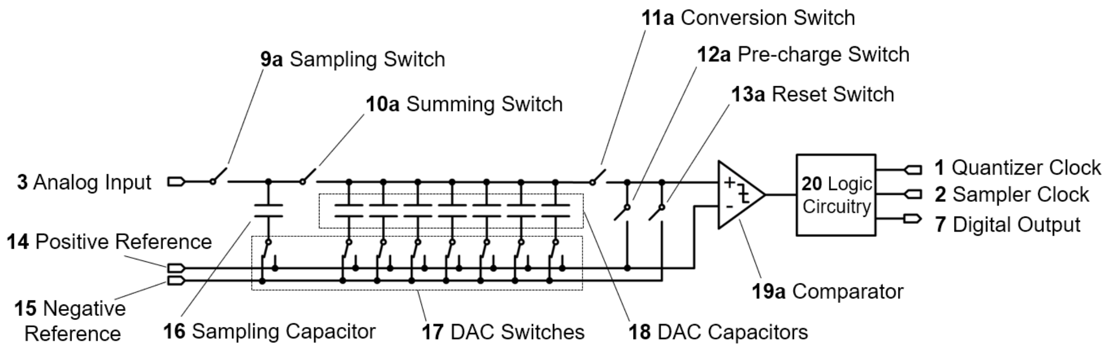


Figure 4-6. Multiplying ADC detailed diagram[21].

Specification	Value (Simulated)
Supply	1.2V
Sampling Rate	>40MHz (with 5Kohm R_S)
Conversion Delay	20 nS
Power	7.8 pJ/conversion (8 sampling + 1 conversion) or 35uW @ 40MHz
Area	7.68um x 210um
Resolution	8 bits
Input Range	0 ~ (V_{REF}+ - V_{REF}-)
Effective Number of Bit	7.5 bit (w/ noise, w/o mismatch)

Table 4-1. ADC performance specifications.

4.4.3 Bit Line TIA[22]

A TIA terminates the bit line with a low impedance and converts the bit line current to a voltage, which drives a bit line ADC. Our unique TIA design addresses: (1) compact size, (2) low power, (3) ADC drive, and (4) stability. For compact size and low power, we choose a gm-boosted common-gate amplifier that drives a grounded resistor load (Fig. 4-7). The output voltage directly drives the ADC without buffering. A challenge is that the resistor load of a common gate amplifier

[21] Figure by Lu Jie
[22] Seungheun Song designed and implemented the TIA

cannot simultaneously provide sufficient gain and bandwidth for the kT/C noise-limited ADC. We break this gain-bandwidth tradeoff with a current amplifier (MP1B) which provides TIA gain without increasing the output time constant. A compensated two-stage amplifier provides high gain and introduces a dominant pole. A bleed current (MP2) ensures sufficient MP1A current so that the TIA input pole remains non-dominant in the case of low bit line current. A high gain mode with an additional series load resistance (R4) supports write-verify of a single bitcell during programming. The entire TIA, including load consumes 30μW.

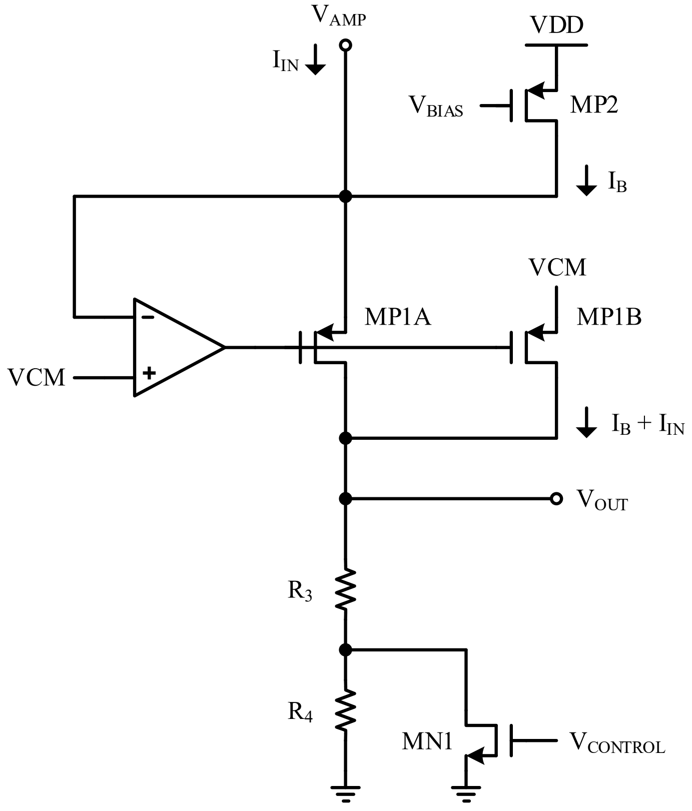


Figure 4-7. Column termination trans-impedance amplifier[23].

[23] Figure by Seungheun Song

4.4.4 Word Line Programming DAC[24]

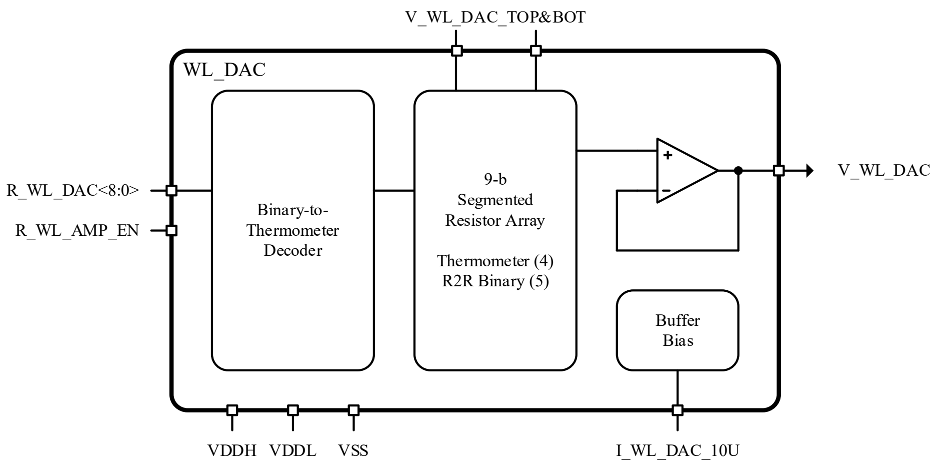


Figure 4-8. Word line 9-bit DAC[25].

The word line programming DAC provides precise gate control during device programming. The gate DAC consists of a resistor-based segmented DAC and output buffer (Fig. 4-8). To subvert non-linearity, the lower 4-bits are implemented with an R2R DAC configuration and the upper 5-bits are implemented with a thermometer-coded DAC architecture. The output voltage is buffered to effectively transmit the voltage to the word line. The output buffer is a class-AB amplifier with a unity-gain configuration and requires a 10 μA bias current. The reference voltage of the 9-bit gate DAC is from 0.5V to 2.9V and the LSB size is 4.687 mV. For low power operation, after device programming, gate DAC can be switched off to save power.

4.4.5 Crossbar Switch Architecture

For maximum integration, the support mixed-signal circuitry should match the pitch of the crossbar array. This pitch requirement causes particular challenges for switch matrices. A specific challenge is that the switch matrices should support low voltages for read mode and high voltage

[24] Seungheun Song designed and implemented the word-line programming DAC
[25] Figure by Seungheun Song

for programming and filament formation. One aspect of this challenge is that high voltage transistors are much larger than standard transistors. Moreover, high voltage transistors may require individual level shifters that are even larger. Our approach limits the number of high voltage transistors and critically minimizes the number of level shifters.

4.4.6 Drive Line

The drive line circuitry interfaces to the rows of the crossbar array. Each row of the array is physically connected to the top electrode of each ReRAM bitcell in that row. The drive line circuitry consists of 256 slices and a global write voltage select block shown in the figure below. Each slice contains a 1-bit pulse DAC and read, write, and ground switches. The global write voltage select block consists of 3 one-hot switches to select the appropriate programming voltage.

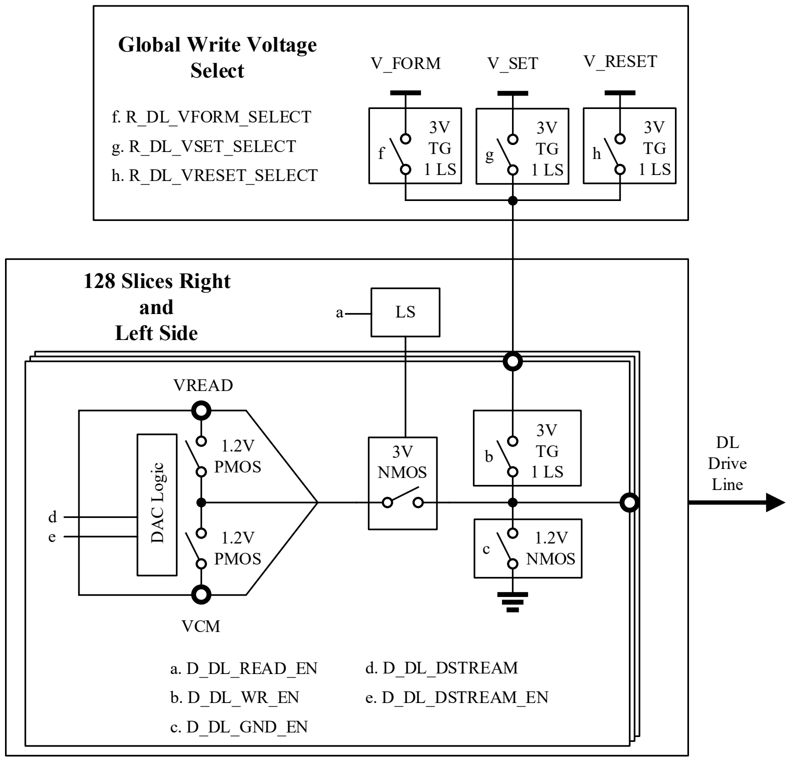


Figure 4-9. Drive line hardware.

The drive line supports 3 modes of operation: read, write, and idle. The modes are determined by the switches enabled by signals “a”, “b”, and “c” shown in Fig. 4-9. The read enable switch is a 3V I/O NMOS transistor that connects the row DAC to the crossbar during read and protects the DAC during programming when the DL voltage may exceed VDDL = 1.2V. All 256 read enable switches activate at once under the control of a global level shifter. During read, switch “a” is selected and switches “b” and “c” are off or disabled. The write enable switch connects the global write voltage select block to the crossbar during read through a 3V I/O transmission gate. The level shifter is local to each slice. The actual write voltage is further selected by the 3 one-hot switches (“f”, “g”, “h”) in the write voltage select block. During write, switch “b” is selected and switches “a” and “c” are disabled. The idle state of the drive line is 0V or GND. To enable idle mode, switch “c” is on and switches “a” and “b” are off or disabled.

4.4.7 Bit Line

The bit line circuitry interfaces to the columns of the crossbar array (Fig. 4-10). Each column of the array is physically connected to the source of each transistor in the 1T1R ReRAM bitcells of that column. The columns of the array are multiplexed by two, meaning two columns share one TIA and ADC. The bit line circuitry is split between the top and bottom of the array in two sections where each section consists of 16 slices for a total of 32 slices (64/2). The top slices include the word line circuitry which is discussed in the next section. The bit line slice consists of a TIA, ADC, read, write, and idle switches.

The bit line supports 3 modes of operation: read, write, and idle. These modes are determined by the switches enabled by signals “a” – “f” shown in Fig. 4-10. Further due to the multiplexed nature, the bit line operates in an even/odd fashion. We describe the operations from the “even” perspective, but this description easily extends to the “odd” case.

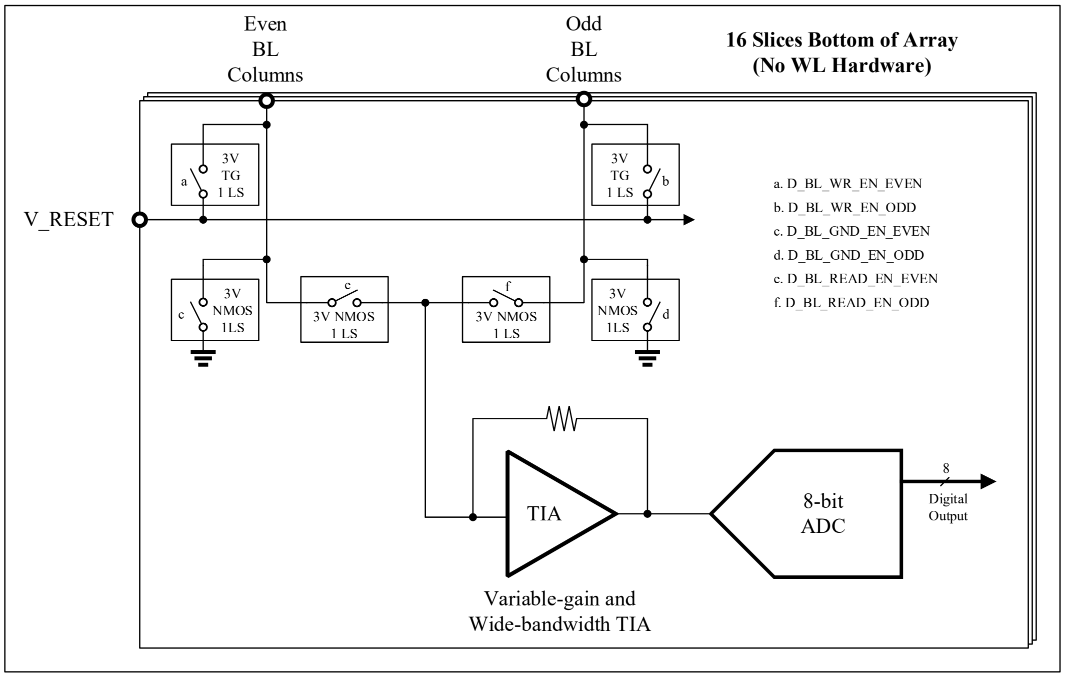


Figure 4-10. Bit line hardware.

