Characterization and Optimization of ReRAM-based Analog Crossbar Arrays for Neuromorphic Computing

by

Jesse Short

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2023 by the
Graduate Supervisory Committee:

Matthew Marinella, Chair
Hugh Barnaby
Ivan Sanchez Esqueda

ARIZONA STATE UNIVERSITY

December 2023

ABSTRACT

Machine learning advancements have led to increasingly complex algorithms, resulting in significant energy consumption due to heightened memory-transfer requirements and inefficient vector matrix multiplication (VMM). To address this issue, many have proposed ReRAM analog in-memory computing (AIMC) as a solution. AIMC enhances the time-energy efficiency of VMM operations beyond conventional VMM digital hardware, such as a tensor processing unit (TPU), while substantially reducing memory-transfer demands through in-memory computing. As AIMC gains prominence as a solution, it becomes crucial to optimize ReRAM and analog crossbar architecture characteristics.

This thesis introduces an application-specific integrated circuit (ASIC) tailored for characterizing ReRAM within a crossbar array architecture and discusses the interfacing techniques employed. It discusses ReRAM forming and programming techniques and showcases chip's ability to utilize the write-verify programming method to write image pixels on a conductance heat map. Additionally, this thesis assesses the ASIC's capability to characterize different aspects of ReRAM, including drift and noise characteristics. The research employs the chip to extract ReRAM data and models it within a crossbar array simulator, enabling its application in the classification of the CIFAR-10 dataset.

ACKNOWLEDGMENTS

I would like to express my thanks to Dr. Matthew Marinella for his invaluable guidance and unwavering support throughout the course of this research. Dr. Marinella served as a cornerstone of this research, providing both mentorship and expertise that were instrumental in its development.

I would also like to give a special thanks to the professionals at Sandia National Laboratory for their generosity in sharing their insights, data, and resources, which have greatly contributed to the success of this research endeavor. Their collective efforts have played a pivotal role in the development of this thesis.

I would like to individually thank Nad Gilbert, a Sandia employee, for the design of Cairn and for working closely with me during the debug process. Nad's expertise in chip design has been instrumental in shaping the core of this research.

I would like to extend my gratitude to the exceptional team of students who collaborated with me under the guidance of Dr. Marinella. Their efforts and dedication were pivotal in achieving data in this thesis and providing insights to this research.

Finally, I would like to extend my appreciation to Dr. Hugh Barnaby and Dr. Ivan Sanchez Esqueda for their contributions as members of my thesis committee. I am grateful for their insight and feedback that may be used to improve the quality of this thesis and the research behind it.

TABLE OF CONTENTS

Page

LIST OF TABLES v

LIST OF FIGURES vi

CHAPTER

1 INTRODUCTION 1

1.1 Resistive Crossbar Arrays 2

1.2 Background 3

1.2.1 'In-Memory Computing with Memristor Arrays' [12] 4

1.2.2 'Reinforcement learning with analogue memristor arrays' [13] 7

1.2.3 'Thousands of conductance levels in memristors integrated on CMOS' [14] 11

1.3 Need for Characterization 15

2 CAIRN DESIGN 16

2.1 Top Level Functionality 16

2.2 Cairn Crossbar 19

2.2.1 Column Access Circuitry 21

2.2.2 Pulser Circuitry 22

3 INTERFACE DESIGN 25

4 CAIRN DEBUG 28

4.1 Polarity Bit Error 28

4.2 Row Resistance Problem 31

4.3 Column 31 Issue 35

5 RESULTS 39

5.1 Forming & Conductance Tuning 39

5.2 Drift & Noise Characterization 44

5.3 CIFAR-10 Accuracy Modeling 53

6 CONCLUSION 56

REFERENCES 59

APPENDIX

A STATEMENT FOR 'ARRAY-SCALE CHARACTERIZATION OF RERAM ARRAYS FOR ANALOG IN-MEMORY COMPUTING' 61

LIST OF TABLES

Table Page

2.1 List of Cairn Configurations 18

5.1 CIFAR-10 Classification Accuracy 55

LIST OF FIGURES

Figure Page

1.1 VMM Crossbar Array Circuit Illustration 2
1.2 Li Et Al's 1T1R Cell Illustration [12] 4
1.3 Li Et Al's ReRAM Voltage Sweep [12] 5
1.4 Li Et Al's ReRAM Conductance Vs Time Measurement [12] 6
1.5 Li Et Al's Neural Network Classification Results [12] 7
1.6 Wang Et Al's 128x64 1T1R Layer Allocation [13] 8
1.7 Wang Et Al's ReRAM Programming Procedure [13] 8
1.8 Wang Et Al's Cart Pole and Mountain Car Games Illustration [13] 10
1.9 Wang Et Al's Cart Pole Game Results [13] 10
1.10 Rao Et Al's 2048 Conductance Levels Sweep [14] 12
1.11 Rao Et Al's Reads Before and After Denoise [14] 13
1.12 Rao Et Al's Denoising Removing Incomplete Channel [14] 14
1.13 Rao Et Al's Denoising Completing Incomplete Channel [14] 14
2.1 Interposer Floor Plan 16
2.2 Cairn Architecture Block Diagram 17
2.3 Cairn Simplified Crossbar Array 20
2.4 Cairn Column to Wire Bond Pad Circuitry 21
2.5 Bitcell Pulser Circuitry 23
2.6 Set and Reset Pulse Configurations 23
3.1 Physical Interface Setup 25
3.2 Interface Design and Data Flow Diagram 26
4.1 Column Output Codes Vs Vin1 Sweep 28
4.2 Absolute Value of Column Output Codes Vs Vin1 Sweep 29
4.3 Heat Map of Currents to Show Polarity Bit Flip 30

Figure Page

4.4 Parallel Read Heat Map of Shunt Device Conductances 31

4.5 Serial Read Heat Map of Shunt Device Conductances 32

4.6 Single Column With Parasitic Resistance Calculation 33

4.7 Parallel Columns With Parasitic Resistance Calculation 34

4.8 Heat Map of Shunt Devices With Column 31 Active 35

4.9 Circuit Configuration for B1500 Experiment 36

4.10 Column 31 Current Versus Vin0 Voltage Sweep 37

4.11 Column 31 Current Versus Vin0 Voltage Sweep With 0v Forced on Column 37

5.1 Write-verify 'Set' Example 40

5.2 Write-verify 'Reset' Example 41

5.3 Write-verify Overshoot Example 41

5.4 Example of Breaking Device 42

5.5 Conductance Heatmap Images 43

5.6 Device Stability for 40 Minutes Measured Serially 44

5.7 Device Stability for 12 Hours 46

5.8 Median Drift at Four Target Conductance Levels 47

5.9 Standard Deviation Vs Target Conductance Level 47

5.10 Standard Deviation of Conductance Vs Time 48

5.11 Median Standard Deviation for 12 Hours With No Biphasic Pulses 50

5.12 Median Standard Deviation for 12 Hours With 1 Biphasic Pulse 51

5.13 Median Standard Deviation for 12 Hours With 5 Biphasic Pulses 51

5.14 Median Drift for 12 Hours With No Biphasic Pulses 52

5.15 Median Drift for 12 Hours With 1 Biphasic Pulse 52

Figure Page

5.16 Median Drift for 12 Hours With 5 Biphasic Pulses 53
5.17 CIFAR-10 Classification Accuracy Vs ReRAM Drift Time With No Denoise 54
5.18 CIFAR-10 Classification Accuracy Vs ReRAM Drift Time With Denoise 54

Chapter 1

INTRODUCTION

Machine learning (ML) has gained significant prominence within the continually evolving domain of engineering. Due to the increase in algorithm complexity, engineers are becoming more commonly faced with the problem of computationally expensive machine learning. Advanced ML algorithms are known to consume a large amount of energy over the long period of time that it takes to train these models [1, 2]. There are many factors causing more energy consumption in ML, but a root cause is the constant transfer of data between separated memory and compute modules [3, 4, 5, 6]. As ML algorithms become more advanced, so do the memory-transfer requirements for performing computations on the data. Furthermore, repetitive matrix computations must be performed on increasingly large amounts of data, further increasing time-energy costs [5, 7].

These challenges have motivated architects to design more optimal methods and hardware architectures for ML work-loads. Most solutions suggest parallel digital architectures for ML acceleration. Parallel hardware architectures are useful for ML due to the parallel nature of matrix multiplication, which is the key ML operation. Using the highly parallel graphics processing unit (GPU) and tensor processing unit (TPU) architectures for training and classification has become a popular solution [8, 9]. Because of the success of GPUs and TPUs in ML, most advanced ML models are no longer trained using CPUs exclusively [1].

1.1 Resistive Crossbar Arrays

Beyond digital accelerators, a resistive crossbar array can perform vector-matrix multiplication (VMM) operations in the analog domain. Crossbar arrays are useful as multiplication can be done at the circuit level by multiplying a voltage across a resistor (Ohm's law) and addition can be performed by summing currents on a single node (Kirchoff's current law). In the case of a VMM operation, the target matrix can be programmed into the array of resistors while the vector values can be supplied by voltages on the rows. The output vector is then realized as the sum of the parallel currents at the end of each column (Figure 1.1). The matrix values for a resistive crossbar VMM are usually represented by the conductance of the resistors since I=VxG. Hence, Figure 1.1 labels the resistors with their conductance.

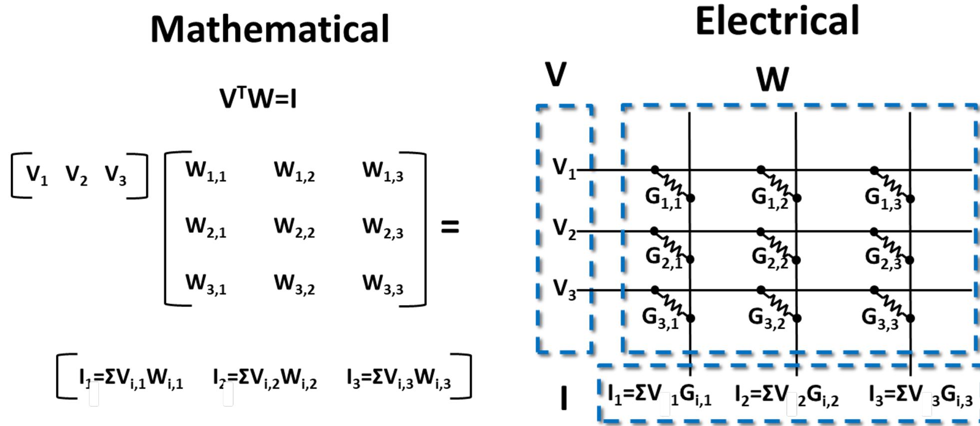

Figure 1.1: VMM crossbar array circuit illustration

Emerging non-volatile memory technologies, referred to as resistive random access memory (ReRAM), may be used as the tunable resistors in Figure 1.1. ReRAM has the ability to finely tune and retain a specific conductance state. The conductance of a ReRAM device is changed by applying voltage pulses across the device. Increasing the conductance of the device can be done by applying a voltage pulse to the top electrode (TE) of the device; this is also known as set operation. Likewise, a reset

operation, to decrease the conductance, is done by applying a voltage pulse to the bottom electrode (BE) of the device.

Using ReRAM in a crossbar array allows for a weight-stationary operation, known as in-memory computing (IMC). IMC is useful in the context of ML as it eliminates significant data movement, ultimately leading to higher energy efficiency [10, 11]. While digital domain calculations are discrete and repeatable, these analog in-memory computing (AIMC) computations are performed in a mix of the analog and digital domains, and typically have a reduced accuracy due to device-level noise and variability. Specifically, for analog resistors such as ReRAM, device-level variability and noise degrade the accuracy of neural network inference. The impact of this variability depends heavily on the workload being run.

In a crossbar array architecture, it is most common to implement a transistor in series with each ReRAM cell as a ”select” switch for the memory-cell or bit-cell. This crossbar array configuration is known as 1-transistor 1-resistor (1T1R). These arrays can be integrated onto an application specific integrated circuit (ASIC) and controlled by test equipment or specified interfacing hardware. In the case of mixed digital/analog interfacing hardware, digital to analog converters (DACs) can be used to provide input voltages and analog to digital converters (ADCs) can be used to convert the result into a digital format.

1.2 Background

ReRAM crossbar arrays have been explored in the past. Most previous work consists of neural network inference simulations. However, more recently, some groups have begun experimenting with this technology in hardware. Below a few of the most notable implementations of the crossbar array are discussed.

1.2.1 'In-Memory Computing with Memristor Arrays' [12]

In a 2018 paper titled 'In-Memory Computing with Memristor Arrays,' Li et al introduces an integrated circuit (IC) containing a 1T1R array of 128x64 Ta/HfO2 ReRAM devices [12]. A figure illustrating a simple 1T1R cell from Li's paper is shown in Figure 1.2. A PCB, probe card, and MATLAB programming are used to

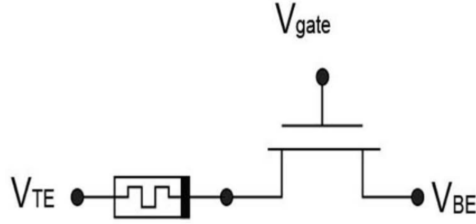

Figure 1.2: Li et al's 1T1R cell illustration [12]

interface with their chip. They implement transimpedance amplifiers (TIA) at the end of every column to convert the current into a voltage. These voltages can be fed back into different rows in the case of a multi-layered neural network, or used to measure the current from the column.

The paper describes a current-limited pulse programming procedure. When programming the ReRAM device starting from a lower conductance (set operation), the pulse voltage is applied to the TE and the current is limited by a pulse to the gate of the series transistor. The magnitude of the pulse voltage is strategically picked to limit a precise amount of current based on the voltage-transfer characteristics (VTC) of the transistor. With this programming technique, the series transistor limits the current through the ReRAM device and, by consequence, the resulting conductance of the device is associated with the applied gate voltage. In the case of a reset operation, the authors first reset the device past the target conductance by applying a pulse only to the BE of the device. They then set the device using the set technique previously

described. Using this programming technique allows the devices to be programmed into a target conductance range using at most only two voltage pulses. The authors do not go into details on the precision of this programming technique or the noise stability of the devices afterwards.

The authors demonstrate their ReRAM devices functioning properly as mostly-linear resistors within an operating range of 0V to 0.2V. Figure 1.3 shows a voltage sweep of all of their working devices at several different conductance levels. In addition

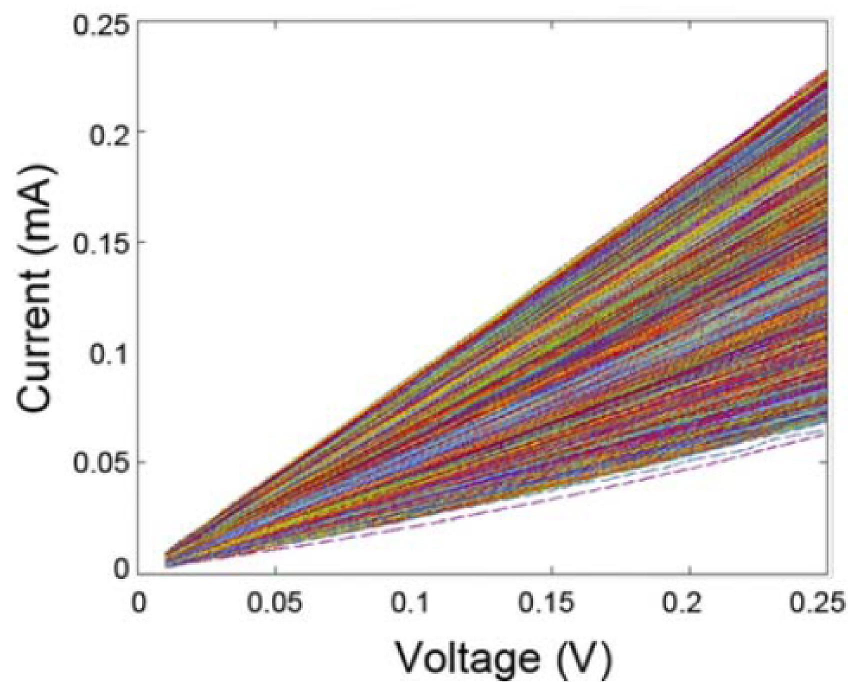

Figure 1.3: Li et al's ReRAM voltage sweep [12]

to their linear functionality, the paper demonstrates the stability of the ReRAM conductance states from 250uS to 900uS over a period of six and a half hours. This data from the paper is shown in Figure 1.4. Note, there are many conductance levels shown in this single plot so it is difficult to visually detect when the conductance levels overlap with each other. Hence, the figure does not portray any information on the noise of these devices.