The read enable switch is a 3V I/O NMOS with a local level shifter that connects the TIA and ADC to the crossbar rows during read and protects the TIA during write when the bit line voltage may exceed VDDL = 1.2V. During read (even), switch “e” is enabled, and all other switches are off or disabled. The write enable switch is a 3V I/O transmission gate with a local level-shifter and connects the crossbar column to the reset voltage V_RESET during the reset write operation. During write (even), switch “a” is enabled and all other switches are disabled. The idle state of the bit line is 0V or GND. To enable idle mode, switch “c” is enabled, and all other switches are disabled.

4.4.8 Word Line

The word line circuitry interfaces to the gates of the 3V I/O transistors in the 1T1R bitcells of the crossbar array (Fig. 4-11). The “selector” transistor turns on fully during read and provides current compliance during write. The word line consists of 16 slices, a global word line multiplexor, and a global 9-bit resistor-based segmented DAC. Each slice has 4 channel enable switches and is located at the top of the array in the bit line slices. The DAC provides fine-grained potentiation steps to program the ReRAM bit cells.

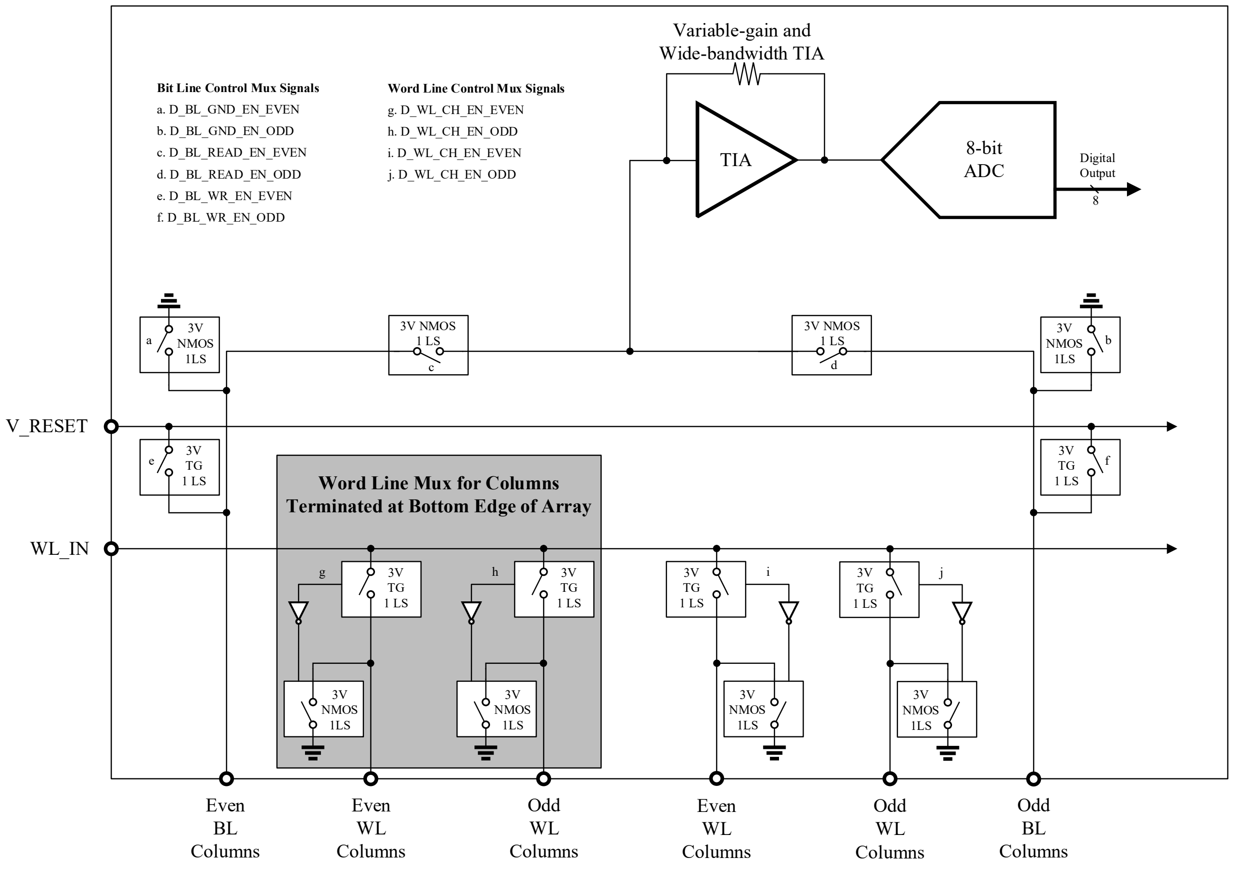


Figure 4-11. Word line hardware.

The word line supports 3 modes of operation: read, write, and idle. The modes are determined by the switches enabled by signals “g” – “j” shown above and by switches “a” and “b” in the word line mux. The word line mux (Fig. 4-12) connects the channel enable switches (1T gates) to either the 9-bit gate DAC or to a 3.3V reference through switches “a” and “b”, a 3V I/O transmission gate and 3V I/O PMOS transistor, respectively. In each channel, signals “g” - “j” drive a 3V I/O transmission gate and 3V I/O NMOS that connects the channel to the word line mux (WL_IN) or to ground. The switches, shown in the grey inset, operate in complementary fashion - while one is on the other is off. This ensures that the gates of the bitcells are never floating. For read mode, in the word line mux, switch “b” is enabled, and switch “a” is disabled. Similarly, for write mode, in the word line mux, switch “a” is enabled, and switch “b” is disabled. The corresponding channels are enabled through the word line control register. The channels are inherently placed in idle mode when not selected.

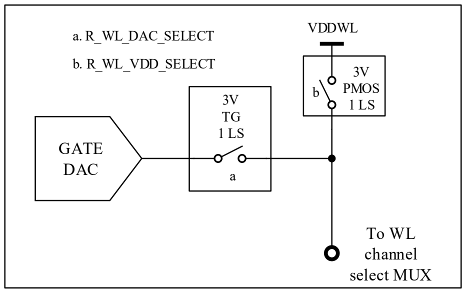


Figure 4-12. Word line multiplexor.

4.5 Digital Architecture[26]

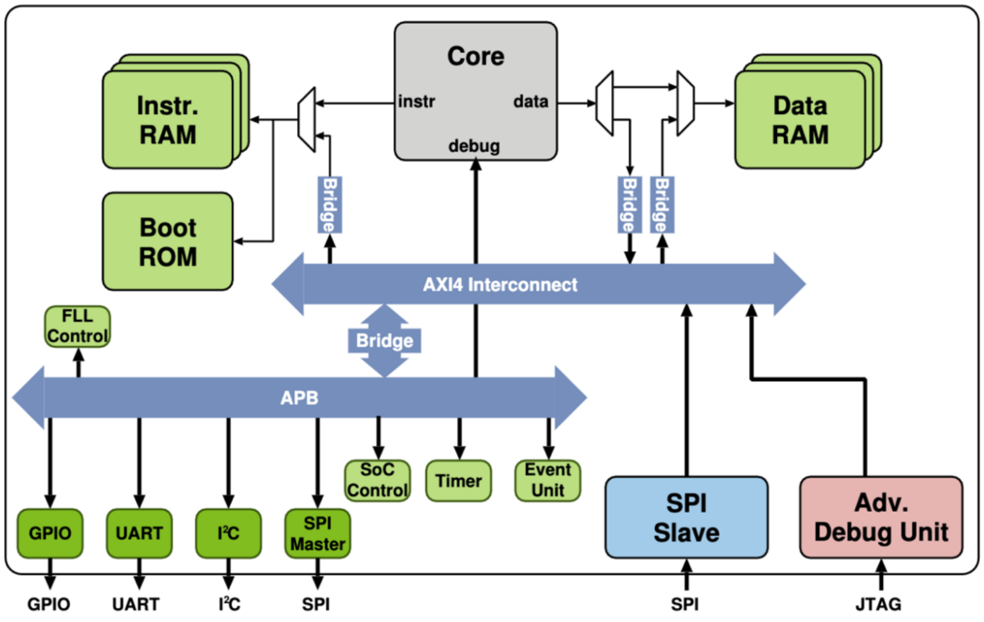


Figure 4-13. PULPino RISC-V processor.

[26] The digital architecture was developed and implemented by Junkang Zhu and Wei Tang

4.5.1 PULPino RISC-V Customization

A custom PULPino RISC-V processor (Fig. 4-13) [41] with 32KB instruction memory and 1MB data memory supports and controls the mixed-signal CIM macros. The RISC-V core acts as a host to provide control and coordination (Fig. 4-1). The 4 CIM tiles are attached to AXI4 interconnect to provide MAC acceleration. SPI and JTAG serve as the debug interface. Each CIM tile is wrapped by a register interface. To enable device characterization and initial system bring-up testing and debugging, the host can be programmed to exercise fine-grained control of form/set/reset/read/MAC read operations to the CIM macros by read/write access to the register interface. To support high-performance DNN inference, the host can be programmed to exercise macro-level control by establishing data streaming between data memory and CIM macros, and between CIM macros, in a DMA-like mode without constant interventions of the host controller to reduce the overhead. In this mode, a simple local controller on each CIM macro paces the sequence of events to run inference, including applying input activations, address enable, MAC read, and analog-to-digital conversion, as well as buffering output activations. The entire sequence can occupy 10's or 100's of cycles at a time. The host relies on the auto control of individual tiles and infrequent polling of the status registers to coordinate the macro-level events. Fig. 4-14 shows a diagram of the burst controller

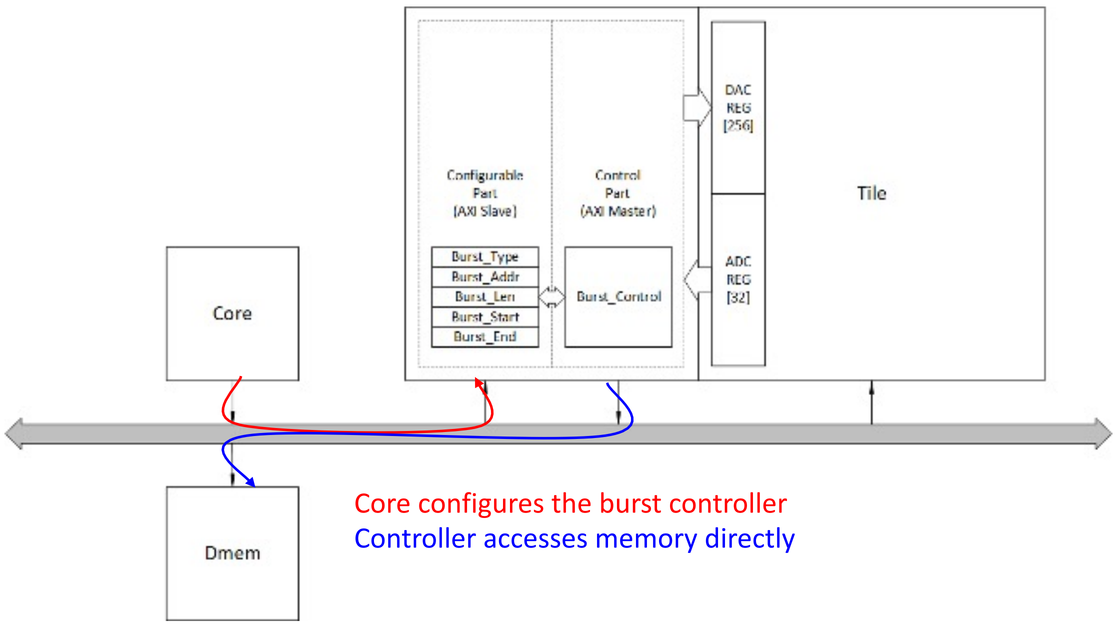

Configurable Part (AXI Slave)
Control Part (AXI Master)
Burst_Type
Burst_Addr
Burst_Len
Burst_Start
Burst_End
Burst_Control
DAC REG [256]
ADC REG [32]
Tile
Core
Dmem
Core configures the burst controller
Controller accesses memory directly

Figure 4-14. Diagram of DMA and burst controller[27].

4.5.2 Mixed-Signal Controller

To ensure the highest throughput, a custom timing controller synchronizes the analog compute independently of the host processor (Fig. 4-15). The controller provides control signals and clock gating enable signals to configure crossbar hardware switches and to drive the DACs and ADCs during device programming and MAC operation. The timing controller drives parallel operation by clocking out serialized data to the 1-bit DACs from the burst controller while simultaneously operating the parallel samplers of bit line ADCs. Fig. 4-16 shows an example timing diagram for an 8-bit bit-serial MAC operation.

[27] Figure by Junkang Zhu

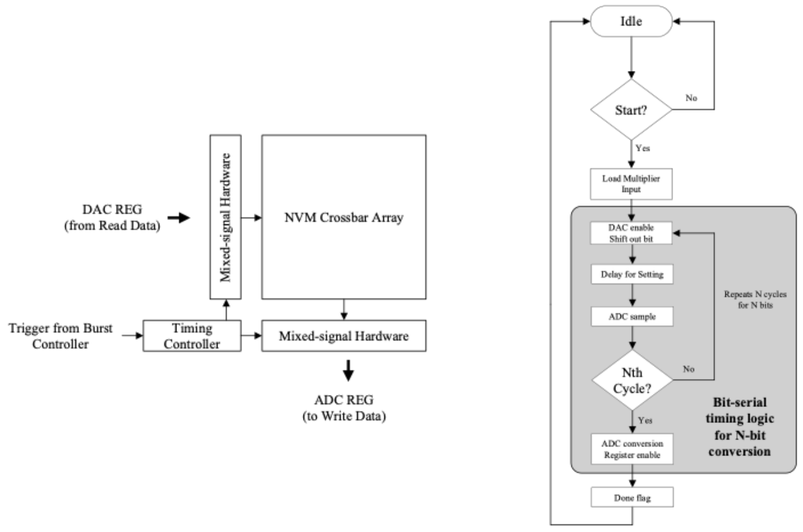


Figure 4-15. Local timing controller for bit-serial operation.