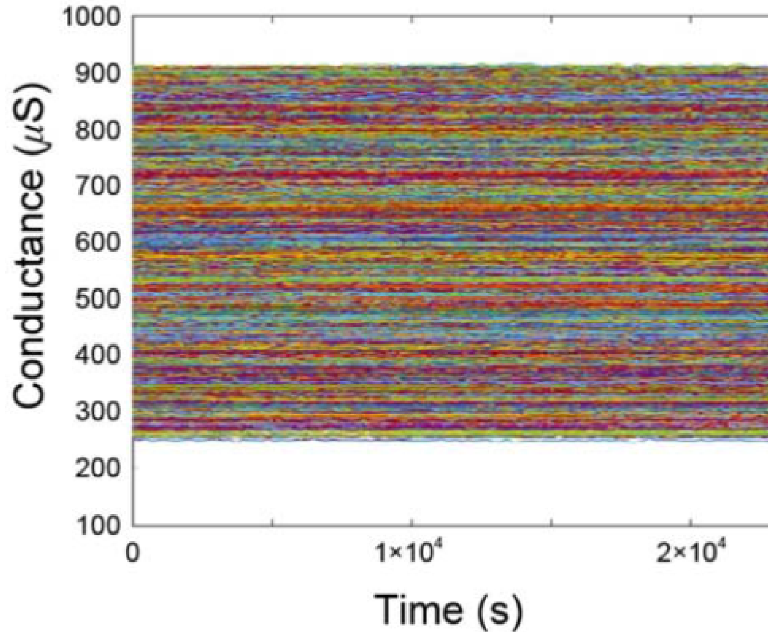

Figure 1.4: Li et al's ReRAM conductance vs time measurement [12]

The main goal of their paper was to use their chip to demonstrate in-situ training in a ReRAM array. The authors trained their array on the Modified National Institute of Standards and Technology (MNIST) dataset. Their model is implemented using a two-layer neural network with positive/negative differential weights. After training, their model placed MNIST samples with 91.71% accuracy (roughly 2.4% from ideal). They demonstrate this data by plotting the ideal and experimental accuracy versus number of training samples (shown in Figure 1.5). Their experimental results fluctuate between 2%-4% from ideal.

This paper has successfully implemented an image processing ML algorithm onto a IC developed with the analog ReRAM crossbar array. The authors have demonstrated the effectiveness of analog ReRAM crossbar arrays for ML, achieving accuracy within almost 2% from ideal. However, one key aspect to in-situ training, as is used in this paper, is that non-idealities of the ReRAM and the array structure are automatically diminished during training. Realistically, chips such as these should be able

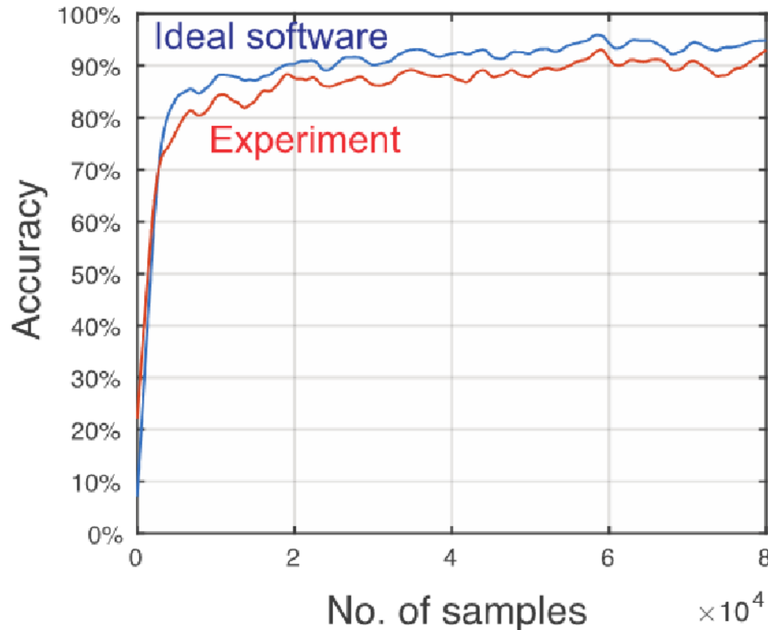

Figure 1.5: Li et al's neural network classification results [12]

to have weights from already trained networks programmed into the ReRAM cells for inference. To achieve this, further research must be done on the non-idealities of ReRAM and issues with various implementations of the crossbar array architecture. Further, the authors mention an 11% failure rate of their ReRAM device which should also be improved upon.

1.2.2 'Reinforcement learning with analogue memristor arrays' [13]

The previous paper has shown the feasibility of ReRAM crossbar arrays for in-situ supervised learning image processing scenarios [12]. In this subsection, we delve into a 2019 paper, authored by Zhongrui Wang et al, that adopts a novel approach to ReRAM crossbars by employing in-situ reinforcement learning (RL) [13]. RL is a branch of ML that allows a model to learn by interacting with an environment and receiving differently weighted rewards for making good or bad decisions. It is commonly used to train neural networks to play games, such as chess.

For this demonstration, the authors use an 128x64 1T1R ReRAM array and they interface with the array by generating signals off-chip. The array is partitioned into three distinct sections to allocate their network layers. The authors illustration of the array and layer allocation is shown in Figure 1.6. For programming, the authors use

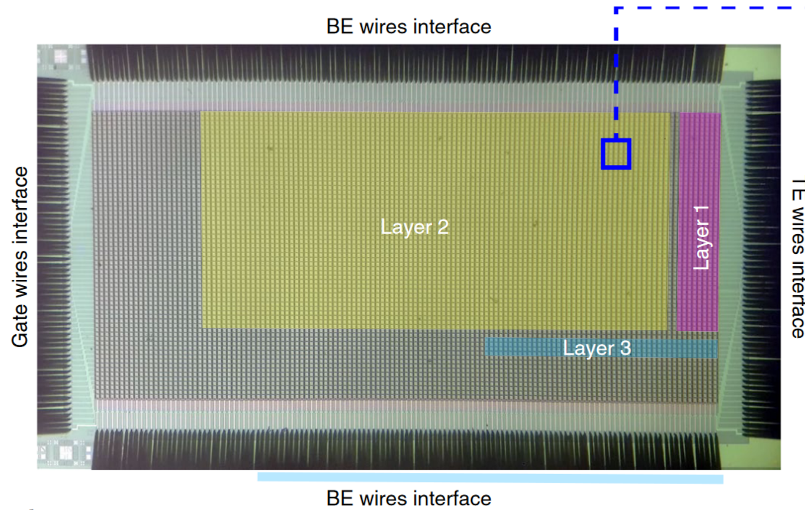

Figure 1.6: Wang et al's 128x64 1T1R layer allocation [13]

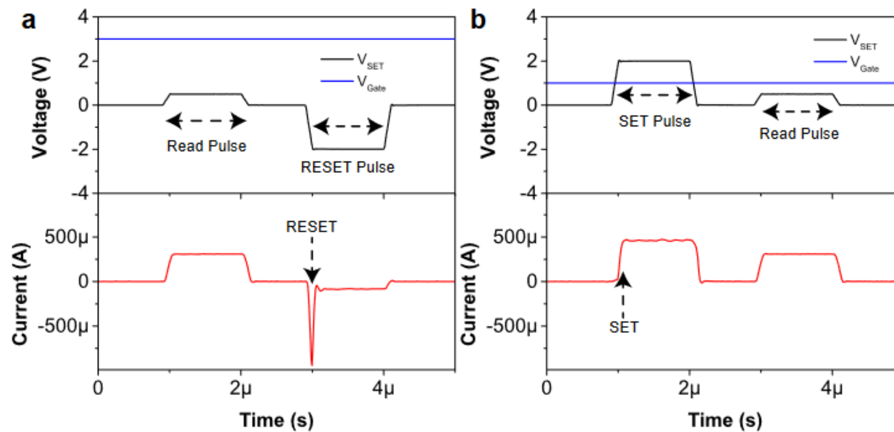

Figure 1.7: Wang et al's ReRAM programming procedure [13]

the same current limiting technique as described in the previously discussed paper [12]. Wang et al demonstrate this two pulse procedure in their supplementary information and it is illustrated here in Figure 1.7. The waveform on the left shows a reset pulse in preparation for a set. They hold a strong gate voltage of 3V for the reset pulse and

perform the actual pulse by applying -2.5V on the TE of the 1T1R cell. The current measurement of the waveform verifies the conductance has been reduced to near zero. The waveform on the right demonstrates the set pulse which occurs directly following the reset pulse. The conductance is determined by the applied gate voltage. In this particular case, the authors use 1V for the gate voltage. When the 2V set pulse is applied to the TE, the current through the cell jumps up, implying the conductance has been increased.

The authors experiment with two separate common RL problems: the cart pole game and the mountain car game. The goal of the cart pole game is to keep a pole balanced in a cart by moving the cart back and forth as the pole begins to sway. The game ends when the pole falls over. In the case of RL, for each state of the game, the neural network outputs a control: either right or left. The inputs to the neural network are the single-dimensional position of the cart, the velocity of the cart, the angle of the pole, and the angular velocity of the pole. In the mountain car game, the goal is to drive up a large mountain by driving the car back and forth between two mountains to build up momentum. The control outputs from the RL neural network are again either left or right for this game and the inputs to the network are the single-dimensional position and velocity of the car. A borrowed illustration of the cart pole game and the mountain car game are shown in Figure 1.8. We will only analyze the results of the cart pole game.

The author's results for the cart pole game are shown in Figure 1.9. The four curves shown in the figure represent their simulated results with 0 µS, 4 µS, and 8 µS noise applied to their model as well as the result from their actual experiment. The graph plots the number of rewards earned by the RL model versus the number of training epochs the model has undergone. For each discrete time step, their model receives a binary reward of 1 as long as the pole has not fallen past a set angle and

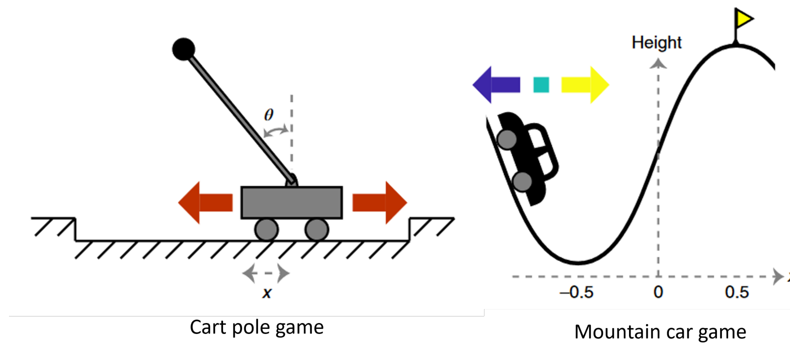

Figure 1.8: Wang et al's cart pole and mountain care games illustration [13]

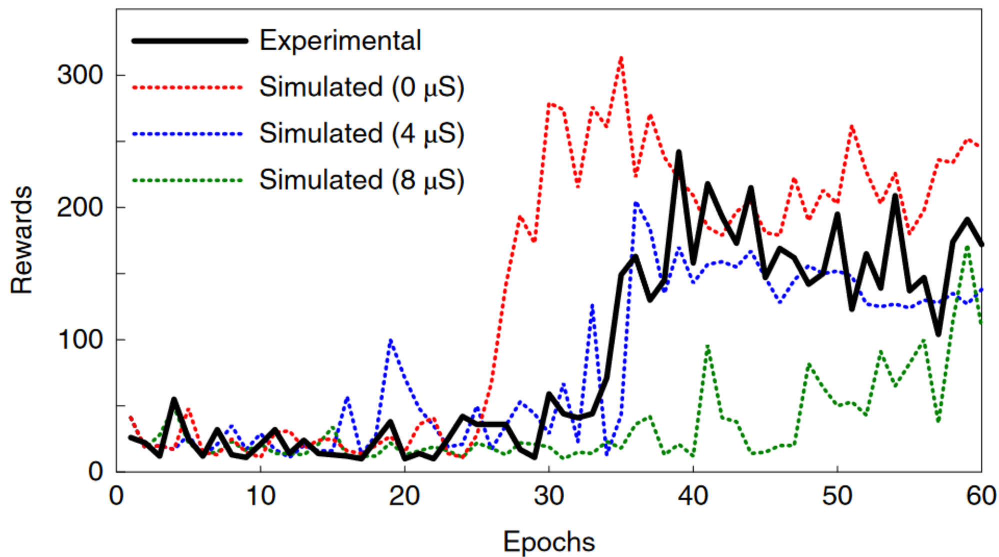

Figure 1.9: Wang et al's cart pole game results [13]

the cart has not run into the wall, and it receives a 0 otherwise. When the model is rewarded a 0, the simulation restarts from the beginning. In other words, the number of rewards earned by the model is equal to the number of discrete time steps the model successfully maintained balance of the pole. Many discrete time steps may occur in a single epoch. The red curve from the figure is their ideal curve with no noise while the blue and green curves simulate 4 μS and 8 μS of noise. The black curve shows the actual experimental results. The similarity between the blue and black curves indicate that 4 μS is a good estimation for device noise. The results

show that the model started finding success around 30 epochs.

This paper has demonstrated the possibilities of using reinforcement learning in ReRAM crossbar array circuits. The authors found success in training a network of ReRAM to play a simple game using reinforcement learning techniques. They have also provided more insight on the two-pulse ReRAM programming procedure. Like the Li et al paper, this paper is a good demonstration on the effectiveness of in-situ training in ReRAM crossbar arrays and it is unique in the sense that it was used for reinforcement learning.

1.2.3 'Thousands of conductance levels in memristors integrated on CMOS' [14]

Another key paper, published in 2023 demonstrates the ability for ReRAM to have up to 2048 discrete, non-overlapping conductance levels [14]. The authors point out that it is difficult to perform in-situ training for datasets that have been previously trained. Rather, it is much more practical for weights should be downloaded and programmed into the ReRAM array for inference. As pointed out previously, the non-idealities of ReRAM are minimized when training in-situ. Thus, when in-situ training is not used, the importance of ReRAM stability becomes critical. This paper emphasises a denoising technique used to drastically increase the stability and precision of ReRAM such that up to 2048 discrete conductance states can be realized without noise overlap.

The paper uses a 256x256 array of devices for experimental results. Figure 1.10 shows their graph of a single device programmed to 2048 different conductance levels. They illustrate this by sweeping the read voltage across the device at each different conductance level and plot them all on the same graph. Note they expanded a small portion to show none of the curves overlap at any point during the sweep. One other interesting detail the authors included in this figure is the 256x256 heat map.

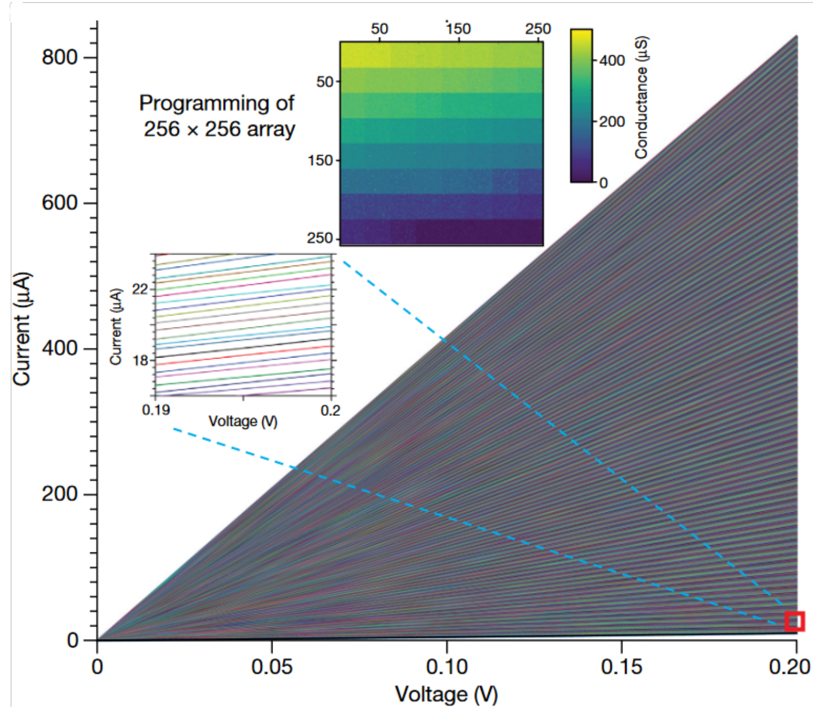

Figure 1.10: Rao et al's 2048 conductance levels sweep [14]

They split the array up into 64 separate 32x32 blocks and program each block to a different conductance level to show that their entire array can be programmed to specific conductance levels.

To achieve the level of ReRAM precision shown in the data above, they used a denoising procedure to minimize the noise of the device after each program. In Figure 1.11, they show the noise on a device before and after the denoising process. In this experiment, the authors first measured the current through a device continuously at 0.2V immediately following programming (blue). They then did the experiment again but this time added in their denoising technique before measuring the device (red). The results show a significant decrease in the noise of the device.

The authors theorize that the discrete noise levels are caused by incomplete conductance channels in the device. Occasionally, current will leak through these incomplete channels causing the current measurements to jump around. The proposed

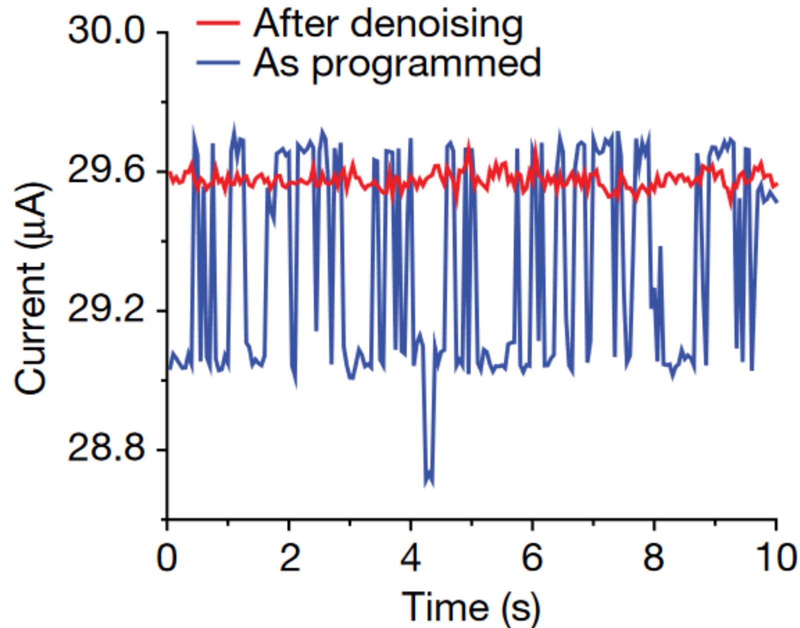

Figure 1.11: Rao et al's reads before and after denoise [14]

denoising solution is to use low magnitude biphasic pulses (one positive pulse followed by a negative pulse) to smooth out the incomplete conductance channels. However, sometimes the biphasic pulses can cause the devices to fall out of the desired target range; thus, a more complicated programming algorithm is required. The authors included a more detailed explanation on their precise programming algorithm in the supplementary information for their paper.

In Figures 1.12 and 1.13, the authors used conductive atomic force microscopy (C-AFM) to help visualize the conductance channels of a ReRAM device on a heat map. In both figures, the incomplete channel is circled. Figure 1.12 shows an example of an incomplete channel that was removed by denoising, and Figure 1.13 shows an example of an incomplete channel that was completed by denoising.

Due to device noise, there is a limit on how many discrete conductance levels can be achieved. The authors point out that, up until this point, no published results have shown more than 200 discrete, stable conductance states. This paper demonstrates

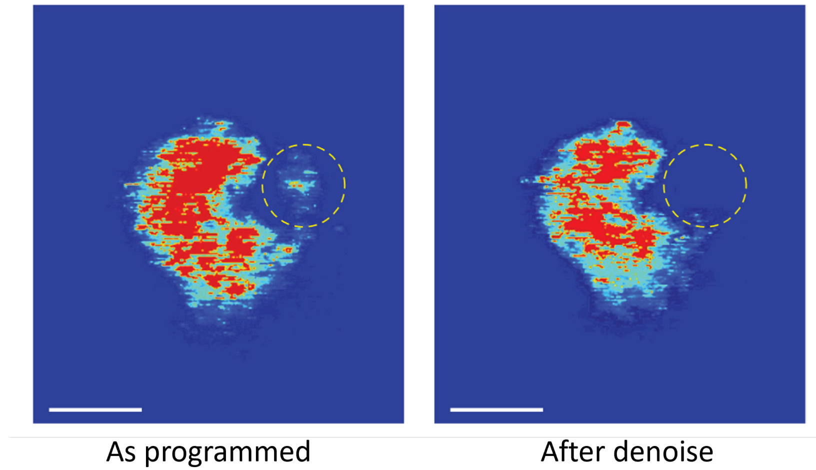

Figure 1.12: Rao et al's denoising removing incomplete channel [14]

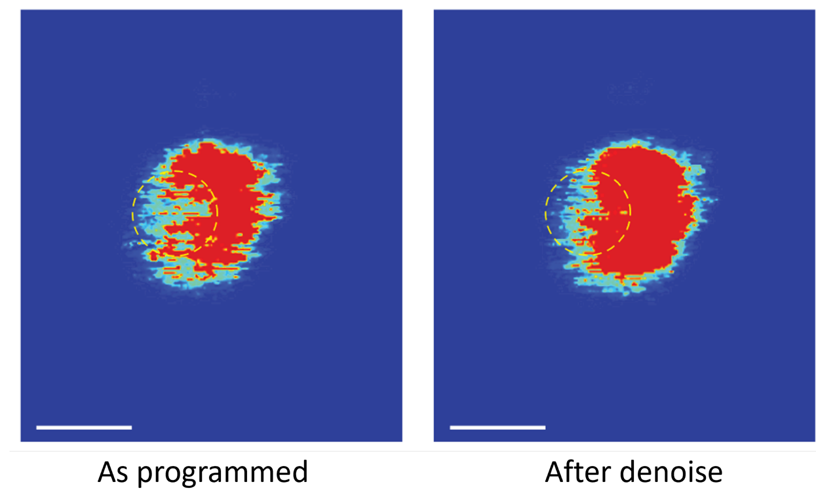

Figure 1.13: Rao et al's denoising completing incomplete channel [14]

over 2,000 conductance states are possible when a proper programming technique is implemented. These results will allow more researchers to begin exploring this denoising technique or various types of ReRAM and lead to more accurate neural network inference in analog crossbar arrays.

1.3 Need for Characterization

The recent research on ReRAM crossbar arrays has revealed the potential for a wide range of applications, including neuromorphic computing and non-volatile memory. However, as ReRAM technology advances, the need for improved characterization of both the arrays and the individual ReRAM cells becomes increasingly apparent. This need is driven by the inherent challenges in crossbar array architectures, including parasitic effects, analog-to-digital conversion issues, and the intricate interplay of nanoscale resistive switching mechanisms. These complexities are crucial to address as they directly impact the array's operational accuracy, stability, endurance, and the unique switching behaviors of individual ReRAM cells. Additionally, it is difficult to accurately predict the impact of device variability and noise on the accuracy of neural network inference based on one or a small handful of devices. Hence, it is important to comprehensively characterize a statistically significant set of ReRAM devices in order to predict the IMC system's viability.

This thesis introduces an ASIC designed to cater specifically to the rigorous demands of characterizing ReRAM at a substantial scale, with a particular focus on the intricate crossbar array architecture. This dedicated ASIC endows us with excellent control over the crossbar array, allowing us to pulse individual devices and interface with it on the digital level. This digital-level interfacing capability empowers us to conduct rapid and extensive measurements, providing a versatile and comprehensive tool for researchers and engineers as we navigate the complexities of this promising technology. Additionally, our ASIC has been designed to accommodate various types of ReRAM, enhancing its versatility and making it an even more valuable characterization tool. The results of this thesis focus specifically on an array of TaOx-based ReRAM devices.

Chapter 2

CAIRN DESIGN

2.1 Top Level Functionality

One of the key design details of the ASIC featured in this research, named Cairn, is its 2.5D architecture capable of supporting multiple types of ReRAM. With this 2.5D architecture, a control ASIC is layered on top of an interposer, which is populated with ReRAM devices. The benefit of this layered architecture is we can fabricate the ASIC and the interposer separately, and if design changes need to be made, they are only made to the layer that requires the change. Hence, Cairn enables the characterization of different types of ReRAM with the same ASIC design. The interposer size defines

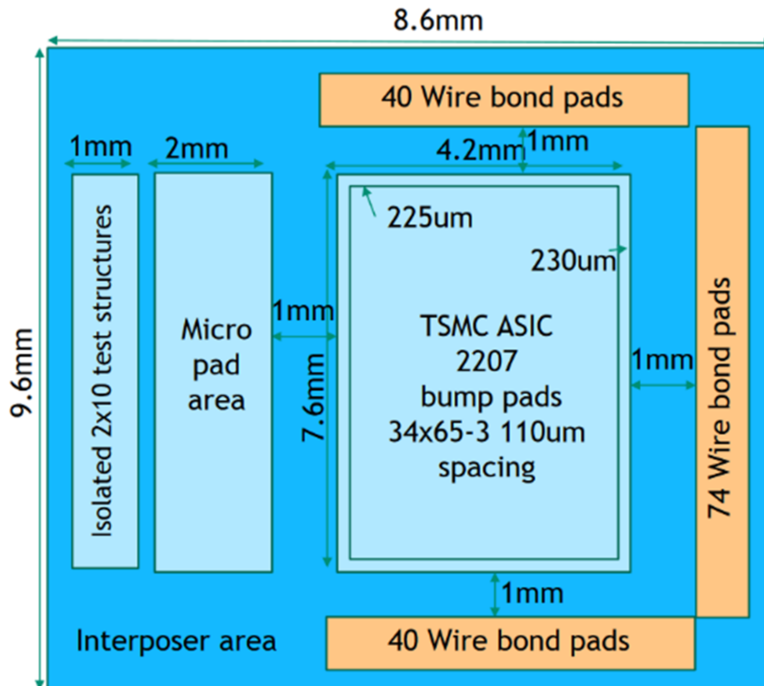

Figure 2.1: Interposer floor plan

the main chip area while the control ASIC is bump bonded on top of it. Figure 2.1 shows the floor plan for the interposer. The micro-pad area, seen in the figure, is where the ReRAM devices are deposited on the interposer. Connections are made between the ReRAM and ASIC through the bump bonds. The test structure block consists of an isolated 2x10 array of ReRAM devices for post-process testing. These devices do not have internal connections to the ASIC and cannot be tested using the interface.

Cairn contains 32x32 array-size architecture with 32 analog input pins for row input voltages. At the end of every column is a 10-bit current-based ramp ADC. The most significant bit (MSB) for every current is determined by the polarity of the current while the remaining 9 bits are the current magnitude. The ADC uses a 512-clock ramp to determine the 9 bits of magnitude. Once the current has been converted, the data for all 32 columns is stored into a 320-bit register (10 bits for each column). The user can then serially read the data out of the register for data analysis. A block diagram is shown in Figure 2.2 to illustrate how the different parts of the chip work together.

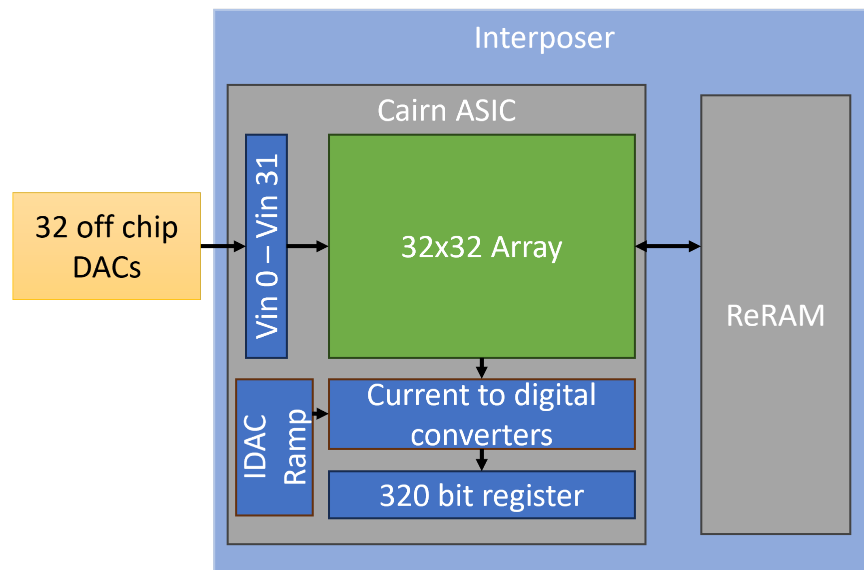

Figure 2.2: Cairn architecture block diagram

Bits	Name	Description
0-31	C[0:31]	Column selects
32-63	R[0:31]	Row selects
64	SLER	Select erase - configures bit-cell for reset
65	SLPR	Select program - configures bit-cell for set
66	SLBID	Select biphasic direction - 0: reset $\rightarrow$ set; 1: set $\rightarrow$ reset (when SLER=SLPR=1)
67	SELGT	Select gate - allows gate to be pulsed for 3-terminal devices
68	ETRGBP	Enable trigger bypass - makes pulse width equal to TRG width
69-73	DLY[0:4]	Delay - pulse width delay stages (32 selections)
74	SELVI	Select Vin - enables switches connecting Vins to array
75	ESALL	Enable shunt all - enables the shunt switches
76	SILM	Select current limit - 0: disables NMOS pullup for reset pulses (this allows for the PMOS to set a current limit)
77	EV	Enable voltage - enables the TIA to wire bond pad
78	EDR	Enable direct read - connects the column to wire bond pad
79	RD	Read - connects column connection to current-based ADC
80	STOF	Select transmission gate offset - connects Vins to VREF (for offset calibration)
81	EADCRCK	Enable ADC ramp clock - 0: manual clock; 1: internal clock
82	ENADC	Enable ADC - turns on ADC circuitry
83-85	SLI[0:2]	Select current - selects current ranging in current comparator
86-88	SLIB[0:2]	Select current bit - selects current bit division steps

Table 2.1: List of Cairn configurations

The ASIC has circuitry to support different configurations of the array. This

circuitry can be controlled by changing bits in an on-chip configuration register that is written through a serial interface. Changing the configurations allows the user to select/deselect ReRAM cells in the array and shunt the ReRAM devices. It also allows the user to configure the internal pulser to apply voltage pulses across the ReRAM devices for forming and programming of the devices. A list of the configurations can be seen in Table 2.1. Some of these circuit configurations will be explained in the following sections.