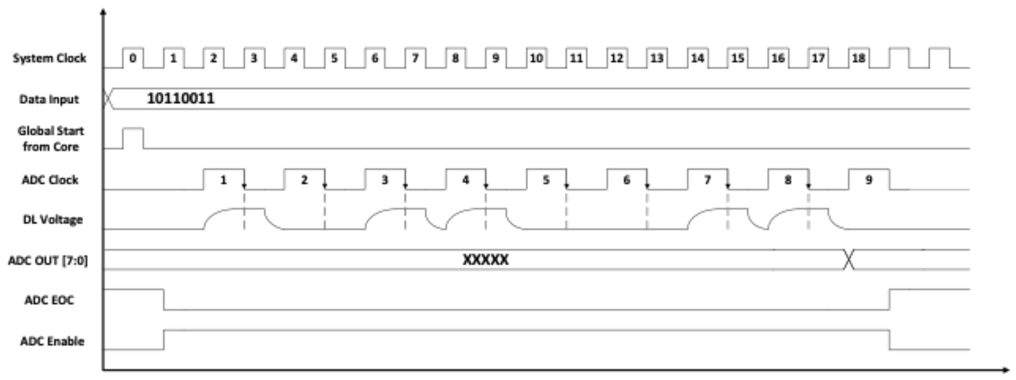


Figure 4-16. Timing diagram of bit-serial operation.

4.5.3 Mixed-Signal Digital Interface

The mixed-signal digital interface is “glue logic” that drives the mixed-signal macro by way of memory-mapped registers to the RISCV core. The timing of the mixed-signal macro is provided by separate read and write state machines. Control and address decoding are provided by memory-mapped registers. The configuration of the mixed-signal macro is set by the programmer by manipulating the memory-mapped registers. Table 4-2 shows an example memory map for a mixed-signal tile.

Address Name	Width		Address
R_SM_WRITE_CYCLES	[31:0]		0x1000_0000
R_DL_ADDRESS	[31:0]		0x1000_0004
R_DL_CONTROL	[31:0]		0x1000_0008
R_WL_DAC_DATA	[31:0]		0x1000_000C
R_WL_ADDRESS	[31:0]		0x1000_0010
R_WL_CONTROL	[31:0]		0x1000_0014
R_BL_ADDRESS	[31:0]		0x1000_0018
R_BL_CONTROL	[31:0]		0x1000_001C
R_ANALOG_CONFIG	[31:0]		0x1000_0020
R_WRITE_CONFIG	[31:0]		0x1000_0024
D_SM_START_READ	pulse	wo	0x1000_0060
D_SM_START_WRITE	pulse	wo	0x1000_0064
D_SM_WRITE_DONE	[0:0]	ro	0x1000_0068
D_SM_EOC	[0:0]	ro	0x1000_006C
ADC_DATA[7:0]	[31:0]	ro	0x1000_00C0 - 0x1000_00DC
READ_DAC_INPUT[64]	[31:0]		0x1000_0100 - 0x1000_01FC

Table 4-2. Mixed-signal digital interface memory-mapped registers.

4.6 Operating Modes

This SoC supports six different operating modes for efficient crossbar operation. Four modes cover ReRAM bitcell programming: filament formation (FORM), positive write (SET), negative write (RESET), and read-verify. The two remaining even and odd VMM modes compute partial sums. The RISC host processor directs bitcell programming algorithms, VMM partial sum summation, applies activation functions, and controls the data path. All programming operations require high-voltage crossbar operation to alter ReRAM conductance. Read-verify and VMM operations are computed at low voltage. Operating mode settings are stored in local macro configuration registers.

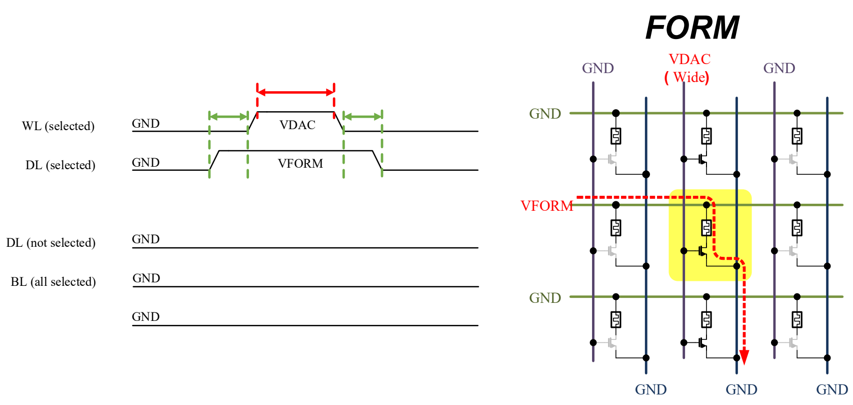


Figure 4-17. Timing and array operation diagram for FORM mode[28].

Programming filamentary ReRAM occurs in two phases. First, a filament is formed in the virgin ReRAM device, then the conductance is tuned to the appropriate value. Filament formation is an abrupt process that can lead to excessive device current catastrophically shorting the device. Further, filament geometry plays a role in ReRAM dynamics such as stochasticity and retention. The 1T1R selector transistor provides precise current compliance to control filament geometry. The selector transistor gate voltage is driven on the WL by a 9-bit segmented resistor array DAC. After the forming process, the bitcell conductance is tuned to the appropriate value under the control of a closed-loop double-sided programming algorithm. Negative RESET pulse trains (long term depression or LTD) decrease bitcell conductance and positive SET pulse trains (long term potentiation or LTP) increase bitcell conductance. The SET and RESET voltage references are supplied by off-chip sources. After any pulse, negative or positive, cell conductance is read-verified to check whether the RESET or SET state has met the algorithm threshold.

In all programming modes (FORM, SET/RESET, read-verify), the RISC processor loads macro registers to configure the DL, BL, and WL switch matrices to the appropriate row and column addresses. In FORM and SET modes, the bitcell is decoded such that current flows through

[28] Figure by Seungjong Lee

the selected 1T1R device from DL to grounded BL. Fig. 4-17 shows an example of FORM mode. In RESET mode, the bitcell is decoded in the opposite fashion such that current flows from BL to grounded DL. The WL of the selected cell is connected to the 9-bit DAC. Unselected WL columns are grounded, switching off unselected bitcells. Since all bitcells in a column are connected to the same WL, unselected DLs and BLs are tri-stated, or high impedance, to inhibit erroneous current flow. The 1T1R selector transistor offers isolation and inhibits sneak path currents in neighboring bitcells. Fig. 4-18 shows an example of RESET mode.

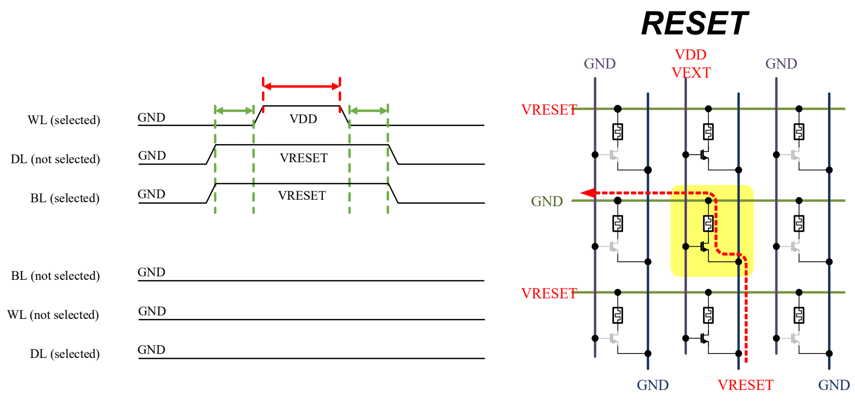


Figure 4-18. Timing and array operation diagram for RESET mode[29].

During read operations, the array operates in two phases to compute VMM partial sums utilizing positive and negative matrix coefficients, respectively. In the first phase, row read DACs are loaded with input data and connected to the crossbar. Even columns are connected to 32 TIA inputs, each providing a 900mV common-mode virtual ground to the column. Read DACs serialize the 8bit input activation vector into 900mV-1.2V RTZ pulse trains. Crossbar rows are excited with 300mV pulses giving rise to bitcell currents that sum along the columns according to physics. Column TIAs convert the MAC currents to voltages for digitization by 8-bit ADCs resulting in the VMM partial sum output activation vector. Fig. 4-19 shows an example of the even phase. In the

[29] Figure by Seungjong Lee

second phase, the process repeats on the odd columns utilizing the same input activation vector. Both ADC output activations for VMM partial sums are subtracted then stored by the processor.

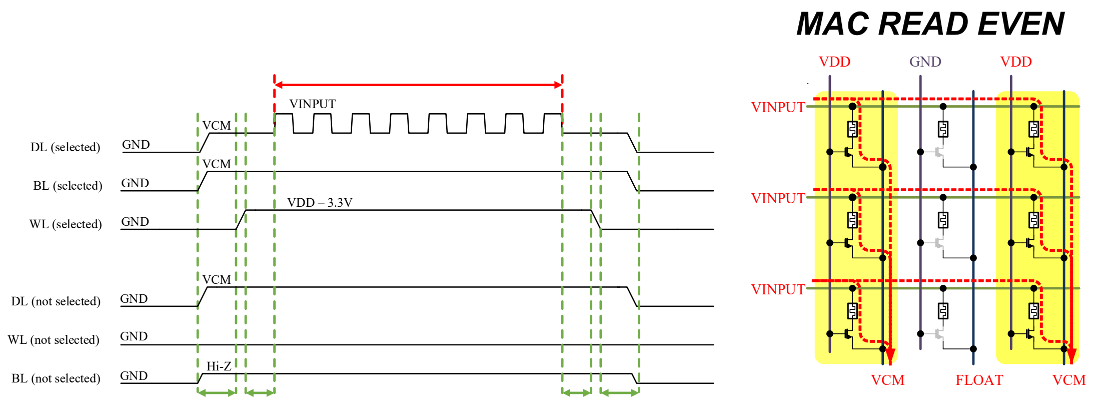


Figure 4-19. Timing and array operation diagram for MAC READ EVEN mode[30].

4.7 Firmware Stack[31]

The firmware stack provides the user with high-level functions to configure and operate the CIM macros through multiple layers of abstraction. Fig. 4-20 shows the layers of the stack. The lowest layer of the stack is the CIM macro. Macro configuration registers, data input (DACs), and data output (ADCs) are memory mapped to the RISC-V AXI interconnect bus. The host processor writes to CIM macro memory addresses to configure the macro and to provide VMM input vectors. Similarly, the host processor reads from CIM macro memory addresses to check CIM macro register settings and to read out VMM results.

The physical layer manipulates the registers of the CIM macros via read/write memory access in board.c. As described in the previous section, each operating mode (FORM, SET, RESET, READ) is executed as a series of register manipulations. First, the crossbar switch

[30] Figure by Seungjong Lee
[31] Luke Wormald and Jack Erhard participated in firmware development

matrices are configured for the specific mode, then the state machine is triggered, and finally, the crossbar switch matrices are configured for idle. The physical layer contains functions that encapsulate the low-level register manipulations for each operating mode. The set of functions in the physical layer are provided for the hard layer.

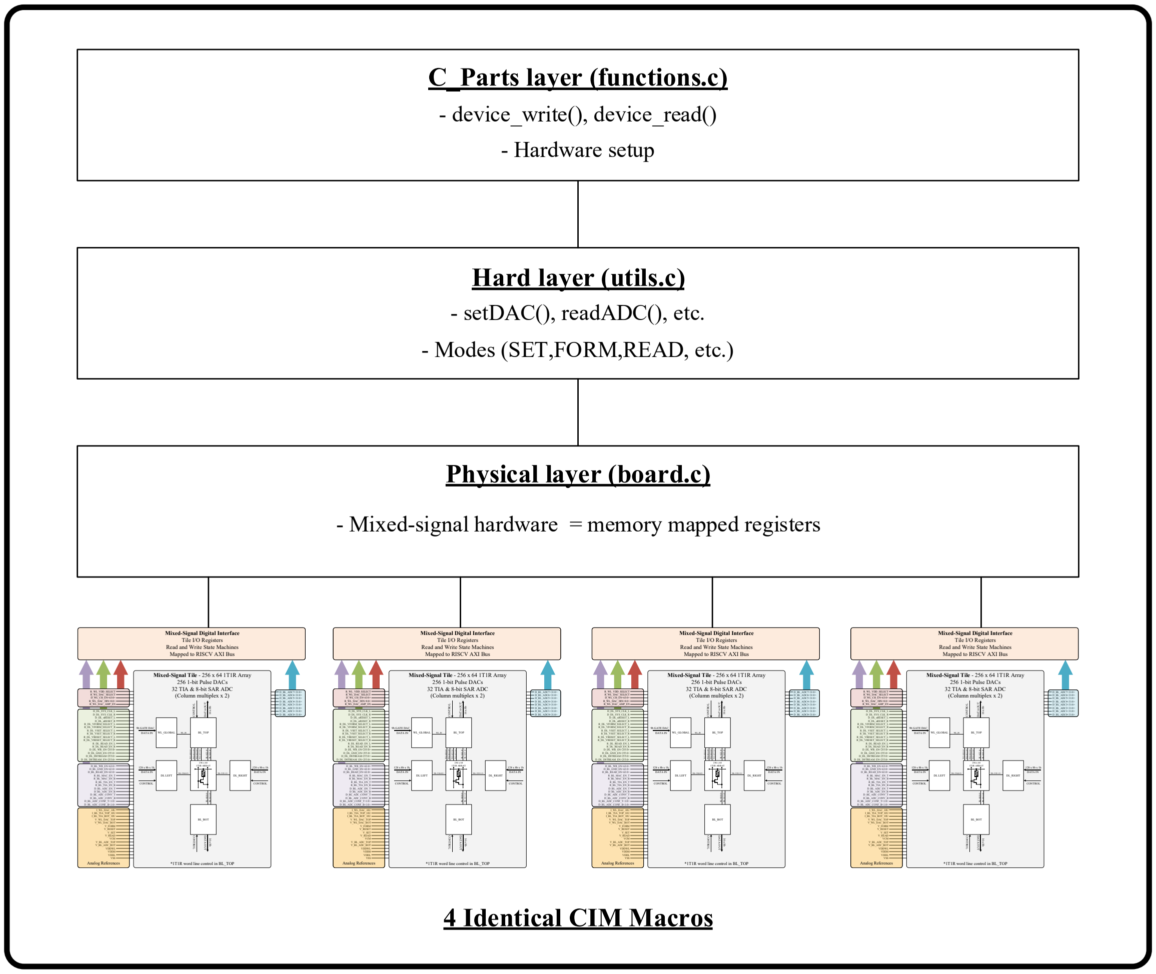

C_Parts layer (functions.c)
- device_write(), device_read()
- Hardware setup
Hard layer (utils.c)
- setDAC(), readADC(), etc.
- Modes (SET,FORM,READ, etc.)
Physical layer (board.c)
- Mixed-signal hardware = memory mapped registers
Mixed-Signal Digital Interface
Tile I/O Registers
Read and Write State Machines
Mapped to RISCV AXI Bus
Mixed-Signal Tile - 256 x 64 1T1R Array
256 1-bit Pulse DACs
32 TIA & 8-bit SAR ADC
(Column multiplex x 2)
4 Identical CIM Macros

Figure 4-20. Firmware interface to memory mapped CIM macros.