2.2 Cairn Crossbar

Figure 2.3 shows a simplified schematic view of Cairn's crossbar array. The figure was put together in such a way that it would be comparable with the crossbar array represented in Figure 1.1. Some circuitry has been omitted from this schematic to prevent over-complicating the array structure. This 32x32 structure has also been simplified down to a 3x3 structure.

Cairn is designed to work on a voltage rail of 0V to 3.3V with a mid-rail reference voltage of VREF = 1.65V. The current to digital converters at the bottom of each column have amplifiers that force the columns to VREF. This means the current through any one device is given by

$$I = (Vix - VREF) * Gx, y,$$

and the current on a column is

$$I = \sum [(Vix - VREF) * Gx, y]$$

summed over x where x is the row number and y is the column number. To multiply a given voltage by the conductance of the ReRAM device, the user can set the input voltage to 1.65V (VREF) higher than the desired multiplying voltage (Vmul) to cancel

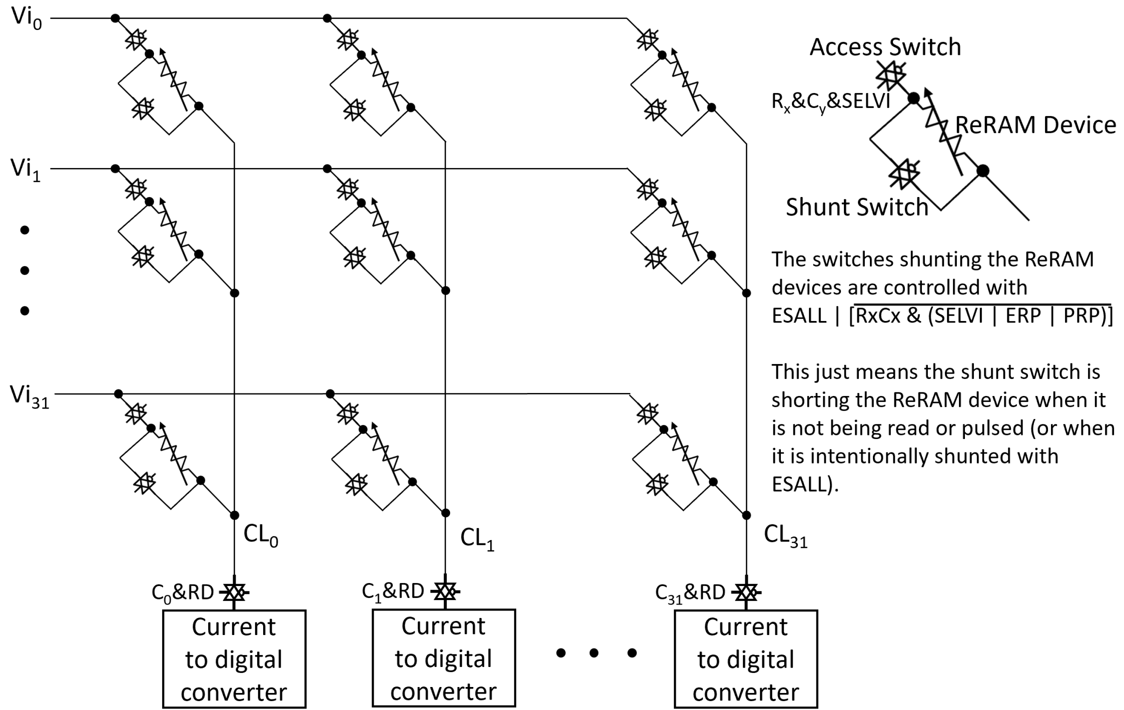

Figure 2.3: Cairn simplified crossbar array

VREF. The equation then becomes:

$$I = [(Vmul + VREF) - VREF] * Gx, y = Vmul * Gx, y.$$

The access switches shown in Figure 2.3 are used to select the devices in the array. This is useful because it allows the user to read and/or apply pulses to individual devices. It also allows the user to perform operations on a single row or column at a time. There are 32 row selects and 32 column selects in the 89 bit configuration register (see Table 2.1). The access switches are controlled with the row and column selects as well as the SELVI configuration, which can be toggled as an easy way to connect/disconnect all ReRAM devices in the array. Additionally, there is a column access switch at the bottom of every column that allows the user to disconnect the column from the current to digital converter. These switches are turned on with the column selects and the RD configuration. A good reason to toggle RD off is so the

current can be directly measured from the wire bond pad connected to the column (not shown in Figure 2.3) to test it against the digital result.

There are also shunt switches that create a low resistance path across the ReRAM devices. The shunt switch is always enabled whenever the ReRAM device is not selected to ensure there is no voltage difference across the device when it is not in use. The shunt devices can also be turned on manually with the ESALL configuration. These switches act as low resistance resistors so they are useful for testing the array when there are no ReRAM devices or when the devices are still unformed.

2.2.1 Column Access Circuitry

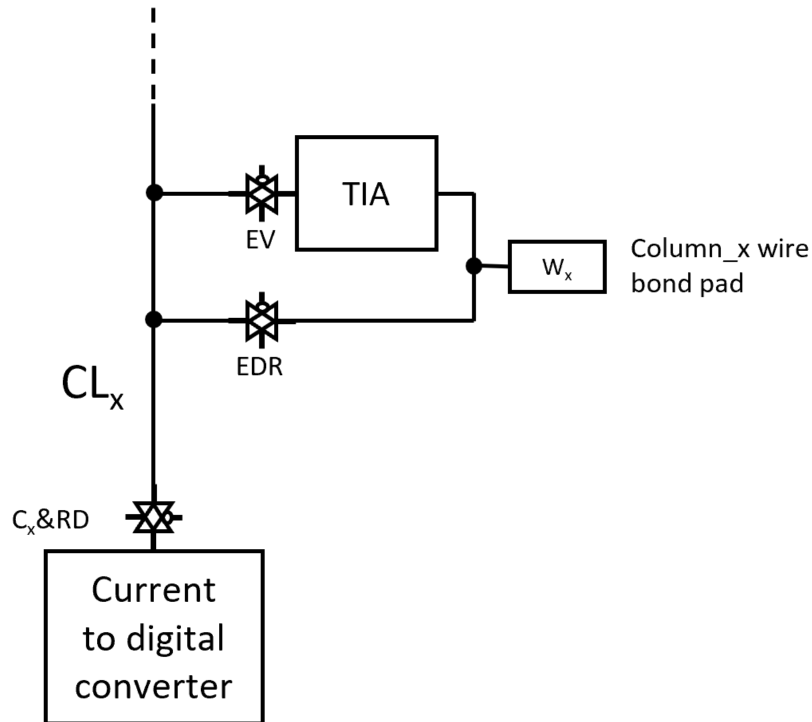

Figure 2.4: Cairn column to wire bond pad circuitry

The column can be directly measured through a wire bond pad. This feature is useful for verifying the current to digital conversion is working properly. There are two switches connecting the column to the wire bond pad. The first is enabled with

the EDR configuration and it creates a direct connection between the column and the wire bond pad. The second is enabled with EV and it creates a connection between the column and the wire bond pad through a TIA with a 50 kΩ feedback resistor so that a voltage can be measured instead of a current. Unfortunately, an error was made in the design that causes the output of the TIA to be shorted to VREF when the TIA is enabled. This means measurements cannot be made using the TIA until this problem is fixed in a future version of the chip. These connections to the wire bond pad, which are omitted from Figure 2.3, are shown in Figure 2.4. In most cases, the column switch should be turned off when making measurements on the wire bond pad to prevent the ADC circuitry from interfering with the measurement.

2.2.2 Pulser Circuitry

Also omitted from Figure 2.3 is the circuitry that allows pulses to be applied across the ReRAM devices. There are multiple switches in both the row and the column that allow the device to be forward or reverse biased (Figure 2.5). In this figure, the disconnected circuitry is shown faded so that the active circuitry can be emphasised. The switches shown in the figure can be configured to issue a set or reset pulse to make the ReRAM device more or less conductive respectively. The two separate set and reset configurations are emphasized in Figure 2.6. The documentation for Cairn refers to a set pulse as a program and a reset pulse as an erase. Every column has a switch to pull the column (TE of the device) up to a program voltage (VPP) in the case of a set operation, and a switch to pull the column down to ground in the case of a reset operation. Likewise, every bit-cell has a pair of switches to pull the BE of the ReRAM device up to an erase voltage (VEE) in the case of a reset operation, and to pull it down to ground in the case of a set operation.

SLPR and SLER are the configurations used to configure the circuit for a set or

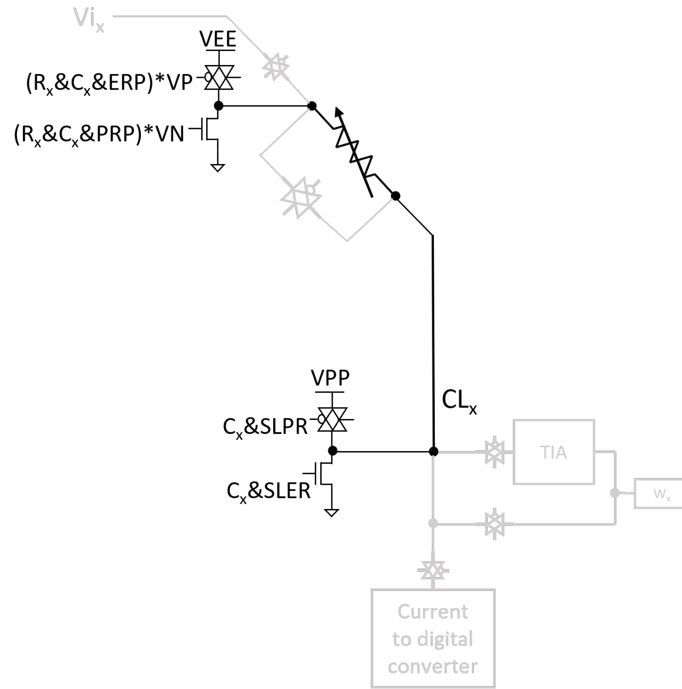

Figure 2.5: Bitcell pulser circuitry

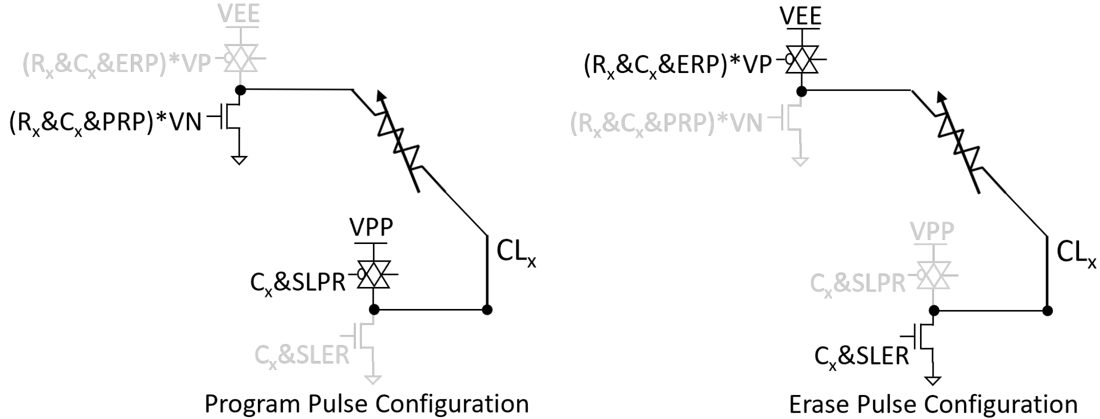

Figure 2.6: Set and reset pulse configurations

reset operation respectively. The device to be pulsed can be selected with the row and column selects. It is important to deselect SELVI to ensure there is no contention on the row side of the device, and to deselect RD, EDR, and EV to ensure there is no contention on the column side of the device. If SLPR and SLER are both set to

1, a biphasic pulse will be issued. The SLBID configuration determines whether a set or reset pulse comes first. After the circuit has been configured for a pulse, an external signal named TRG can be brought high to cause the pulse to occur. The pulse begins when PRP or ERP show up on the gate of either the pull down or pull up transistor in the bit-cell and the pulse ends when PRP/ERP go low. PRP and ERP come from an on-chip pulser which can be configured for different pulse widths. These pulse widths are determined by 32 selections from an inverter chain, which, in turn, are controlled by five DLY configuration bits. Additionally, the pulse width can also be controlled manually with TRG by setting the ETRGBP to 1. This setting makes the pulse width equal to the length of TRG. Note that ETRGBP must be 0 in order to perform a biphasic pulse.

The circuitry described above has a built-in current limiting feature to prevent excessive current from going through the device during a pulse. The current limit is determined by the gate voltage applied to the pull up and pull down transistors in the bit-cell during the duration of PRP or ERP. These gate voltages are supplied from the external sources VN and VP. Setting VN lower will cause a stricter current limit during a set pulse and setting VP higher will cause a stricter current limit during a reset pulse. VP is applied to the PMOS device in the pull up transmission gate. For current limiting to work properly, the NMOS device in the transmission gate must be turned off so that all current is limited by the PMOS transistor alone. This NMOS transistor can be turned off by setting the SILM configuration to 0 (1 turns on the NMOS device). These current limiting transistors are useful for setting a current limit while trying to form ReRAM devices to help prevent the devices from becoming over-formed during forming. Additionally, they can be used to experiment with the two-pulse programming procedure described in [12] and [13] in the future.

Chapter 3

INTERFACE DESIGN

To interface with Cairn, we are using a customized hardware setup. The setup consists of a PC, Opal Kelly FPGA, and a custom PCB with a pin grid array (PGA) socket for Cairn. An image of the physical setup is shown in Figure 3.1. The PCB connects directly to an FPGA which, in turn, is connected to a PC through a USB cable. The custom PCB is populated with DACs as a means to provide analog signal inputs to the ASIC. All of the digital signals can be fed from the FPGA directly into the ASIC through digital level shifters on the PCB.

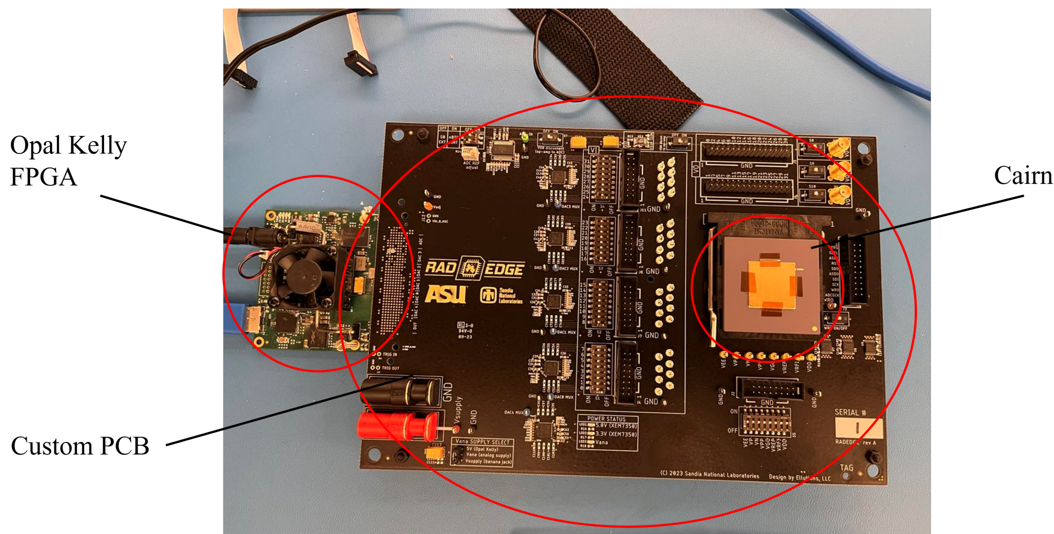

Figure 3.1: Physical interface setup

The end user can make use of all of the ASIC's functionalities from a PC interface program. All functionalities are abstracted to a higher level to make using the interface program more trivial. A diagram of the data-flow path through the hardware is illustrated in Figure 3.2. The FPGA has been configured to operate similar to a microprocessor, which entails its ability to interpret a set of instructions containing

opcodes. These instructions are processed sequentially, directing the FPGA to execute a range of predefined tasks. The FPGA's logic is designed to decode the opcodes, initiating the corresponding operations. The FPGA is programmed with two BRAM blocks, with one serving as an instruction memory (BRAM I) for the FPGA and the other (BRAM O) storing data results from Cairn, which are then transmitted back to the PC. The instructions are encoded in the interface program and simultaneously piped into the FPGA's instruction BRAM for decoding.

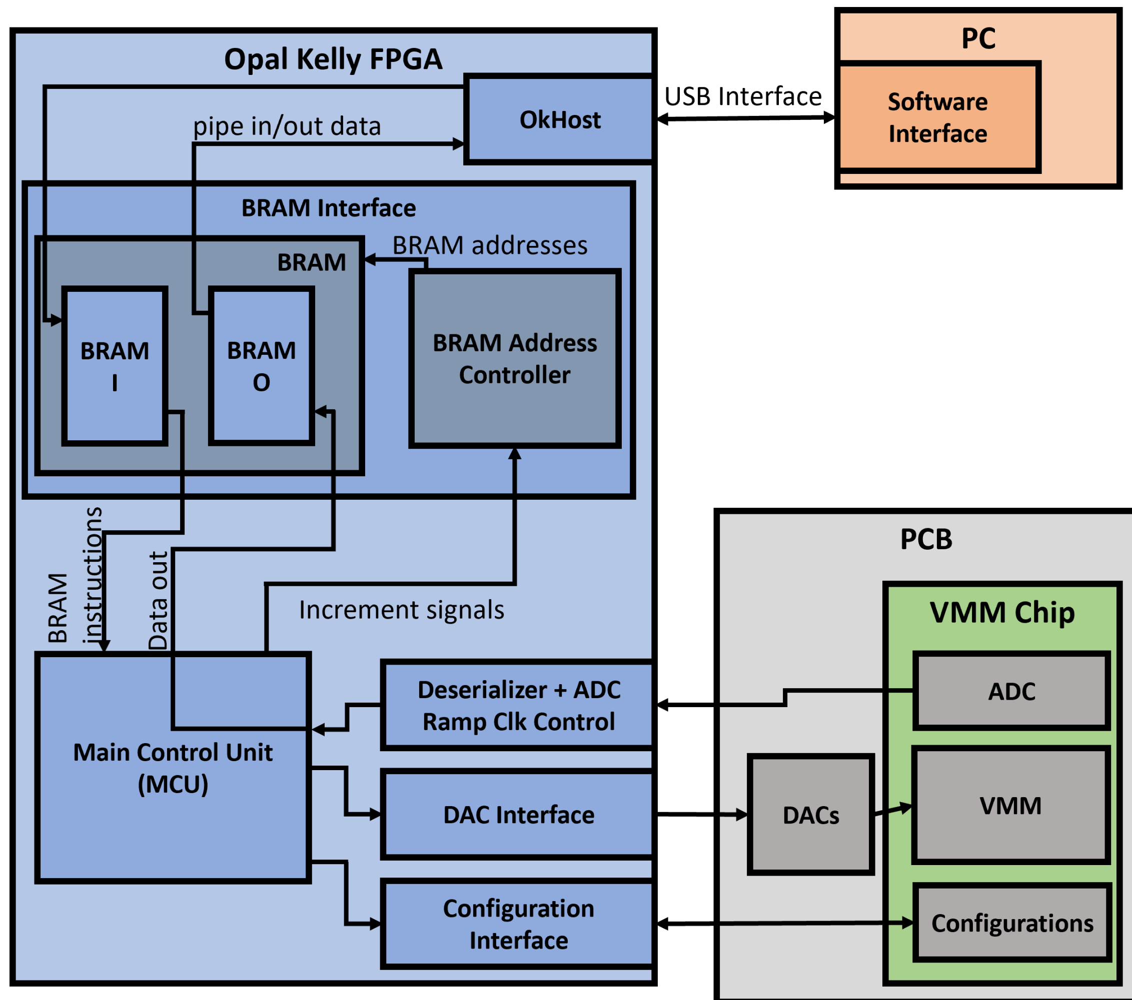

Figure 3.2: Interface design and data flow diagram

There are four key modules from Figure 3.2 critical to the functionality of this system:

- Main Control Unit (MCU).
- Configuration Interface.
- DAC Interface.
- Deserializer/ADC Ramp Clock Controller.

The MCU is tasked with decoding the instructions in BRAM I, outsourcing jobs to the other modules, and writing results into BRAM O. Each instruction is encoded with an opcode which the MCU uses to determine which job needs to be done. The instructions also contain data when the given operation requires it. The configuration interface is responsible for serially writing and reading the 89 configuration bits into the configuration register. The DAC interface serially writes the digital data to the DACs as a means to provide analog signals to the Cairn. Finally, the ADC ramp clock controller provides the ramp clock to the ADC to prepare the results from the VMM operation while the deserializer serially collects the results from the on chip ADC register.

Chapter 4

CAIRN DEBUG

4.1 Polarity Bit Error

One of the first tests performed on Cairn involved a basic voltage sweep on one of the row inputs, during which the shunt switches were activated to verify the appropriate reaction of both the analog and digital circuitry to the sweep. The raw ADC output data from this sweep is plotted in Figure 4.1. Note, the most significant bit (MSB) from each column is used to determine the polarity on the graph. The remaining bits of the ADC output code exhibit an inverse relationship with the current in the columns, indicating that as the voltage increases, the current rises while the output codes decrease. For this test, all columns were measured in parallel.

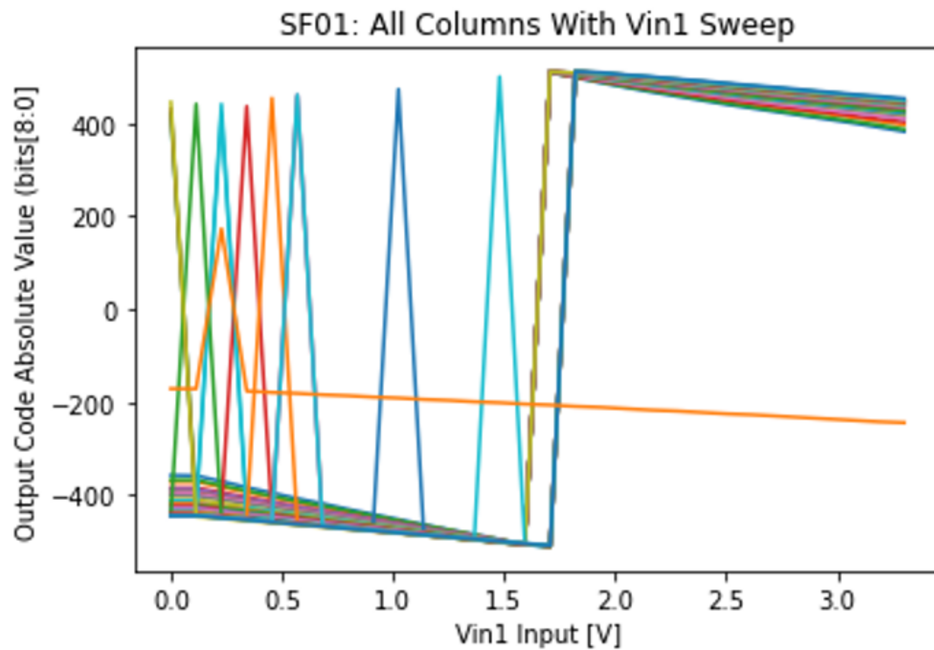

Figure 4.1: Column output codes vs Vin1 sweep

As expected, the output codes decrease as the voltage is increased until Vin1 reaches Vref (1.65V). At this point, the MSB flips, causing the codes to jump positive. However, there are several problems with this graph. The first problem is that the curves do not overlap more closely. Ideally, the shunt switches should all have the same resistance, or at least be very close in value. However, the figure indicates that the resistance increases as the column number rises. This problem will be explored in detail in section 4.2. Another problem is the column 31 curve (orange) is not following the same trend as the rest of the curves. The column 31 issue is discussed in section 4.3.

The most outstanding problem seen in Figure 4.1 is the spikes for when Vin is less than Vref. When the current on the column is negative, there is a possibility for the MSB to signify a positive current. While the MSB may be captured incorrectly, the remaining nine bits of the output code remain correct. To demonstrate this, Figure 4.2 re-plots the data ignoring the MSB. Note, this shows the absolute value of Figure 4.1.

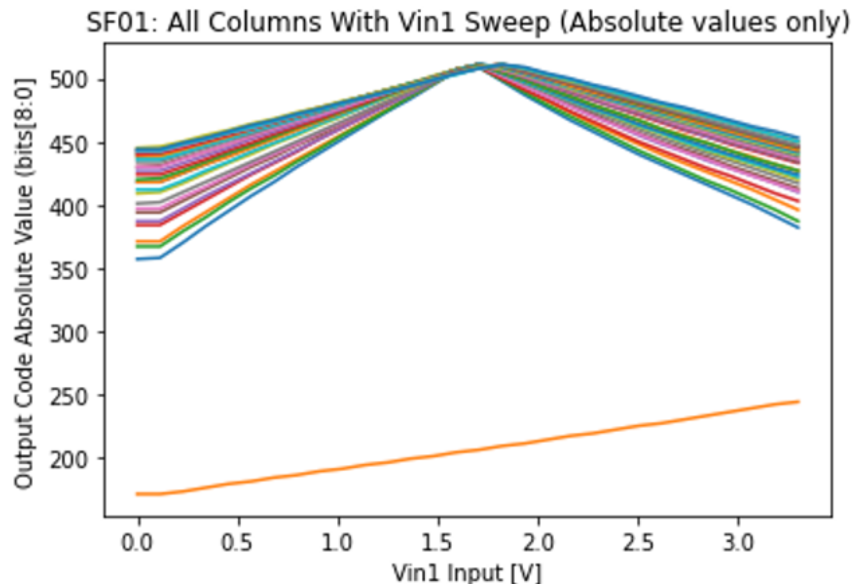

Figure 4.2: Absolute value of column output codes vs Vin1 sweep

Figure 4.3 further illustrates the polarity bit error on a heat map of currents. In this experiment, a small reverse bias is applied across the shunt switch to create a negative current on the columns. Each misread shows up as a yellow pixel in the heat map. This test was performed four times to verify the misreads were occurring randomly. The polarity bit shows up as 'one' if the current is negative and 'zero'

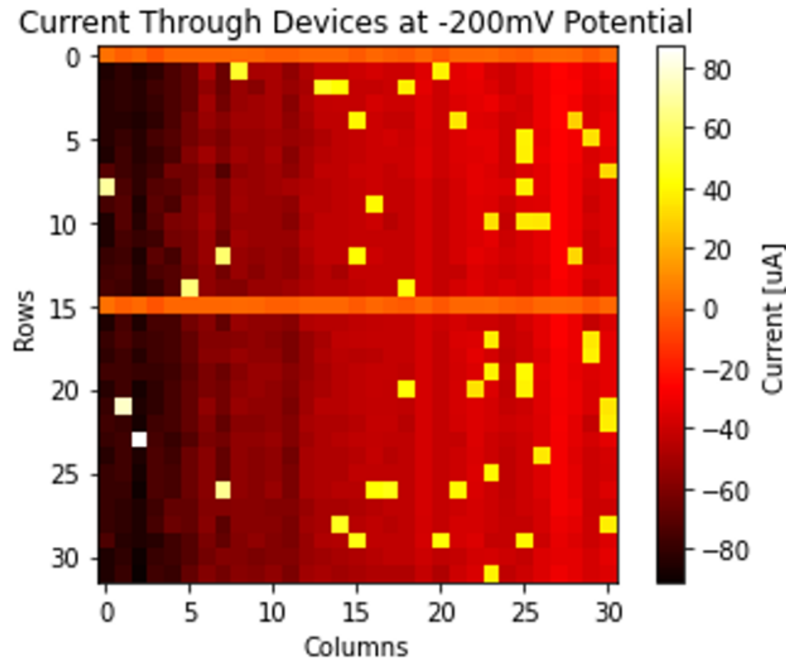

Figure 4.3: Heat map of currents to show polarity bit flip

otherwise. This shows us the data is occasionally being latched as 'zero' when it should be 'one.' The randomness in this bit flipping problem indicates a race condition is occurring when capturing the data. After inspecting simulations of the circuit, we found there was a possible race condition between the comparator that detects the current direction and the latch for the polarity bit. To remedy this issue, we've updated the circuitry to latch the data on the rising edge of the polarity bit, rather than latching it with the ADC ramp clock. This update will be implemented in future versions of Cairn. The results in this paper intentionally work only in the positive current domain to prevent this problem from skewing results.

4.2 Row Resistance Problem

As pointed out in the previous section, the results in Figure 4.1 suggest a larger than ideal resistance difference between the columns, which is better illustrated in Figure 4.4. Figure 4.4 is a heat map showing the conductances of the shunt devices across the array (excluding column 31). For these measurements, one row is read at a time while each column in the row is measured in parallel (a total of 32 reads). The figure demonstrates a gradient in the conductance along the rows. We initially assumed the extra resistance across the rows was from resistance on the row wires. However, there is about a 300 µS difference between both ends of the gradient. Assuming the conductance of a single shunt switch is the most conductive device in the heat map (450 µS), this means the row sees a total resistance of 1/(300 µS) = 3.33 kΩ. This much parasitic row resistance would be a significant problem.