The hard layer organizes moderate-level CIM macro hardware utility in utils.c. It provides access to CIM hardware through the physical layer. The hard layer provides task functions such as loading DACs and reading ADCs as well as configuration settings specific to each mode. The set of functions in the hard layer are provided for the C Parts layer.

The C Parts layer is the highest level of abstraction to the hardware. Here, the functions are designed to encapsulate to complete functionality of each of the operation modes described in the previous section. Main applications for bitcell programming and neural network inference are written on top of the C parts layer.

4.8 Prototype

The prototype device is fabricated in 65nm CMOS and occupies 16.54 mm^2 (Fig. 4-21). This includes 4 CIM macros, the RISC-V PULPino processor, 32KB instruction memory, and 1MB data memory as shown in Fig. 4-22. A single CIM macro comprised of a 256x64 ReRAM crossbar array, 256 DACs, 32 TIAs, 32 ADCs, and crossbar periphery multiplexors occupies 0.354 mm^2 (Fig. 4-23).

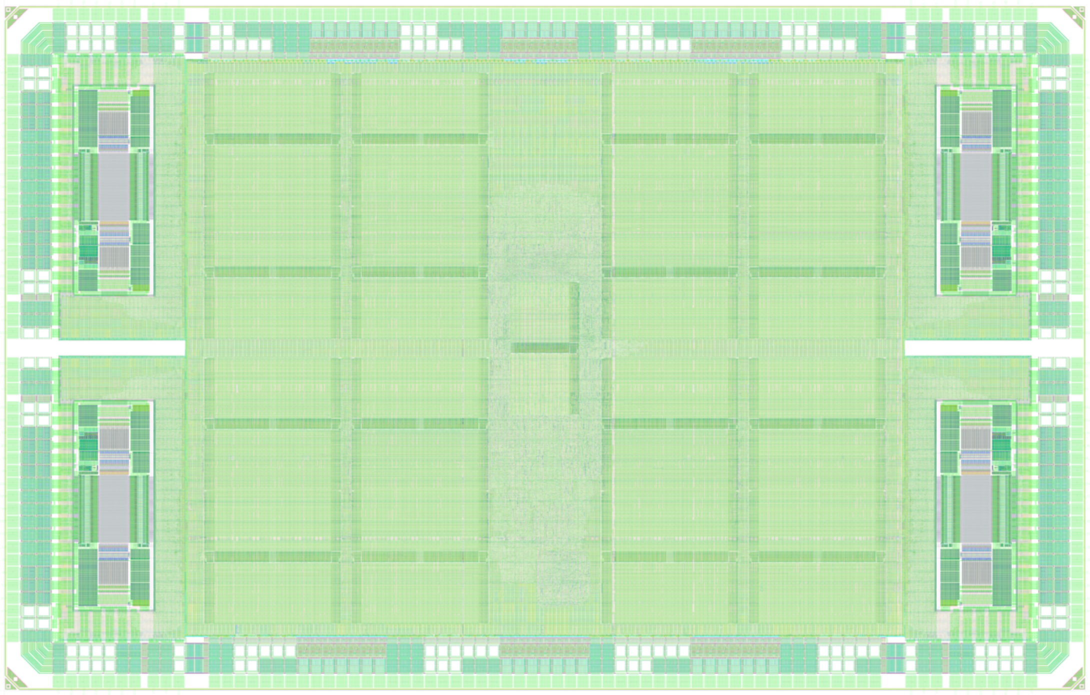


Figure 4-21. Prototype IC layout.

Fig. 4-23 shows a detailed diagram of the pitch-match CIM macro floor plan. Row and column mixed-signal circuitry is channelized to match the row and column dimensions of the crossbar array. On the left side of the crossbar, even rows are terminated with 128 channels each containing a DL mux and 1-bit DAC. The same structure is mirrored to the right side of the crossbar to the odd rows. The top of the crossbar array contains the WL multiplexors for 1T1R transistor gate control, BL multiplexors, TIAs, and ADCs. Since each BL hardware channel shares two crossbar columns, every other pair of columns is terminated with 16 channels each containing WL mux, BL mux, TIA, and ADC. The same structure, excluding the WL mux, is mirrored to the bottom of the crossbar.

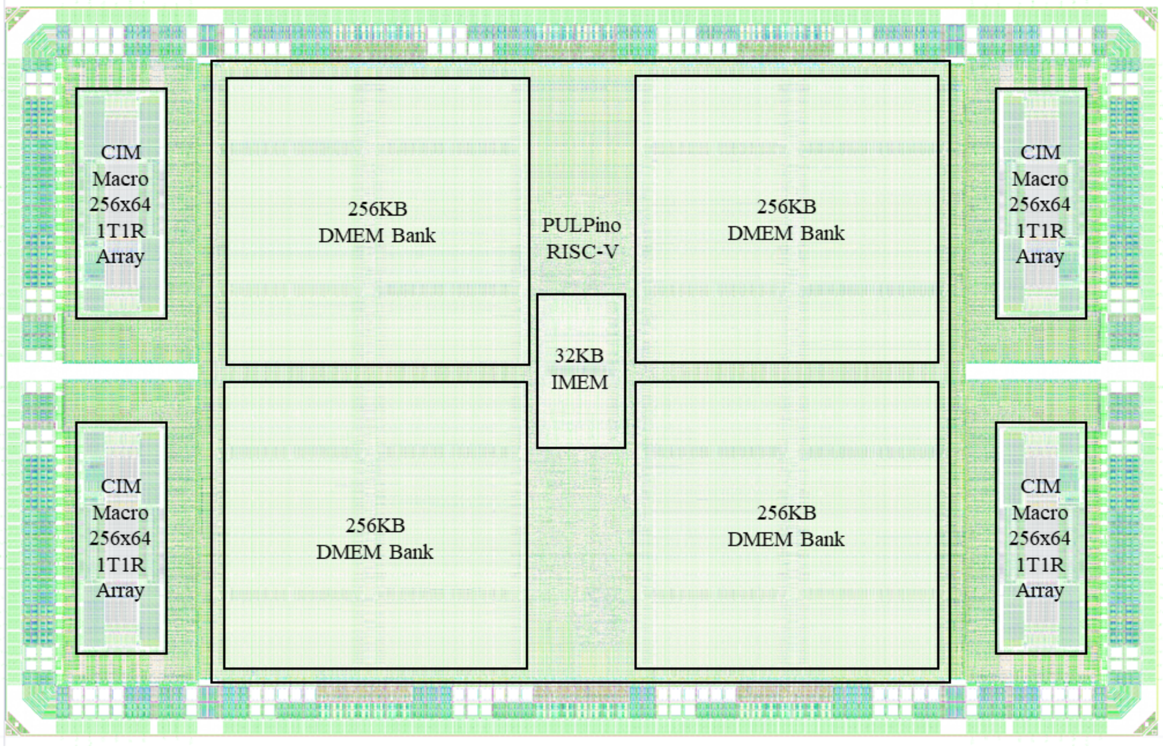

CIM Macro 256x64 1T1R Array
256KB DMEM Bank
PULPino RISC-V
256KB DMEM Bank
CIM Macro 256x64 1T1R Array
32KB IMEM
CIM Macro 256x64 1T1R Array
256KB DMEM Bank
256KB DMEM Bank
CIM Macro 256x64 1T1R Array

Figure 4-22. Prototype IC SOC floor plan.

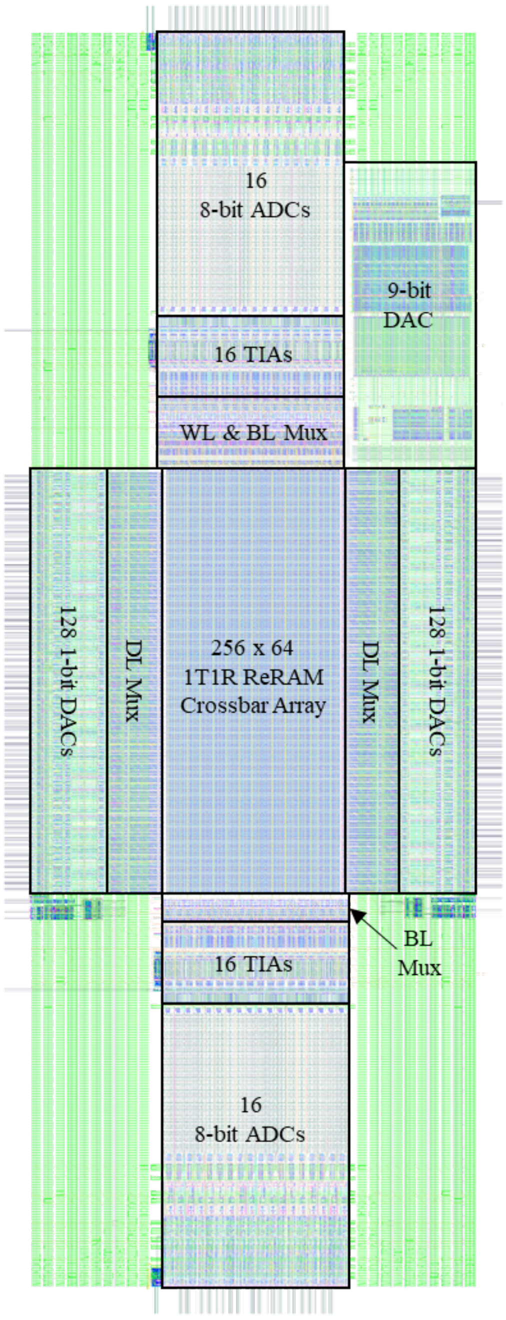


Figure 4-23. Prototype IC CIM macro floor plan.

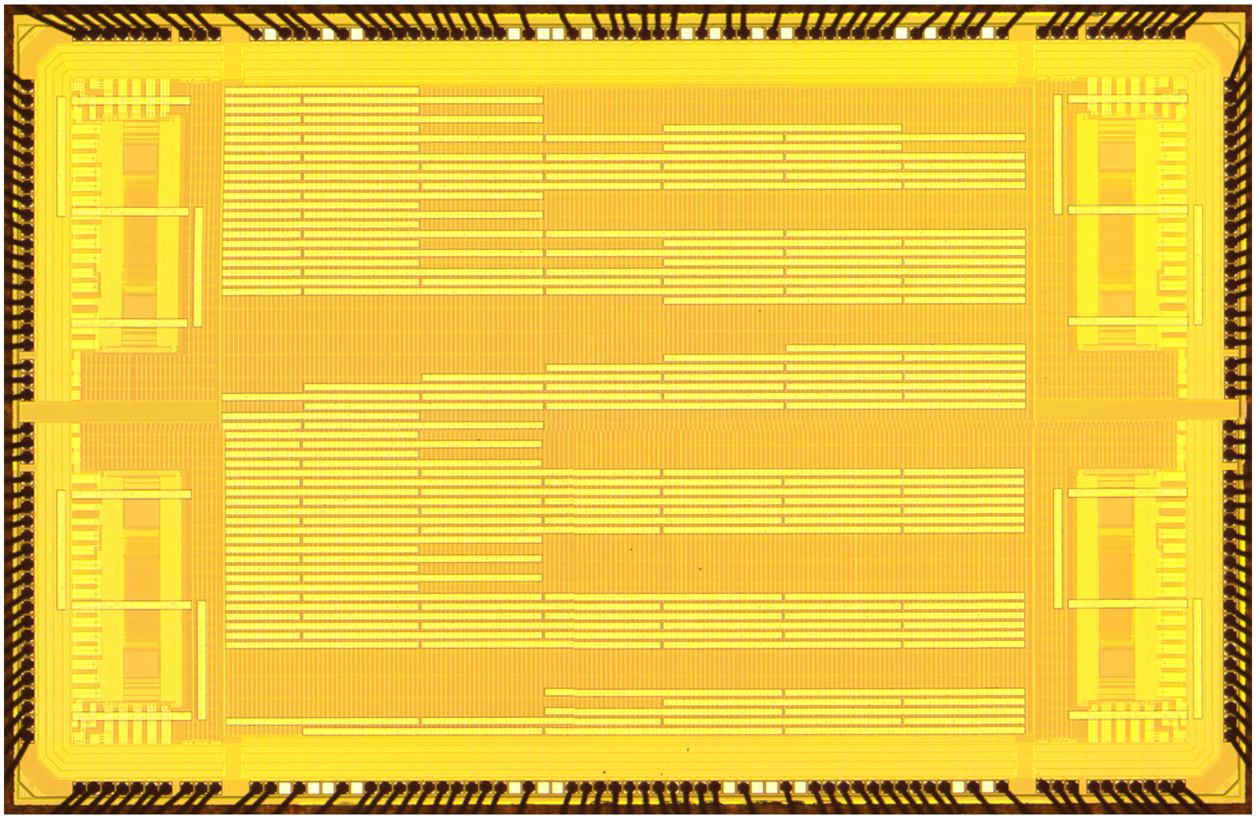


Figure 4-24. Fabricated IC die micrograph.