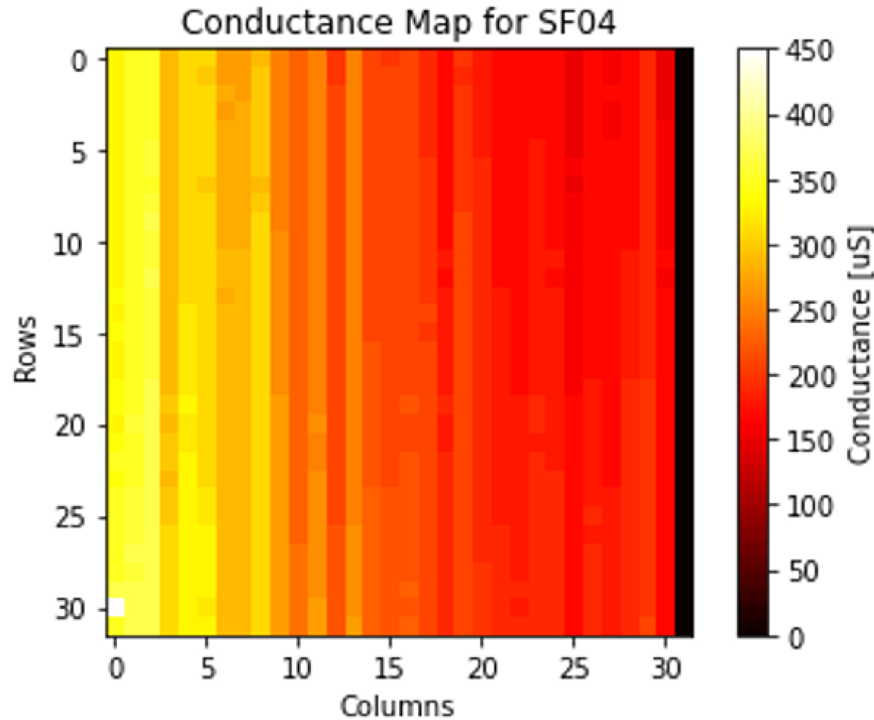

Figure 4.4: Parallel read heat map of shunt device conductances

For the purpose of troubleshooting, instead of reading each column in parallel, each shunt device was measured individually to see if the columns were somehow affecting each other. Figure 4.5 shows a heat map of each device in the array read separately from each other (1024 reads). Although there is still a very slight gradient due to row resistance, it is negligible compared to the gradient seen in the parallel read heat map from Figure 4.4. Not only is the gradient drastically improved, but the shunt device conductance level appears to be closer to 650 µS instead of 450 µS as previously assumed. This implies that reading multiple columns simultaneously affects the current readings on the columns.

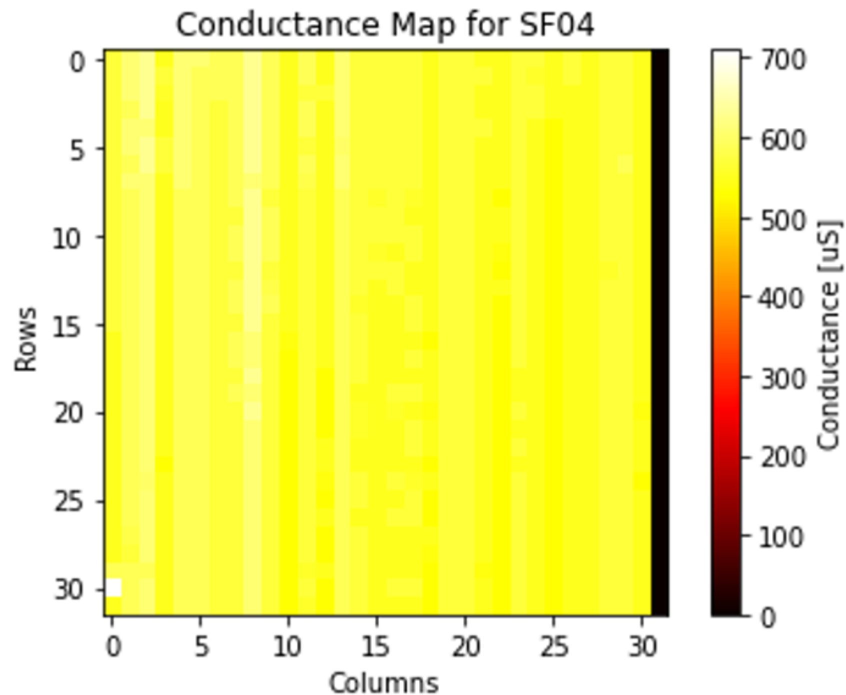

Figure 4.5: Serial read heat map of shunt device conductances

To better understand how parasitic row resistance could affect the currents differently between a single column measurement and parallel column measurements, some calculations were performed for both cases; they are shown in Figures 4.6 and 4.7 respectively. In the figures, the Rx1 and Rx2 resistors represent the shunt devices

in series with the access switches and the Rwire resistors represent the parasitic row resistance. The parasitic terms in the equations are circled in red.

Figure 4.6 corresponds to the serial read heat map in Figure 4.5. In this figure, only a single column is active meaning Rwire and Rx1 are in series and all current flowing through the row also flows to the end of the column. Ideally, the voltage across the shunt device is Vin - Vref. The reduction in current is caused by a voltage drop across Rwire. When the active column is the farthest from the Vin source, Rwire may be 32 times greater compared to the scenario where the active column is closest to the source. Thus, we still see a small gradient in Figure 4.5. For the equation in Figure 4.6, as long as Rwire is small, the effect due to parasitics is negligible. According to simulations, the wire resistance is only about 3 Ω between each column. Because there is only a minor change in current measurements across the rows, the parasitcs can be ignored for serial device reads.

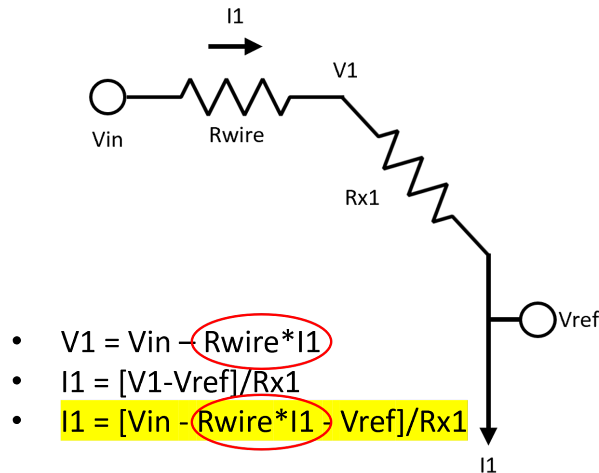

Figure 4.6: Single column with parasitic resistance calculation

Figure 4.7 shows the case of reading multiple columns in parallel. The main difference between this and the previous scenario is the amount of current running along the row. In the previous scenario, the only current through Rwire was the

individual column current. However, in this scenario, the current through the resistor between Vin and V1 is increased since Vin supplies current to both columns. The increase of current through the resistor causes a larger voltage drop to be seen at V1. Additionally, there will be another voltage drop from V1 to V2 caused by the current I2 flowing through the parasitic resistance in the next segment of wire. We can see from the equation for current as worked out in the figure that I2 is diminished by both I1 and 2*I2. If there was a third column active with current I3, then I3 would be diminished by I1, 2*I2, and 3*I3. Even with a row resistance of only 3 Ω per segment, this stacking effect causes the current to be decreased exponentially as more columns are activated (as seen when comparing Figures 4.4 and 4.5). Furthermore, the calculations to correct for the diminished currents is more complicated than the VMM operation this architecture is meant to achieve.

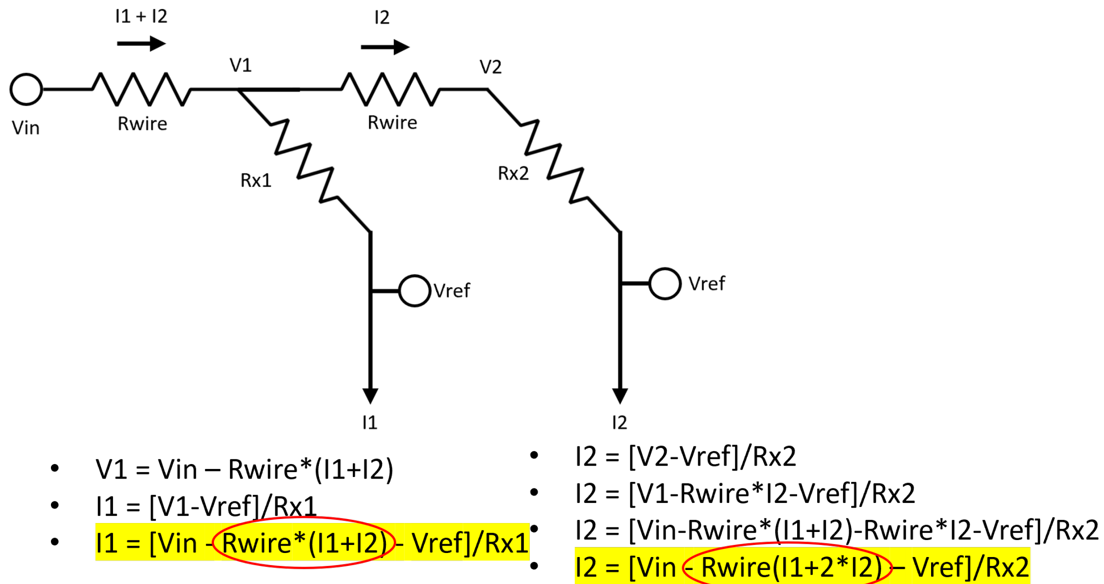

Figure 4.7: Parallel columns with parasitic resistance calculation

There are several other factors that are likely contributing to this problem. One is the resistance of the access switches and column switches which are ignored for the calculations above. Another is a built-in 15 Ω resistance on the Vin pads to help

reduce electro-static discharge (ESD). The remaining results in this paper use only a single column at a time to avoid this row resistance problem. To negate this problem in the future, further versions of Cairn will have the 15 Ω resistors removed, increased access switches by 4x, and wider row wires to reduce the row resistance.

4.3 Column 31 Issue

Previously seen in Figure 4.1 and 4.2, column 31 shows up as a large negative current throughout the entire sweep. This problem also shows up on the a heat map of shunt devices as seen in Figure 4.8. Note, in this heat map, every cell on column 31 shows up with maximum negative current for the current-range settings (512 µA), making the other columns look more uniform. Previous heat maps were obtained with column 31 disconnected from the circuit due to the extreme current difference between column 31 and the rest of the columns.

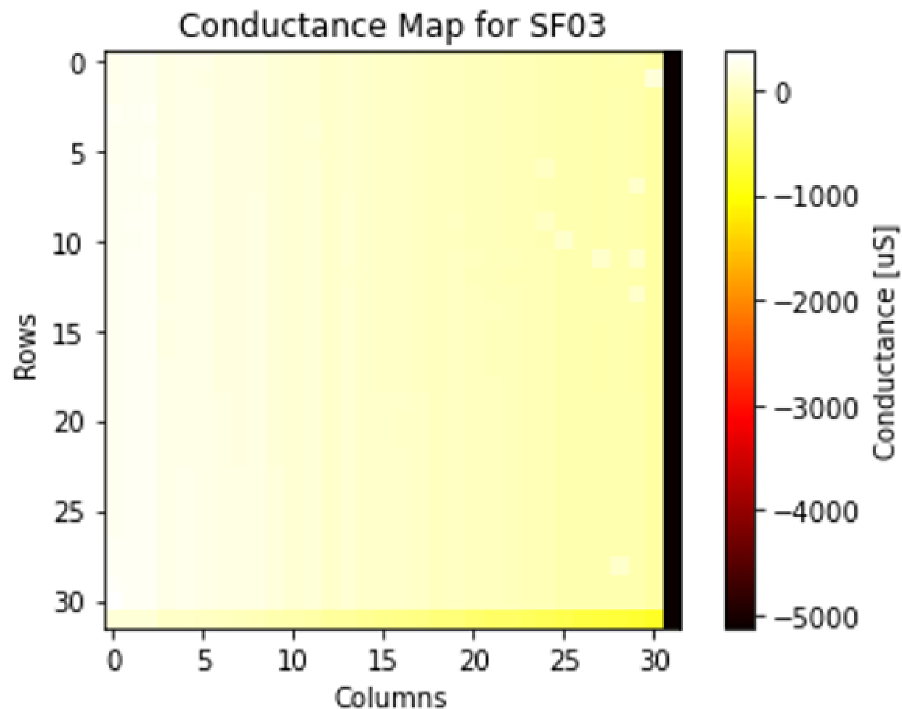

Figure 4.8: Heat map of shunt devices with column 31 active

The biases applied across the shunt devices in the figure is 100 mV. The negative current signifies either a problem with the analog to digital circuitry or a short. To test this, column 31 was connected to an external pin through a switch so that it could be measured with a Keysight B1500 Semiconductor Parameter Analyzer. The circuit configuration is shown in Figure 4.9.

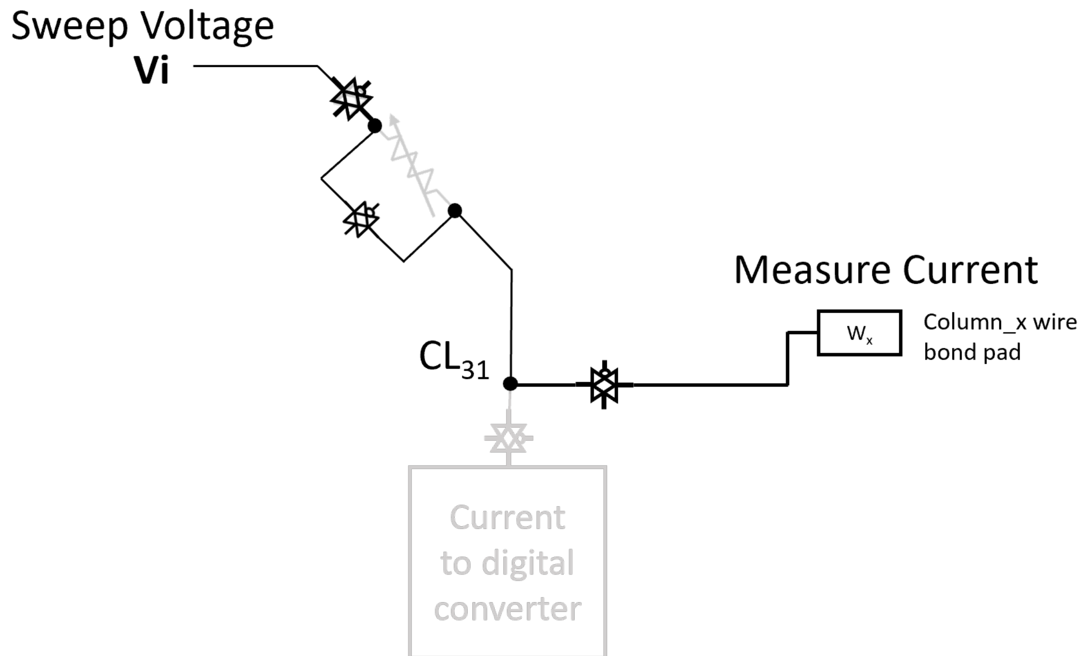

Figure 4.9: Circuit configuration for B1500 experiment

In this experiment, a voltage sweep is forced on the Vin input pin and the current is measured on both the column 31 pin and the Vin pin. This experiment disconnects the ADC circuitry and allows us to inspect the current. Ideally, the only current path is from Vin to the column 31 pin, so the current on Vin should match the measured current on the column pin. The B1500 was used to hold the column at 1.65V (VREF) for the duration of the sweep. The results from this experiment are shown in Figure 4.10. In the figure, the blue curve represents the current flowing into the column from the Vin pin, while the orange curve represents the current flowing into the column from the column pin. Ideally, these currents are equal and opposite to each