4.9 Measurement Results

The SOC measurement results section presents and discusses the evaluation of the prototype IC as an edge neural network accelerator. Fig. 4-24 shows a die micrograph of the prototype IC under test. First, the test setup and PC to IC communication are discussed. Then we show 2BPC MLC ReRAM programming on the prototype arrays with our on-chip high-voltage programming circuitry. Now, with an acceptable programming algorithm, we introduce a naïve neural network mapping approach to 2BPC ReRAM. Next, LeNet1 is fully mapped onto 3 different macros on a single chip and then we run batch inference. Finally, we discuss system benchmarking of 8-bit fully parallel VMM accelerators.

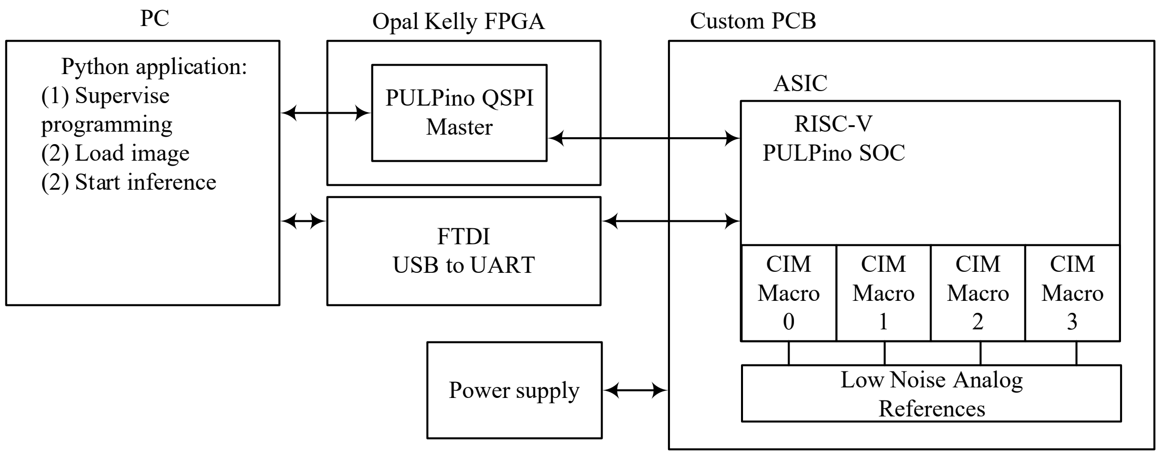


Figure 4-25. ASIC test interface.

4.9.1 Test Setup

Fig. 4-25 shows a diagram of the testing interface between PC and custom SOC. Processor instructions written in C are compiled into machine code. The machine code is loaded onto the PCB by a python application and FPGA-based quad-SPI master controller. Since the PULPino acts as an AXI QSPI slave during programming, we modified the ABP QSPI master (Fig. 4-13) HDL for implementation on the FPGA. An FTDI USB to UART bridge serves as a data link between C applications running on hardware and a parent python application. Figure 4-26 shows the test PCB complete[32] with analog and digital references, clock input, digital control, and daughterboard PCB. The prototype IC is wirebonded to the daughterboard PCB which both reduces I/O count between the ASIC die and printed circuit board; and provides necessary decoupling capacitance exceptionally close to the die. The daughterboard is easily replaced in the socket enabling the evaluation of a wide array of prototypes.

[32] PCB and package designed by Luke Wormald

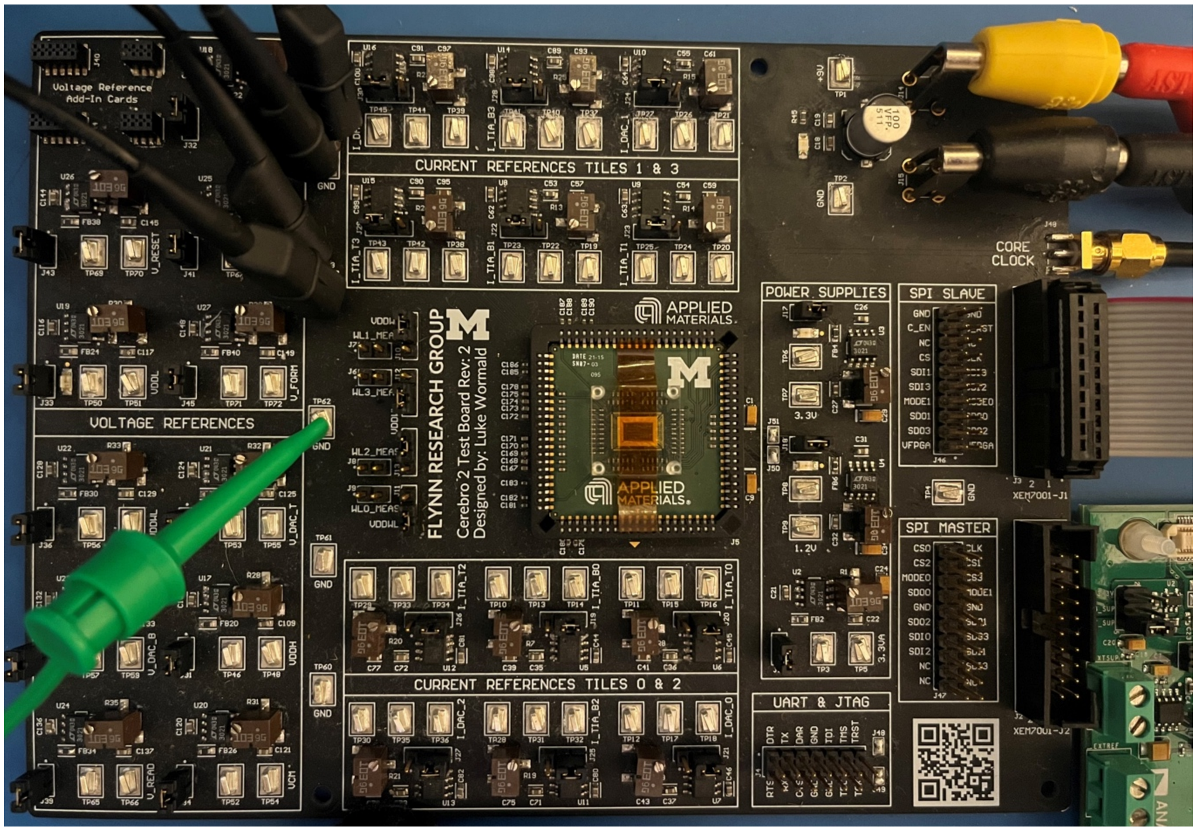

CURRENT REFERENCES TILES 1 & 3
VOLTAGE REFERENCES
FLYNN RESEARCH GROUP
Cerebro 2 Test Board Rev: 2
Designed by: Luke Wormald
APPLIED MATERIALS
POWER SUPPLIES
SPI SLAVE
SPI MASTER
CORE CLOCK
CURRENT REFERENCES TILES 0 & 2
UART & JTAG

Figure 4-26. Testing PCB with DUT.

Action	Operation Mode	Voltage	Pulse Duration	Pulse Number
Form a bit	FORM	4.9V	100ns	3
Set a bit	SET	4V	100ns	1-5
Reset a bit	RESET	2V	100ns	1
Program a bit	SET & RESET	4V & 2V	100ns	50

Table 4-3. ReRAM programming conditions.

4.9.2 Multi-Level Cell ReRAM

We show multi-level ReRAM programming with a closed-loop write-verify algorithm using fixed amplitude, fixed duration pulses delivered from the prototype IC. Table 4-3 shows the programming conditions used to achieve 2 BPC or 4 distinctive programmed states. Fig. 4-27 shows cumulative distribution function (CDF) plots of target programmed distributions and the effect of device retention. The plots show the distributions: (a) after programming, (b) after one hour, and (c) after three days. The states experience some drift but maintain excellent separation.

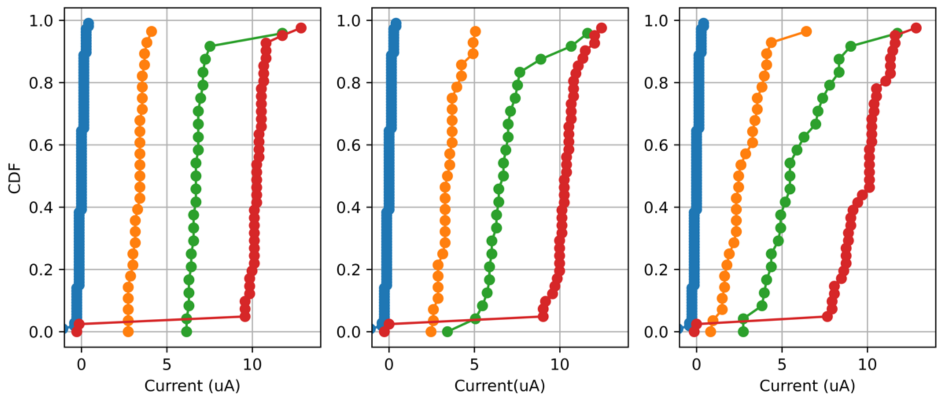


Figure 4-27. ReRAM devices programmed to 2-BPC shown in CDF plots.

4.9.3 Naïve Neural Network Mapping

Fig 4-28 shows a diagram of the LeNet 1 CNN architecture. The network is comprised of two convolutional layers and one fully connected layer connected to ten output classes. LeNet 1 is a reduced version of LeNet 5 and is used to classify MNIST handwritten digits (0-9) [42]. The network is considered a "toy network" but is well suited for demonstrating quantization-aware training on low-precision (2BPC) ReRAM and full on-chip neural network mapping.

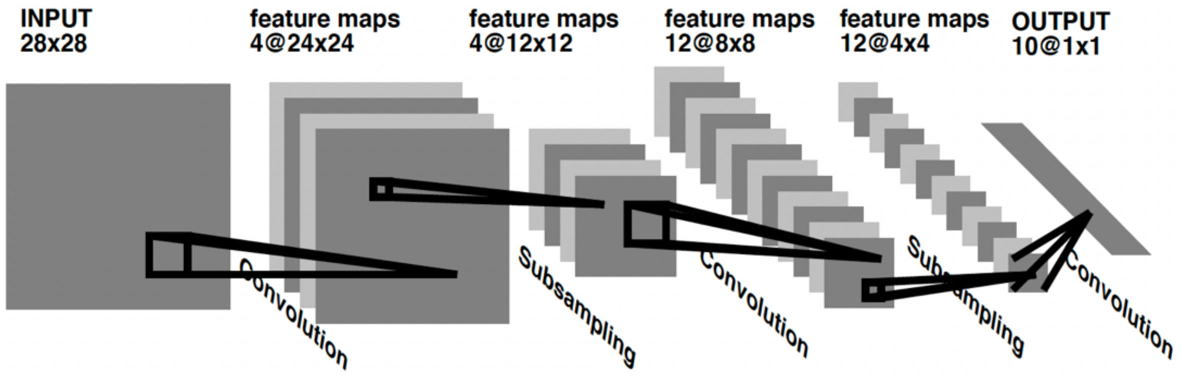


Figure 4-28. LeNet 1 convolutional neural network architecture [29].

Layer	Kernels	Shape	Total Parameters
Conv1	4	5x5x1	100
Conv2	12	5x5x4	1200
Fully Connected	10 (Classes)	192x1	1920
		Total	3220

Table 4-4. LeNet 1 architecture trainable parameters.

Table 4-4 lists the trainable parameters for LeNet 1. For demonstration, LeNet 1 is trained using quantization aware training for 2BPC ReRAM. The model is trained at 3BPC precision then mapped to two 2BPC ReRAM devices per parameter. A total of 6440 parameters are stored in ReRAM devices to implement on-chip LeNet1.

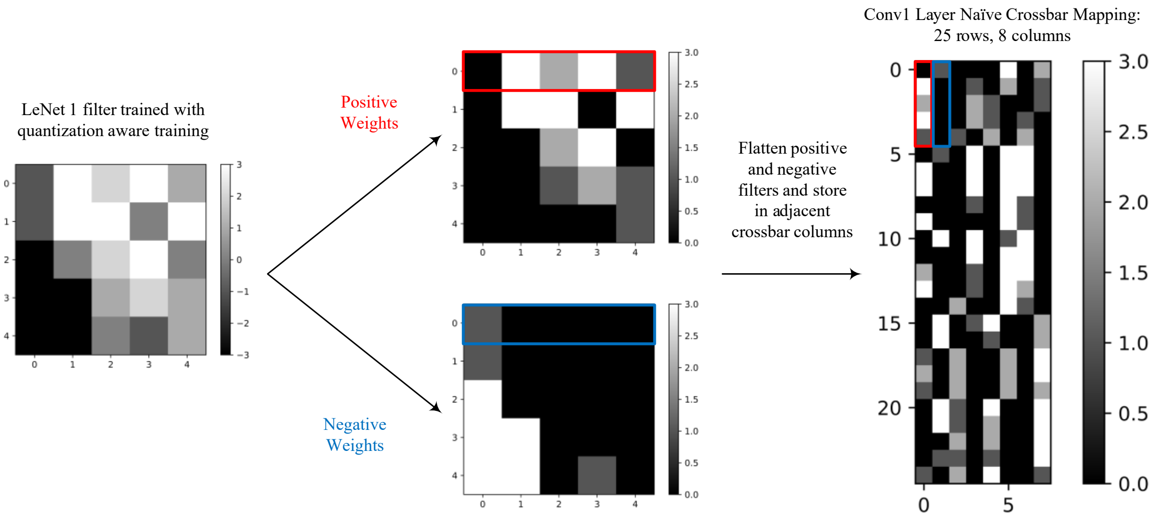

LeNet 1 filter trained with quantization aware training
Positive Weights
Negative Weights
Flatten positive and negative filters and store in adjacent crossbar columns
Conv1 Layer Naïve Crossbar Mapping: 25 rows, 8 columns

Figure 4-29. Dual array mapping

Trained weights are separated into positive and negative weight matrices then mapped to ReRAM devices in adjacent columns of the crossbar [27-30]. Mapping includes flattening the positive and negative weight matrices and then programming the ReRAM devices accordingly. Fig. 4-29 shows an example of mapping a trained kernel to ReRAM devices in two crossbar columns.

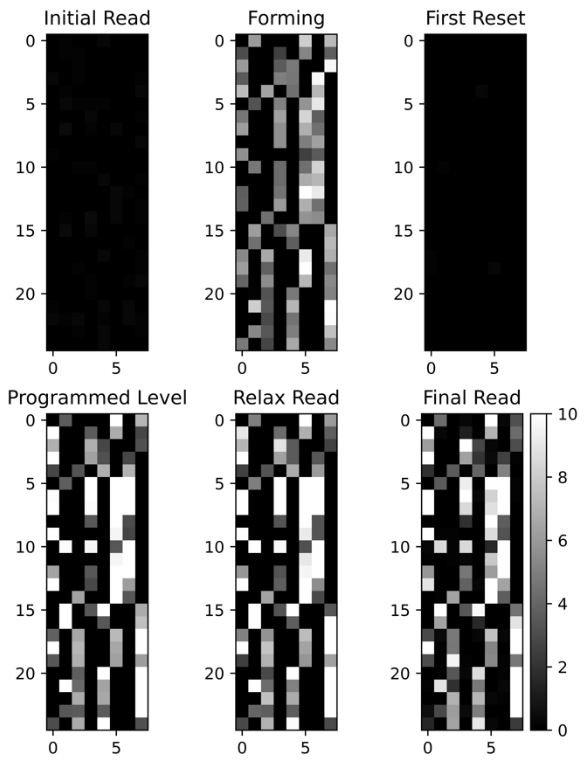


Figure 4-30. Example programming sequence for layer 1 weight storage.

Fig. 4-30 shows 6 grayscale the first LeNet 1 layer mapped onto the crossbar. Each image is a 25x8 grid of pixels, where each pixel represents the current of the ReRAM bitcell at that physical location. In the first image, the range of bitcells used for Conv1 storage is read out. The devices are all in the virgin state or "off" state. Next, the devices are formed with 3 pulses (Table 4-3). Then the devices are reset to check for any shorted devices. The third image shows that all devices reset appropriately. The bottom three images are related to programming. The bottom left image shows the final value of the write-verify algorithm for each bit. The quantized target programming map is shown in Fig. 4-29. The final two images in Fig. 4-30 are (1) readout 10ms after programming and (2) readout after the entire array is programmed. Fig. 4-31 shows CDF plots of all 6440 LeNet 1 parameters (1) after programming and (2) after inference.

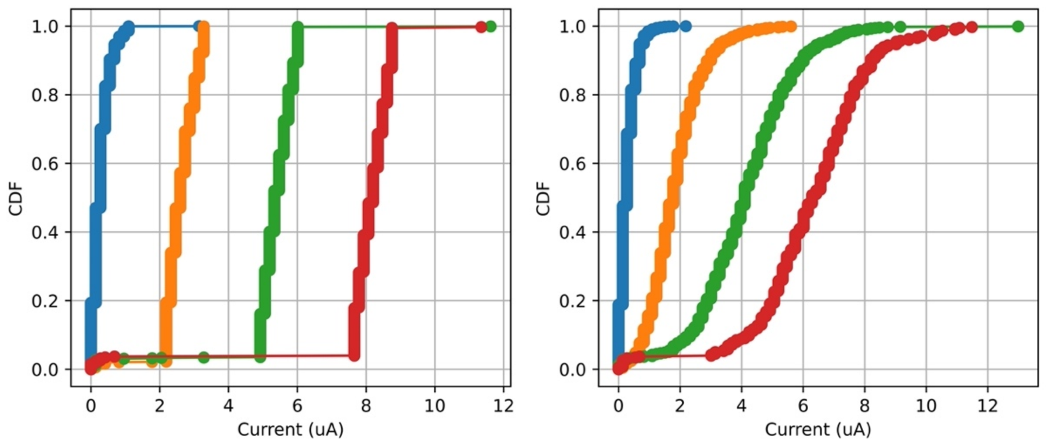


Figure 4-31. CDF plots of LeNet 1 parameters after programming and after inference.

4.9.4 Batch Inference

We performed the necessary on-chip VMMs to compute a batch of 128 images from the MNIST dataset using the programmed ReRAM devices shown in Fig. 4-31. On our parallel architecture, we compute 576 VMMs on conv1, 64 VMMs on conv2, and 1 VMM on the fully connected layer per image. Since we use positive and negative arrays, each input image is classified with a total of 1282 VMMs. A critical contribution of this work is that we apply the full input vector in parallel. The input vectors for LeNet 1 are as follows: 25x1 for conv1, 100x1 for conv2, and 192x1 for the fully connected layer.

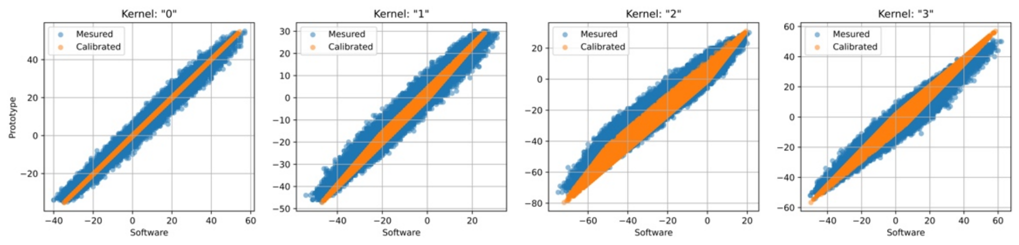


Figure 4-32. Measured and calibrated output activations from the LeNet 1 conv1 layer for 128 batch images.

Figs. 4-32 – 4-34 show the correlated batch output activations, or VMM results, for each kernel in each of the LeNet 1 convolutional layers. Each data point represents the measured channel MAC output activation plotted against the expected value. Since we know the VMM result a priori, the measured output is correlated to the expected output for channel calibration. Each of the figures shows the uncalibrated (blue) and calibrated (orange) channel outputs.

4.9.5 Calibration

Each kernel output activation requires two levels of calibration. In the first, the TIA and ADC gain and offset error are corrected for the positive and negative MAC results. Then the calibrated channels are subtracted to form the kernel output activations. The kernel activations also contain an error signature related to the ReRAM programming quality of the weights in that column. This too appears as a constant gain and offset error and can be corrected.

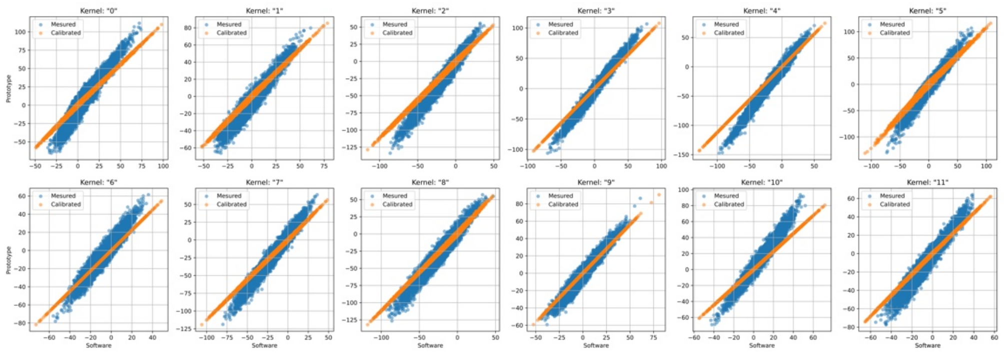


Figure 4-33. Measured and calibrated output activations from the LeNet 1 conv2 layer for 128 batch images.

Both the channel-level and activation-level calibration follow the same procedure. For brevity, we will discuss the calibration procedure on the fully connected layer. There are ten output activations (classes) in the fully connected layer and twenty output channels considering positive and negative array mapping. Since we know the programmed weight mapping, we can calculate the channel-level and activation-level results for a batch of inputs. For channel calibration, the correlation is measured between the measured channel outputs and the calculated result. Next, the channel with the highest R^2 value is selected, and the slopes (gain) and intercepts (offset) of all other channels are adjusted to match the slope and intercept of the selected channel.

For activation calibration, the correlation of the ten output classes is measured after subtracting the twenty calibrated positive and negative channels. The calibration process repeats, and all the activation slopes and intercepts are adjusted to match the one with the highest R^2 value. Fig. 4-34 shows the calibrated output class VMM results from the fully connected layer. The linearity of the orange plots shows the effectiveness of channel-level and activation-level calibration. The learned gain and offset parameters are stored on-chip for usage.

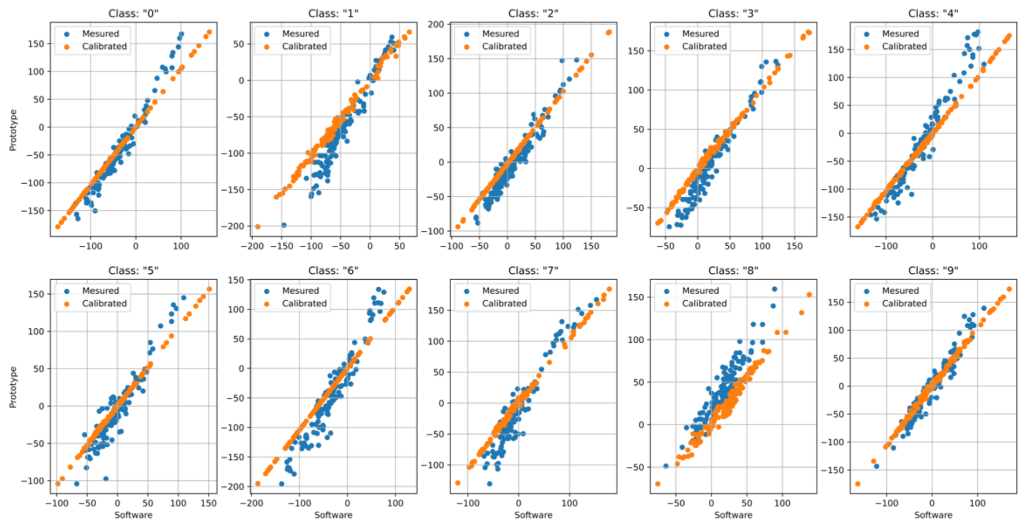
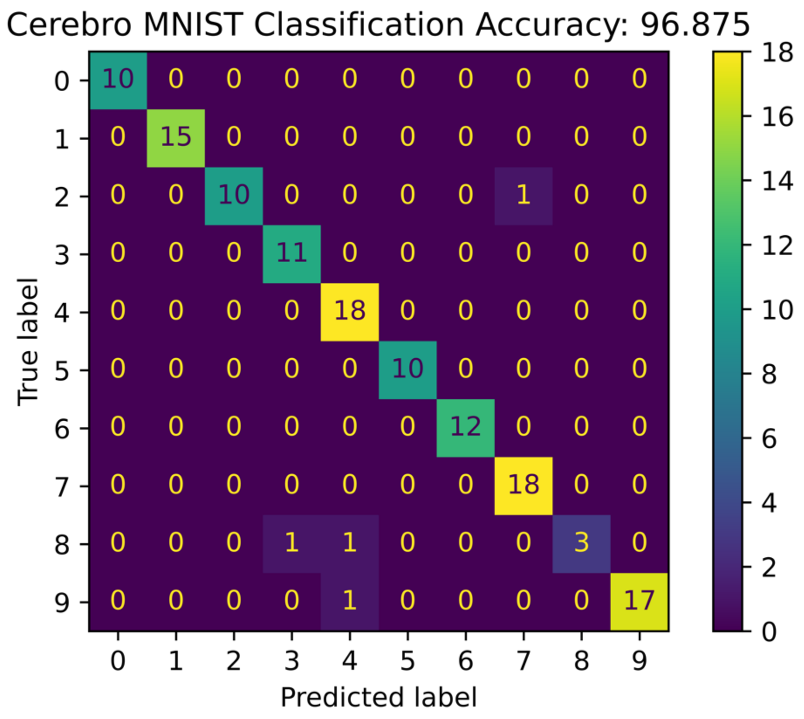


Figure 4-34. Measured and calibrated output activations from the LeNet 1 fully connected layer for 128 batch images.

4.9.6 Confusion Matrix

Fig. 4-35 shows a confusion matrix for a batch of 128 MNIST validation set images run on the prototype chip. The y-axis plots the true label provided by the MNIST dataset and the x-axis plots the output generated by the prototype. The plot shows that the network correctly classified 124/128 images corresponding to 96.8% accuracy. The quantization aware software model classified 126/128 images correctly corresponding to 98.4% accuracy. Figure 4-36 shows some examples of incorrectly classified images.