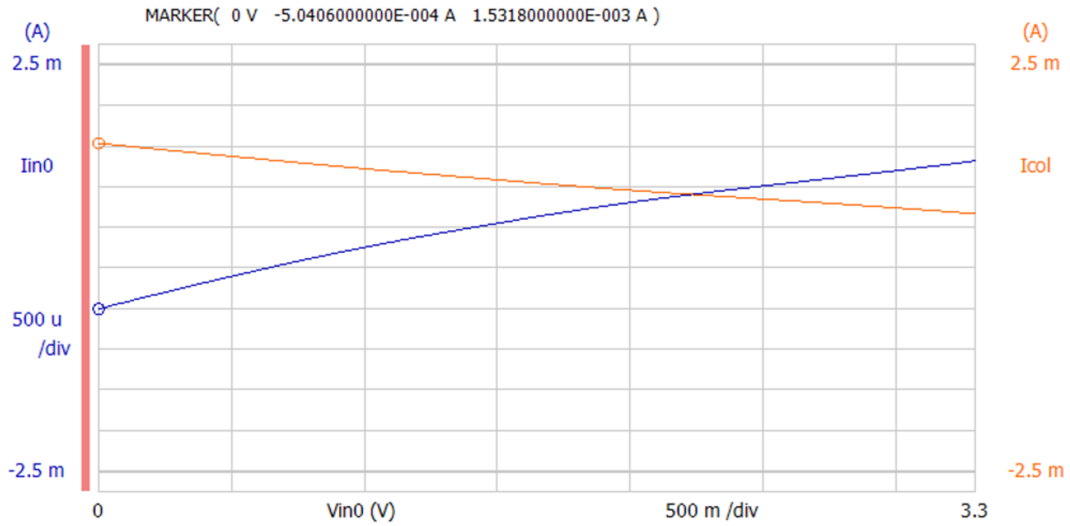

Figure 4.10: Column 31 current versus Vin0 voltage sweep

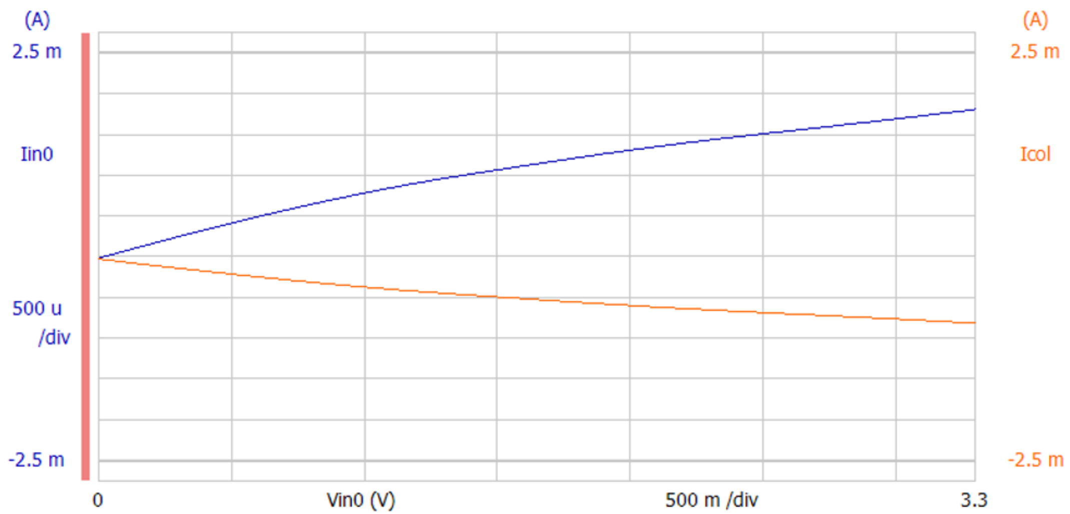

Figure 4.11: Column 31 current versus Vin0 voltage sweep with 0V forced on column

other. However, the figure shows that the curves do not match each other, implying that there is an unwanted connection somewhere on the column. Additionally, both pins are sourcing current when only one should be sourcing/sinking at a time. This implies that the unwanted connection is collecting the current and causing the column current to be higher than expected.

We can test for a short to ground by sweeping again and forcing the column to 0V. If the column is shorted to ground, the current from Vin to the column pin should be roughly equal and opposite since the column will not be fighting against the short. This experiment is shown in Figure 4.11. The results are as expected implying a short to ground.

This problem has been consistent across every Cairn prototype. The problem does not occur in simulation and the chip has under-gone extensive failure analysis to help us discover how this short may be occurring. We discovered a line of metal running underneath the column 31 bump was causing a connection between the bump and the metal and creating the short we were seeing. Fortunately, this problem only occurs on column 31 and all other columns can be used normally while we wait for this problem to be corrected in a Cairn re-spin.

Chapter 5

RESULTS

5.1 Forming & Conductance Tuning

The Cairn platform has been utilized to measure properties of candidate ReRAM arrays built from Sandia's TaOx-based process, and the quality of these devices was assessed relative to AIMC neural network workloads. When initially fabricated, most of the ReRAM devices are still unformed, which means there has not been a current channel formed between the two electrodes of the device. Forming is accomplished by applying strong voltage pulses across the device. Cairn was designed with an on-chip pulser specifically to apply typical forming and programming voltage pulse routines to the ReRAM bit, such as monophasic and biphasic pulses.

Ninomiya et al found that biphasic-formed TaOx ReRAM are more dependable than monophasic-formed devices [15]. We investigated both biphasic and monophasic pulse forming routines and found a monophasic reset pulse lasting for several microseconds was the procedure that most consistently formed these TaOx ReRAM cells. Most devices required voltage pulses approaching 5V to successfully form, which is the maximum voltage possible with this system. Forming routines are still being optimized, and the Carin characterization platform is accelerating this research.

After a device has been formed, the ReRAM bit can be programmed to a target conductance using a write-verify routine. This routine consists of iteratively applying voltage pulses across the device and reading the conductance back to verify the change. A negative pulse applied to the TE (known as a reset) is used to lower the conductance of the device and a positive pulse to the TE (known as a set) is used

to raise the conductance of the device. The required programming voltage is significantly lower than the forming voltage. The routine starts by applying relatively small magnitude pulses to the ReRAM cell. If the conductance does not reach its target after a predetermined number of write attempts, the pulse magnitude is increased. If the conductance overshoots its target, the algorithm starts over with reverse polarity. This write-verify procedure is plotted in Figure 5.1. The blue curve is the pulse voltage applied to the device and the red curve is the measured conductance. In this particular example, the device starts at a lower conductance and the write-verify algorithm writes it up to 350 µS. The pulses start at 0.1V for 10 program attempts. If they device has not reached its conductance by the end of the 10 pulses, the pulse voltage is raised by 0.1V. Two other examples are shown in Figures 5.2 and 5.3. Fig-

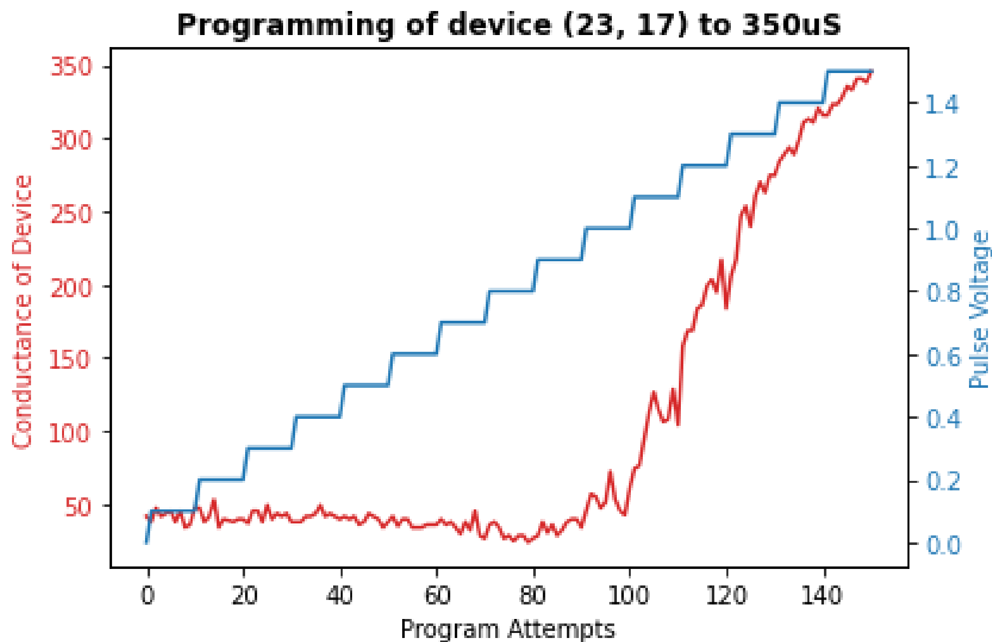

Figure 5.1: Write-verify 'set' example

ure 5.2 shows an example of a reset operation where the device starts at a higher conductance and is reset down while Figure 5.3 shows how the programming voltage changes when the conductance overshoots the target.

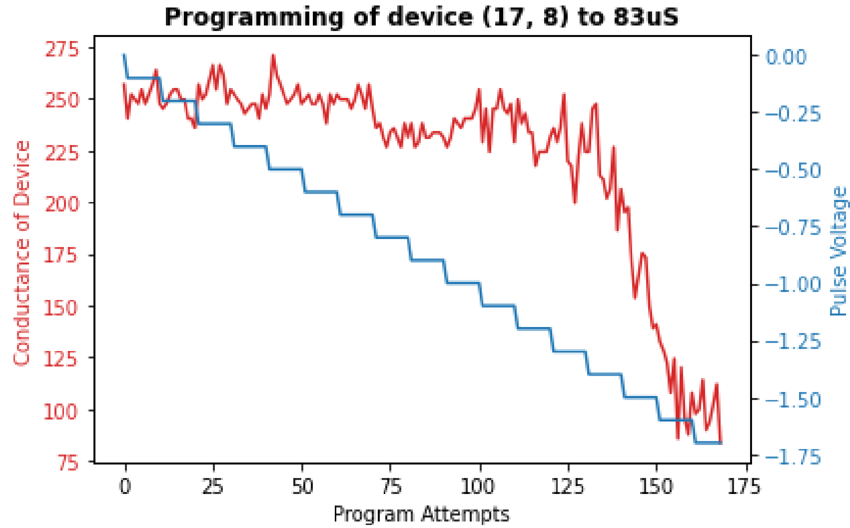

Figure 5.2: Write-verify 'reset' example

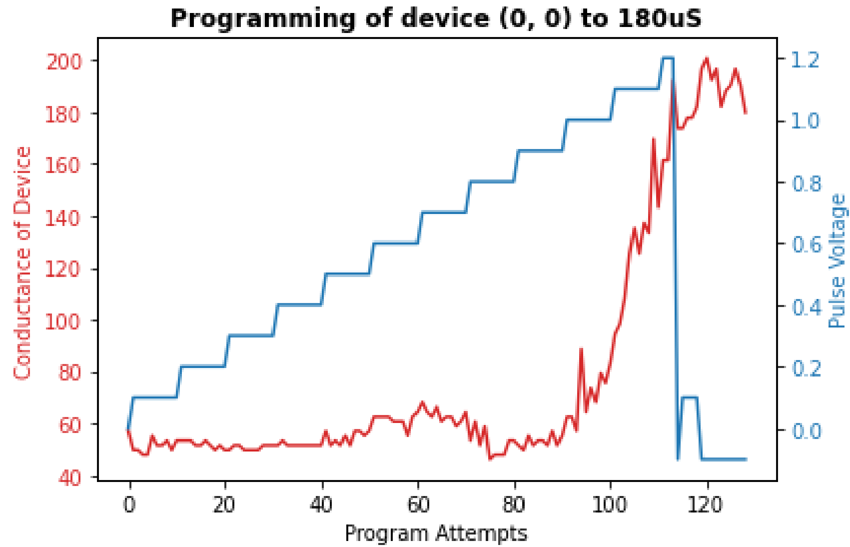

Figure 5.3: Write-verify overshoot example

This write-verify algorithm has been the exclusive method of programming throughout this research. There are several ways this algorithm could be optimized. For example, pulses under about 1V do not seem to have a large effect on the conductance of the device; so the algorithm could start with a higher pulse voltage. It is also worth trying to use the current limiting two-step programming technique explained previously [12, 13]. Cairn could theoretically accomplish this using the current limit-

ing feature built into the bit-cell pulser circuitry. These programming techniques will continue to be explored in the future.

Immediately following forming, devices may become permanently stuck at a high conductance if pulsed too aggressively or continually pulsed in the high conductance state. When a device breaks, the conductance will spike and fall back down before eventually hitting its target during a write-verify. However, once this has occurred, the device cannot be programmed back down to a lower conductance; both set and reset pulses will only cause the device to become more conductive. Figure 5.4 shows an example of the device breaking during a write-verify and then running off to a higher conductance when undergoing a reset. Fortunately, a burn-in procedure was

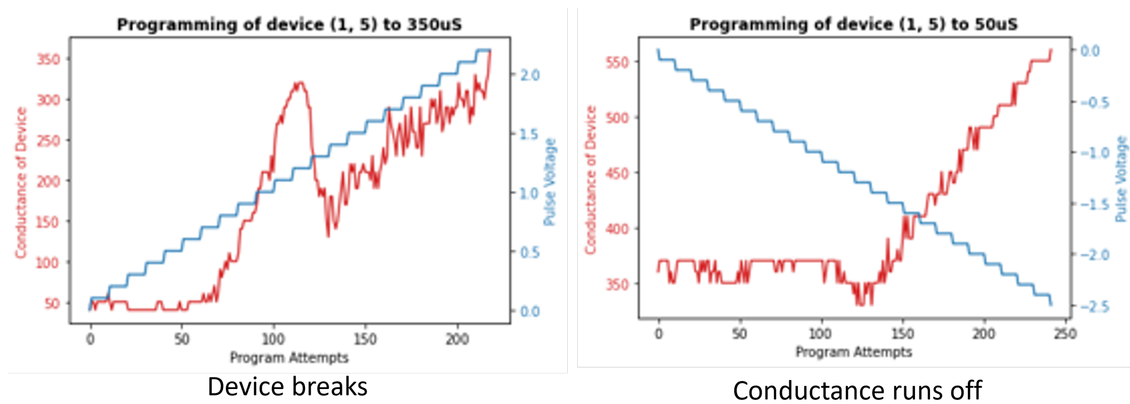

Figure 5.4: Example of breaking device

found that prevents this device from easily becoming shorted, as follows:

- Write the device to a low conductance level of 50 µS.
- Increase the conductance by about +10 µS to 60 µS.
- Repeat writing it back and forth between 50 µS and 60 µS about 10 times.
- Increase the upper bound by another +10 µS to 70 µS.
- Repeat writing it back and forth between 50 µS and 70 µS about 10 times.

- Continue this process, each time raising the upper conductance by 10 µS, until the device can be safely written all the way up to 350 µS without breaking.

This procedure enabled devices to be tuned to a high conductance immediately following forming, without shorting. Once this algorithm is complete, the device is safe to write to any conductance between 50 µs and 350 µS. This one-time burn in procedure enabled 100% of newly formed devices to function correctly following forming. We have also verified the device functions correctly following the programming routine successfully over 20,000 times.