	0	1	2	3	4	5	6	7	8	9
0	10	0	0	0	0	0	0	0	0	0
1	0	15	0	0	0	0	0	0	0	0
2	0	0	10	0	0	0	0	1	0	0
3	0	0	0	11	0	0	0	0	0	0
4	0	0	0	0	18	0	0	0	0	0
5	0	0	0	0	0	10	0	0	0	0
6	0	0	0	0	0	0	12	0	0	0
7	0	0	0	0	0	0	0	18	0	0
8	0	0	0	1	1	0	0	0	3	0
9	0	0	0	0	1	0	0	0	0	17

Figure 4-35. Confusion matrix for a batch of 128 images from the MNSIT validation set.

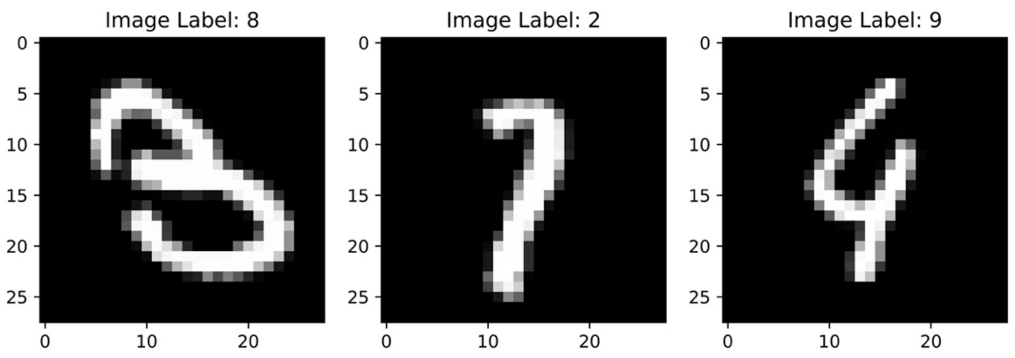


Figure 4-36. Examples of incorrect network classification.

4.9.7 Power Measurement

Table 4-5 summarizes the macro-level mixed-signal power measured on the prototype IC. Crossbar array power is a function of array utilization or the number of active cells in the array. An array utilization of 61.3% was used for crossbar array power consumption measurement. Figure 4-37 shows a breakdown of the macro power consumption during inference mode.

System	Power (µW)
Crossbar	61.22
ADC	449.36
Level Shifters	453.65
TIA	2131.19

Table 4-5. Power measurement summary.

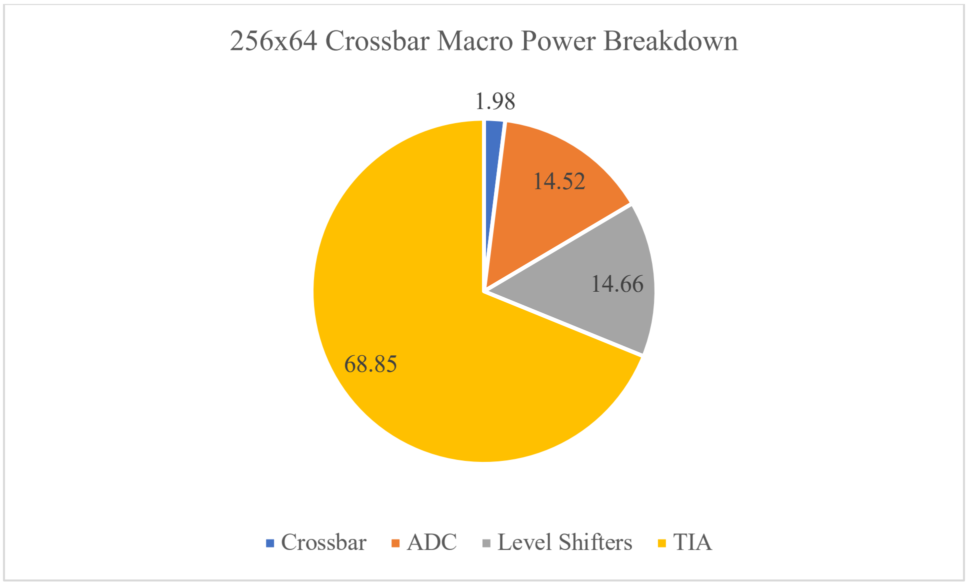


Figure 4-37. Single macro power distribution during MAC operation.

4.9.8 TOPS/W and System Summary

Maximum efficiency is evaluated on a single CIM macro by calculating the tera-operations per second per watt or TOPS/W. Next, we provide a straightforward method to calculate TOPS/W for 8-bit input, 8-bit output parallel architectures. The maximum efficiency describes the relationship between system throughput and system power consumption.

Crossbar	ReRAM Crossbar	1T1R
	ReRAM Bitcell	Filamentary, foundry integrated
	Array Dimension	256 x 64
	Minimum Current (nA)	300
	Maximum Current (µA)	3
	Levels	4,8
	Power (µW)	61
Single Mixed-signal Macro	# of Macros	4
	# of ADCs	32
	ADC Resolution (bits)	8
	# of DACs	256
	DAC Resolution (bits)	8
	Power (mW)	3.10
RISC-V	Power (mW)	100.44
System Performance	System Clock (MHz)	62.74 MHz
	Input (bits)	8
	VMM/S	1.74M
	OP/S	57.1 G
	OPS/W	18.45 T
Mixed-Signal Efficiency	Energy/VMM (nJ)	1.78
	Energy/Op (fJ)	54.21

Table 4-6. System summary.

System throughput is determined based on accelerator design and CIM array size. The system throughput is calculated as follows:

$$\frac{Operations}{Sec} = 2 * \frac{VMMs}{sec} * CIM\ rows * CIM\ columns$$

where the first term describes multiply & accumulate and the second term describes the computation time of the array size described in the last two terms. Next, the computation time is defined as:

$$\frac{VMMs}{sec} = \frac{1}{VMM\ period}$$

where the VMM period is the latency associated with computing the array described by CIM rows and CIM columns in Eq. 4-1. Our bit-serial macro computes the entire 256x64 crossbar VMM with 8-bit inputs and 8-bit outputs in 36 system clock cycles or 574ns at 62.74MHz. The maximum efficiency is 14.63 TOPS/W or 68.36 fJ per operation. Table 4-6 shows a detailed system summary.

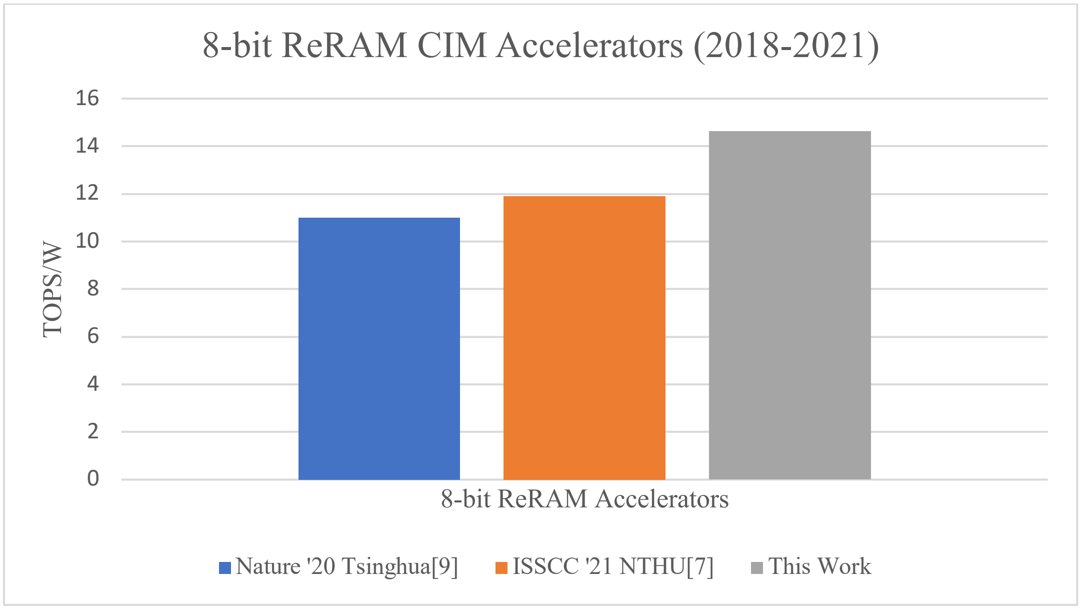


Figure 4-38. Comparison of 8-bit input ReRAM CIM Accelerators.

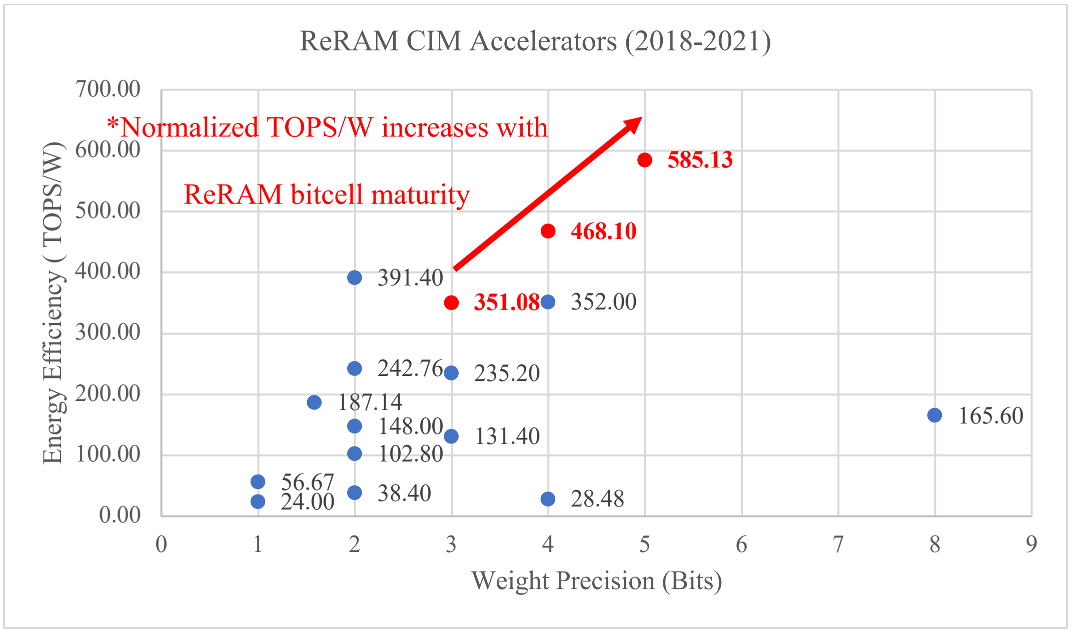


Figure 4-39. Survey of ReRAM CIM Accelerators from 2018-2021. The red points show our architecture with 2, 3, and 4 BPC MLC ReRAM bitcells used as 3-, 4-, and 5-bit DNN weights, respectively.

Finally, we compare this work with other recently published accelerators from 2018-2021. Fig. 4-38 compares our work to other published 8-bit input ReRAM accelerators. Our work shows that ADC-assisted bit-serial operation leads to a 33% increase compared to current state-of-the-art energy efficiency. Next, we normalize all ReRAM accelerators to 1-bit by 1-bit MAC operation using the method presented by the authors in [43]. For fair comparison among different accelerator architectures, the maximum efficiency (TOPS/W) of multi-bit input, multi-bit weight ReRAM accelerators is scaled by the product of the input and weight precision. For example, our prototype has 8-bit input and 3-bit weight, and the maximum efficiency to compute the entire array is 14.63 TOPS/W. When scaled to 1-bit by 1-bit MAC operation, the normalized maximum efficiency is 351.08 TOPS/W. Fig. 4-39 shows recent normalized ReRAM accelerators and Table 4-7 provides details. It is important to note that the current limitation of the MLC capability is ReRAM device quality and not the mixed-signal circuit operation. Therefore, we project the expected TOPS/W for four and five BPC (shown by the red arrow in Fig 4-39). Table 4-8 shows a detailed comparison table for multi-bit input (non-binary mode) ReRAM accelerators.

Publication	**Year**	**Group**	**Input Bit**	**Bits Per Cell**	**Weight Bit**	**Reported TOPS/W**	**TOPS/W Normalized to 1-bit Input**	**Reference**
ISSCC	2018	NTHU	1	1	2	19.20	38.40	[1]
VLSI	2018	PSCS	1	8	8	20.70	165.60	[39]
ISSCC	2019	NTHU	2	1	3	21.90	131.40	[2]
Nature	2019	UMICH	6	Analog	Analog	1.78	14.24	[30]
ISSCC	2020	NTHU	1	1	2	121.38	242.76	[3]
TED	2020	ASU/GT	1	1	1	24.00	24.00	[4]
SSCL	2020	ASU/GT	1	2	2	51.40	102.80	[10]
ISSCC	2020	Tsinghua	1	2	3	78.40	235.20	[40]
ISSCC	2020	Stanford	1	Analog	Analog	74.00	148.00	[8]
Nature	2020	Tsinghua	8	3	4	11.00	352.00	[9]
ISSCC	2021	GT	1	1	1	56.67	56.67	[5]
CICC	2021	GT	1	1	1.58	118.44	187.14	[6]
ISSCC	2021	NTHU	1	1	2	195.70	391.40	[7]
This Work	2021	UMICH	8	2	3	14.63	351.08	

Table 4-7. Summary of ReRAM CIM accelerator survey.

	ISSCC '19 NTHU [2]	Nature '19 [30]	ISSCC '20 NTHU [3]	VLSI '20 Stanford [8]	Nature '20 Tsinghua [9]	CICC '21 GT [5]	ISSCC '21 NTHU [7]	This Work
Technology	55	180	22	130	130	40	22	65
	Macro	Fully Integrated	Macro	Macro	Discrete	Macro	Macro	Fully Integrated
ReRAM Bits	1	Analog	1	1	3	1	1	2
Subarray	N/A	N/A	N/A	N/A	8	N/A	8	4
Subarray Size	256x512	54x108	512x512	256x256	128x128	256x256	1024x512	256x64
Capacity	1Mb		2Mb	N/A	N/A	100Kb	4Mb	64Kb
Precision (I)	2	4	4	2	8	1	1	8
Precision (W)	3	Analog	4	1	4	1.58	2	3
Precision (O)	4	13	11	N/A	8	4	14	8
Column Sensing	4b ADC	13b ADC	6b ADC	N/A	8b ADC	4b ADC	1b SA	8b ADC
Active Rows	9	108	16	16-96	128	9	4	256
Operations Per ADC	36	N/A	N/A	N/A	N/A	N/A	4	256
Supported Algorithm	CNN	MLP, Sparse	CNN	CNN	CNN	CNN	CNN	CNN
Reported TOPS/W	21.9	1.78	28.9	N/A	11	118.44	195.7	14.62
Energy Efficiency Normalized to 1-bit	131.4	28.48	242.76	N/A	352	187.14	391.4	351.08
Fully Mapped	No	No	No	No	Yes	No	No	Yes
Dataset	CIFAR10	N/A	CIFAR10	N/A	CIFAR10	CIFAR10	CIFAR10	MNIST
Accuracy	85.5%	N/A	90.18%	N/A	91.38%	90.18%	N/A	96.80%

Table 4-8. Comparison table of recent 8-bit CIM accelerators.

Chapter 5 Conclusion

Efficient large-scale matrix operations are essential for the future of AI and machine learning. However, conventional computing architectures have separate "processing" and "memory" which induces significant latency and wastes energy. To solve this bottleneck, processing and memory are combined and the compute is performed in the physical domain with analog values on Multi-Level Cell (MLC) Resistive Random-Access Memory (ReRAM) crossbars. The compute is in parallel and in place with O(1) time complexity and energy consumption is amortized across the crossbar array structure. This new computing paradigm presents new challenges from the bottom up, including devices, circuits, integration, system, and algorithm.

This work addresses these challenges and was performed by the students recognized in this thesis. Together our work makes significant contribution to the field of analog Compute-in-Memory (aCIM). We present two mixed-signal architectures for fully parallel aCIM on MLC crossbar arrays. Our first prototype is the first fully integrated CMOS-ReRAM SoC for AI applications. We introduce a mixed-signal architecture for charge-domain crossbar operation. Dedicated 6-bit RTZ pulse DACs apply row inputs in parallel and dedicated 13-bit charge-domain ADCs integrate column charge. A 54x108 ReRAM array is post-processed on top of the die for NVM aCIM. An on-chip RISC host processor directs on-chip write verify, on-chip training, and inference. We demonstrate several basic functions for AI applications.

The author contributed to the first prototype in the following areas: (1) System concept, design, and integration, (2) mixed-signal timing controller concept and design, (3) 1-bit RTZ concept, design, and layout (4) mixed-signal register interface concept and design, (5) top-level mixed-mode verification, (6) test PCB design, (7) test setup design, (8) ASIC bring-up and calibration. (9) RISC host processor firmware development.

In our second prototype, we address the lessons learned from the first prototype with a multi-macro RISC-V-based CMOS-ReRAM SoC. We introduce ADC-assisted bit-serial operation to deliver an 8-bit resolution parallel MAC with minimal latency. A compact low-power binary-weighted ADC enables one ADC per bit line pair. A pitch-matched switch architecture supports high and low voltage operations on foundry integrated, multi-bit 1T1R ReRAM crossbar arrays. Both, lead to a pitch-matched design and solve the integration challenges. Finally, on-chip circuits support write-verify programming and ReRAM characterization. A RISC-V host processor directs device programming algorithms for DNN parameter mapping to ReRAM devices. For demonstration, we map a full DNN on-chip and perform inference.

The author contributed to the second prototype in the following areas: (1) System concept, design, and integration, (2) mixed-signal timing controller and design, (3) mixed-signal register interface concept and design, (4) crossbar DL, BL, WL, analog switch interface concept and design, (5) pre-silicon SOC/mixed-signal verification and co-sim including RISC-V compiler and C firmware development, (6) post-silicon validation and test including test environment design and mixed-signal calibration,(7), ReRAM device programming to 2 BPC, (8) on-chip DNN mapping to three macros on a single chip, (9) LeNet1 neural network demonstration, and (10) Quantization-aware training for ReRAM weight mapping.

In conclusion, the future of aCIM depends upon aggressive scaling of current progress. Neither the research community nor industry has yet to implement a large-scale practical ReRAM-based DNN with millions of parameters. Lab-to-fab transfer of new memristive materials for NVM

CIM remains a key bottleneck toward widespread adoption. Given that these problems will be solved, there is still room for improvement at the circuit level. First, throughput and energy efficiency will improve by moving the design to a modern CMOS process node. We can significantly increase the clock speed to increase VMM throughput. Second, a fully automated crossbar switching matrix would further improve system performance by removing costly register manipulations in-between matrix operations. Finally, each macro should maintain the ability to apply activation functions locally. This will further reduce host processor responsibility and enable efficient tile-to-tile communication. With these improvements, aCIM can achieve maximum efficiency above 1000’s of TOPS/W.

Bibliography

[1] W.-H. Chen et al., “A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors,” in Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC), 2018.

[2] C.-X. Xue et al., “A 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN-based AI edge processors,” in Proc. IEEE Int. Solid- State Circuits Conf. (ISSCC), 2019.

[3] C.-X. Xue et al., “A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices,” in Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC), 2020.

[4] S. Yin, X. Sun, S. Yu, and J.-S. Seo, “High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90nm CMOS,” in IEEE Trans. Electron Devices, vol. 67, no. 10, pp. 4185–4192, 2020. DOI: 10.1109/TED.2020.3015178

[5] J.H. Yoon et al., “A 40nm 64Kb 56.67 TOPS/W Read-Disturb-Tolerant Compute-in-Memory/Digital RRAM Macro with Active-Feedback-based Read and in-situ Write Verification,” in IEEE International Solid-State Circuits Conference (ISSCC) Digest of Technical Papers, 2021

[6] J.H. Yoon et al., “A 40nm 100Kb 118.44TOPS/W Ternary-weight Compute-in-Memory RRAM Macro with Voltage- sensing Read and Write Verification for reliable multi-bit RRAM operation,” in Custom Integrated Circuits Conference (CICC), 2021.

[7] C. Xue et al., “A 22nm 4Mb 8b-Precision ReRAM Computing-in-memory Marco with 11.91 to 195.7 TOPS/W for Tiny AI Edge Devices,” in IEEE Int. Solid-State Circuits Conf. (ISSCC), San Francisco, CA, USA, Feb. 2021, pp.246-247.

[8] W. Wan et al., “A 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models,” in Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC), 2020

[9] P. Yao et al., “Fully hardware-implemented memristor convolutional neural network,” Nature, vol. 577, no. 7792, pp. 641–646, 2020.

[10] W. He et al., “2-bit-per-cell RRAM based in-memory computing for area-/energy-efficient deep learning,” in IEEE Solid-State Circuits Lett., vol. 3, pp. 194–197, 2020.

[11] S. Cosemans *et al*., “Towards 10000TOPS/W DNN inference with analog in-memory computing – a circuit blueprint, device options and requirements,” *IEEE Int. Electron Devices Meeting (IEDM),* San Francisco, CA, USA, Dec. 2019, pp. 22.2.1-22.2.4.

[12] S. Kim, J. Zhou, and W. D. Lu, “Crossbar RRAM arrays: selector device requirements during write operation,” in *IEEE Trans. on Electron Devices,* vol. 61, no. 8, pp. 2820-2826, Aug. 2019.

[13] J. Zhou, K. Kim, and W. D. Lu, “Crossbar RRAM arrays: selector device requirements during read operation,” in *IEEE Trans. on Electron Devices,* vol. 61, no. 5, pp. 1369-1376,

May 2014.

[14] M. A. Zidan *et al*., "Memristor-based memory: the sneak paths problem and solutions," *Microelectronics J.*, vol. 44, no. 2, pp 176-183, Feb. 2013.

[15] S. H. Lee, X. Zhu, and W.D. Lu, "Nanoscale resistive switching devices for memory and computing applications," in *Nano Res.,* Jan. 2020.

[16] H. Kim, H. Nili, M.R. Mahmoodi, and D.B. Strukov, "4K-memristor analog-grade passive crossbar circuit," *arXiv:1906.12045,* Jun. 2019.

[17] H. D. Lee *et al.,* "Integration of $4F^2$ selector-less crossbar array 2Mb ReRAM based on transition metal oxides for high density memory applications," *Symposium on VLSI Technology (VLSIT)*, Honolulu, HI, USA, Jun. 2012, pp. 151-152.

[18] M. Hsieh *et al.,* "Ultra high density 3D via RRAM in pure 28nm CMOS process," *IEEE Int. Electron Devices Meeting (IEDM),* Washington, DC, USA, Dec. 2013, pp. 10.3.1-10.3.4.

[19] T. Liu *et al.,* "A $130.7mm^2$ 2-layer 32Gb ReRAM memory device in 24nm technology", *IEEE Int. Solid-State Circuits Conf. (ISSCC),* San Francisco, CA, USA, Feb. 2013, pp. 210-211.

[20] J. Zahurak *et al.,* "Process integration of a 27nm, 16Gb Cu ReRAM", *IEEE Int. Electron Devices Meeting (IEDM),* San Francisco, CA, USA, Dec. 2014, pp. 6.2.1-6.2.4.

[21] R. Fackenthal *et al.,* "A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2014, pp. 338-339.

[22] S. H. Jo, T. Kumar, S. Narayanan, W.D. Lu, and H. Nazarian, "3D-stackable crossbar resistive memory based on Field Assisted Superlinear Threshold (FAST) selector," *IEEE Int. Electron Devices Meeting*, San Francisco, CA, Dec. 2014, pp. 6.7.1-6.7.4.

[23] S. H. Jo, T. Kumar, S. Narayanan, H. Nazarian, "Cross-point resistive RAM based on field-assisted superlinear threshold selector," in *IEEE Trans. Electron Devices,* vol. 62, no. 11, pp. 3477-3481, Nov. 2015.

[24] L. Ni *et al*., "Distributed in-memory computing on binary RRAM crossbar," in *ACM Journal on Emerging Technologies in Computing Systems,* vol. 13 no. 3, pp. 1-18, Mar. 2017.

[25] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2704-2713, doi: 10.1109/CVPR.2018.00286.

[26] A. Pappalardo (2021) Xilinx/brevitas[Source code]. https://github.com/Xilinx/brevitas

[27] Q. Wang, X. Wang, S. H. Lee, F. Meng and W. D. Lu, "A Deep Neural Network Accelerator Based on Tiled RRAM Architecture," *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 14.4.1-14.4.4, DOI: 10.1109/IEDM19573.2019.8993641.

[28] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," Nature Communications 11, 2473 (2020).

[29] Y. LeCun et al., "Handwritten Digit Recognition with a Back-Propagation Network," *1989 NIPS,* 1989, pp. 396-404.

[30] F. Cai et al., "A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations," Nat. Electron., vol. 2, no. 7, pp. 290-299, Jul. 2019.

[31] S. Kin, J. Zhou, W. D. Lu, "Crossbar RRAM arrays: selector device requirements during

write operation," *IEEE Trans. Electron Devices*., vol. 61, no. 8, pp. 2820-2826, Jun. 2014.

[32] E. J. Merced-Grafals *et al.*, "Repeatable, accurate, and high speed multi-level programming of memristor 1T1R arrays for power efficient analog computing applications," *Nanotechnology*, vol. 27, no. 36, Sep. 2016.

[33] Y. Lim and M. P. Flynn, "A 1 mW 71.5dB SNDR 50 MS/s 13 bit fully differential ring amplifier base SAR-assisted pipeline ADC," *IEEE J. Solid-State Circuits,* vol. 50, no. 12, pp. 2901-2911, Sep. 2015.

[34] Y. Lim and M. P. Flynn, "A calibration-free 2.3 mW 73.2 dB SNDR 15b 100 MS/s four-stage fully differential ring amplifier based SAR-assisted pipeline ADC," *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Kyoto, Japan, Jun. 2017, pp. 98-99.

[35] M. Miyahara *et al.*, "A low-noise self-calibrating dynamic comparator for high-speed ADCs," *2008 IEEE Asian Solid-State Circuits Conference*, Fukuoka, Japan, Nov. 2008, pp. 269-272.

[36] "Overview :: AltOR32 – alternative lightweight openRISC CPU :: openCores." [Online]. Available: https://opencores.org/projects/altor32

[37] T. Chang *et al.*, "Synaptic behaviors and modeling of a metal oxide memristive device," *Appl. Phys. A*, 102, pp. 857-863, Mar. 2011.

[38] T. Chang *et al.*, "Short-term to long-term memory transition in a nanoscale memristor," *ACS Nano,* 5, 9, pp. 7669-7696, Aug. 2011.

[39] R. Mochida et al., "A 4M synapses integrated analog ReRAM based 66.5 TOPS/W neural-network processor with cell current controlled writing and flexible network architecture," in Proc. IEEE Symp. VLSI Technol., 2018

[40] Q. Liu et al., "A fully integrated analog ReRAM based 78.4 TOPS/W compute-in-memory chip with fully parallel MAC computing," in Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC), 2020.

[41] ETH Zurich (2017) PULPino[Source code]. https://github.com/pulp-platform/pulpino

[42] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[43] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-Memory Chips for Deep Learning: Recent Trends and Prospects," in IEEE Circuits and Systems Magazine, vol. 21, no. 3, pp. 31-56, 2021, DOI: 10.1109/MCAS.2021.3092533.