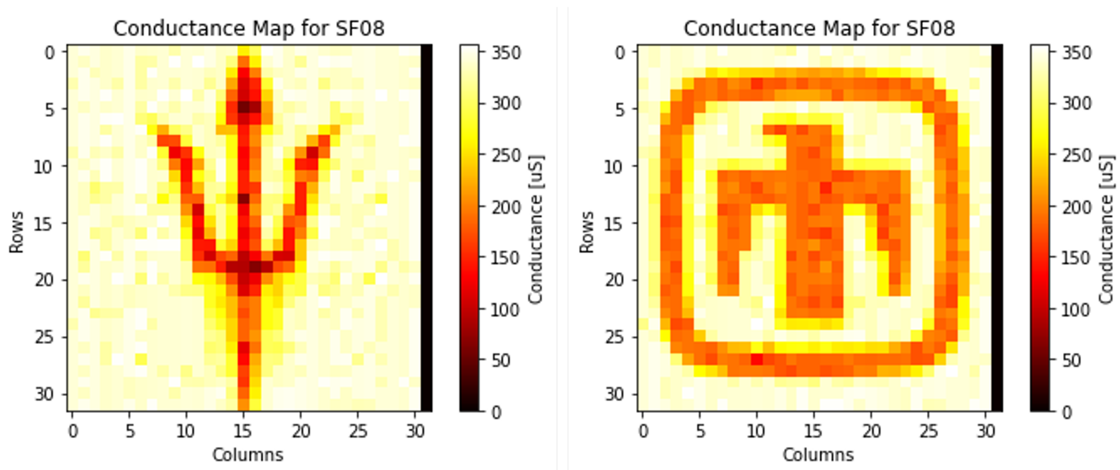

Figure 5.5: Conductance heatmap images

The forming and write-verify procedures described above have enabled the programming of the entire 32x32 array of devices to specific conductances. To illustrate this, Figure 5.5 shows a 32x32 heat map of conductances programmed on Cairn to make up the pixels of the ASU pitchfork logo and the Sandia National Laboratories Thunderbird Logo. These heat map images demonstrate Cairn's ability to effectively program ReRAM devices to a level accurate enough to visualize an image. It also demonstrates our 100% yield of working devices for forming and programming.

5.2 Drift & Noise Characterization

Ultimately, these ReRAM devices will be used to represent the weights of an AIMC neural network inference accelerator. Hence, ReRAM bits must be able to retain their memory states (or conductance levels) with minimum noise or variability. ReRAM tends to drift from its targeted conductance level over time. In order to provide a detailed understanding of how the devices drift from their initial state, we can periodically measure the conductance of select devices and plot the data. Time dependent conductance was collected by setting separate devices in the array to a different conductance levels ranging from 50 µS to 350 µS using the write-verify routine above.

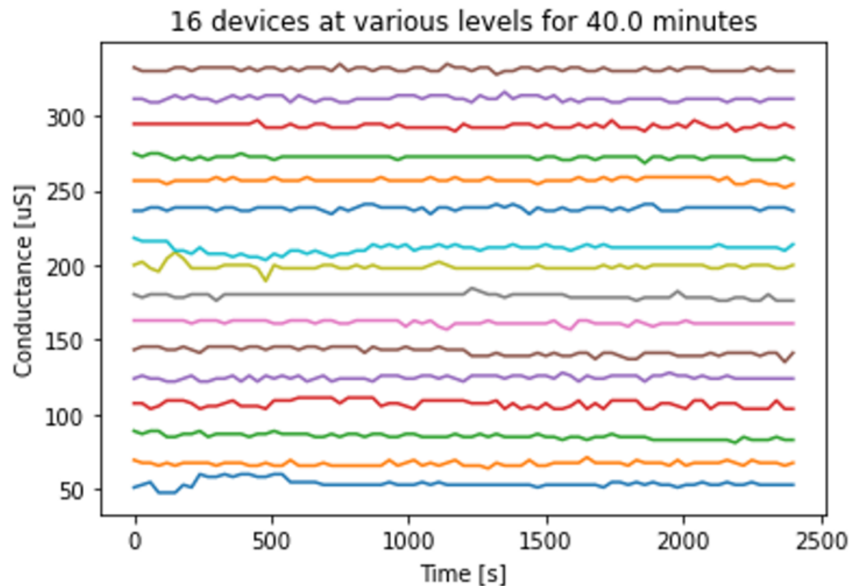

Figure 5.6: Device stability for 40 minutes measured serially

Figure 5.6 shows the drift of 16 devices each programmed to 16 different conductance levels for a total of 40 minutes. To combat the challenge of ensuring the device starts in the targeted conductance range, the devices must pass a write-verify with three consecutive verifies before drift measurements begin. Each of these three

verifies are separated by a few hundred milliseconds to allow the ReRAM device time to settle. In Figure 5.6, the devices were each individually measured for 40-minutes at a time to negate cross-talk effects from reads/writes on other devices. The figure shows the devices hold their conductance with minimal drift. However, noise may cause some of the conductance states to overlap. This noise greatly reduces the number of possible conductance states. If the ReRAM is used as digital memory, state overlap causes read errors. In the case of AIMC, this overlap tends to degrade the accuracy of the network. It may be possible to increase the number of states by pushing the dynamic range of the devices past 350 µS. The Li et al paper exemplifies this by showing their devices reach up to 900 µS [12] and the devices in the Rao et al paper are programmed as high as 4,144 µS [14]. However, as a precaution against shorting devices, our devices on Cairn have not been tested above 350 µS. More experimentation for pushing the conductance limit on our devices will take place in the future.

As mentioned previously, the measurements in Figure 5.6 are taken serially. However, it is unreasonable to use serial measurements for a large number of devices. Thus, a routine was written to overlap measurements so that multiple devices could be measured simultaneously. The routine works as such: once a device has been successfully verified to its target, the routine keeps track of what time the next measurement should be made on that device. Figure 5.7 depicts another experiment involving the same 16 devices, using the new measurement routine, this time lasting for 12 hours. In this experiment, some of the devices fall to a lower conductance shortly after measurements begin. They also tend to shift again after several hours. Note the extreme random telegraph noise (RTN) seen on the pink curve may be explained by incomplete conductance channels as discussed in the Rao et al paper [14]. While these devices may look significantly more noisy than the device shown in

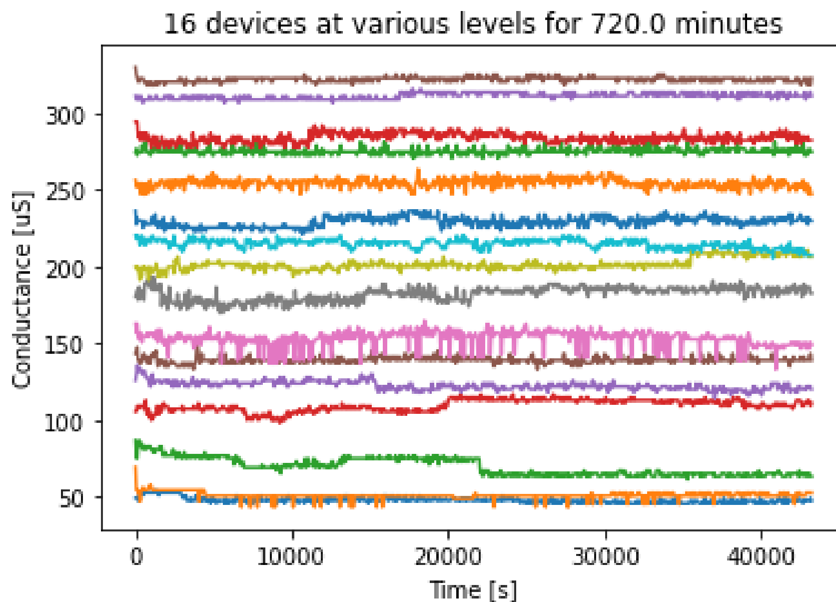

Figure 5.7: Device stability for 12 hours

Figure 5.6, these devices were measured for a much longer period of time with many more data points, which may cause them to appear more noisy. With this routine, writes and reads may affect other devices via cross talk, posing a possible issue. It is initially unclear if any of the extra noise or shifts in conductance in Figure 5.7 is due to cross-talk. While the scope of this thesis ignores the possibility of this problem in the results, it is worth investigation in the future.

For the following experiments, the conductances of the entire array were measured in 30 second increments over 300 seconds (five minutes) with each row set to a different conductance level. This data was used to measure the median change in conductance over time for four conductance levels (each with 31 devices). This data is plotted in Figure 5.8. Initially, the difference between the measured conductance and target can be considered a programming error due to the imperfect write-verify routine. The initial difference in the conductance from the target is only -0.2 µS (-0.05%), and the worst case is -0.8 µS (-0.24%). The majority of the drift occurs over the first 300s. At the end of 300s, the departure of the conductance from the target is about -2.8

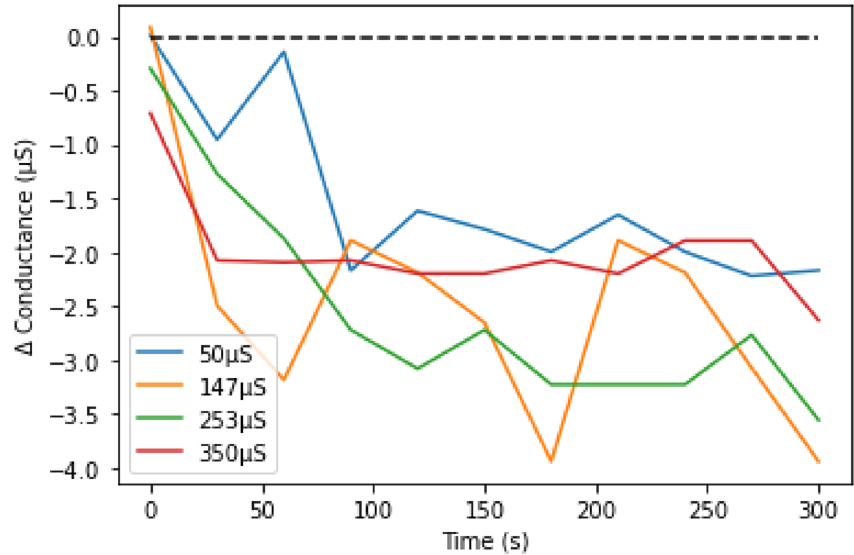

Figure 5.8: Median drift at four target conductance levels

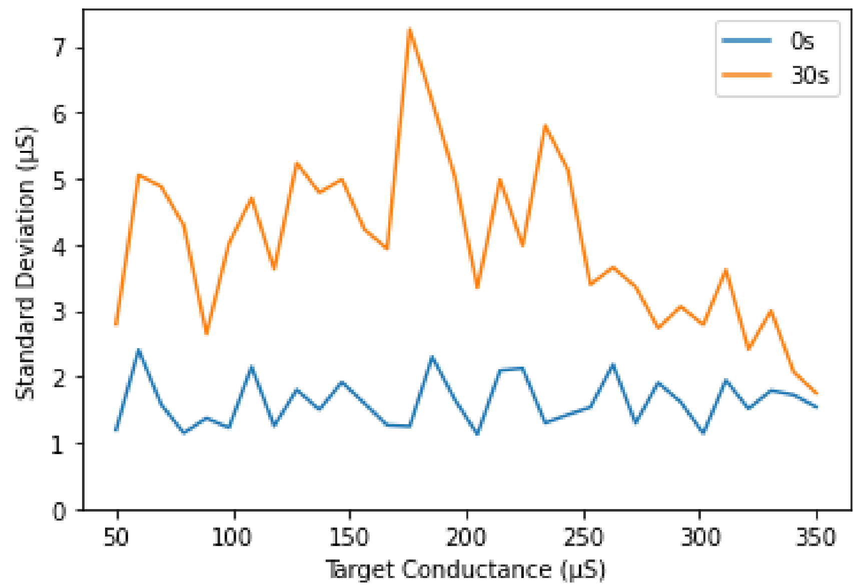

Figure 5.9: Standard deviation vs target conductance level

µS (-0.8%), and the maximum drift is about -4.2 µS (-1.2%). This level of uniform departure from the target conductance is not expected to significantly degrade the accuracy of analog IMC inference.

However, in addition to the drift, it is important to consider the evolution of the standard deviation from the target conductance over time, which can be considered

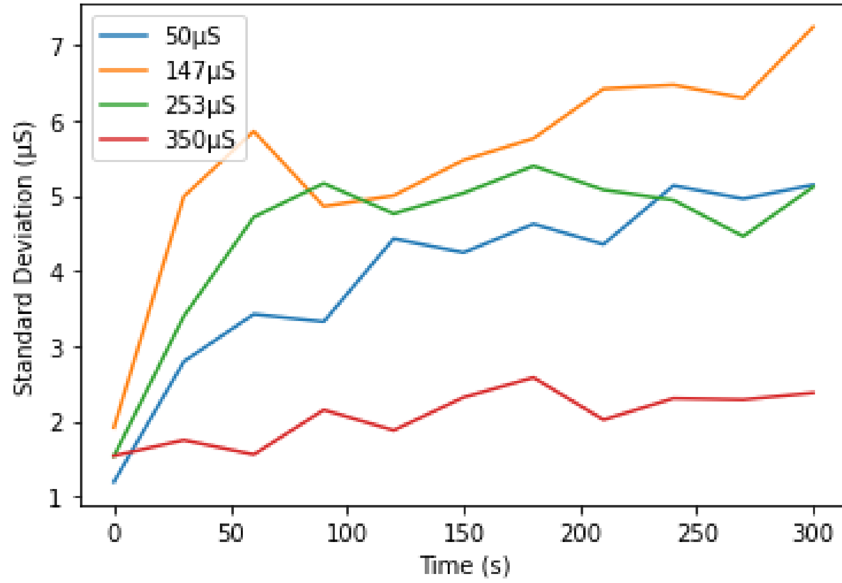

Figure 5.10: Standard deviation of conductance vs time

a measure of noise. It should be noted that standard deviation is taken for all 31 devices that were programmed to the same target level in a particular row. Figure 5.9 plots the standard deviation vs target conductance of the array at the 0 and 30 second marks and Figure 5.10 plots the standard deviation in conductance at four different target conductance levels over 300s. These results indicated that device noise increases significantly over time. The initial median standard deviation was 1.61 µS, (0.46%) and a worst case of 2.4 µS (0.69%). After 300s, median standard deviation has increased to 5.5 µS (1.58%) and a maximum of 8.1 µS (2.30%). Hence, the noise, as measured by standard deviation of conductance for a target level is worse than the drift from that target. Additionally, devices with a mid-range conductance level (100 µS to 250 µS) deviate more drastically from their targets than devices targeted at the two ends of the conductance spectrum, as illustrated by Figure 5.9. This is also indicated in Figure 5.10 by curves 147 µS and 253 µS both trending higher than the 50 µS and 350 µS curves. This data suggests the ReRAM devices are significantly more stable at high conductance levels.

An important conclusion of these results is that increase in standard deviation is higher than drift from the target conductance. This random noise is likely to be the most significant factor for degrading neural network accuracy, and the most difficult to correct for. One possible solution to minimize this noise is to implement the denoising technique discussed in the Rao et al paper [14]. As explained in chapter 1 in the background section (1.2.3), Rao et al's denoising technique involves issuing biphasic pulses after programming as an attempt to either complete or remove the incomplete conductance channels responsible for the noise. This research begins to experiment with this technique by implementing biphasic pulses into our device measurement routine. In the following experiments were taken on the array of the devices to see how they react to having a given number of biphasic pulses applied after programming. The programming procedure is as follows:

- A device is programmed to a target conductance using write-verify.
- One or more 0.35V biphasic pulses are applied to the device.
- Three reads, each separated by 0.5 seconds, are performed on the device.
- The process repeats if the device does not pass all three reads in the target conductance range.

Once the device passes all three reads, the measurements begin for that particular device. This experiment was done both with and without the biphasic pulses to see if the pulses make any kind of a difference. It should be noted that this procedure varies slightly from Rao et al's procedure. In their procedure, they use a different programming algorithm and they do multiple measurements of the conductance to measure the standard deviation to determine if a pulse is needed to decrease the standard deviation. Our procedure uses our write-verify algorithm and issues the

biphasic pulse(s) regardless. Additionally, in their procedure, they use much longer biphasic pulses which we have yet to try.

Figures 5.11, 5.12, and 5.13 show the results of the experiment described above. Each figure presents the median standard deviation over a 12-hour period at four distinct target conductance levels, with time represented on a logarithmic scale. These figures depict the results when no biphasic pulses, one biphasic pulse, and five biphasic pulses are applied after programming, respectively.

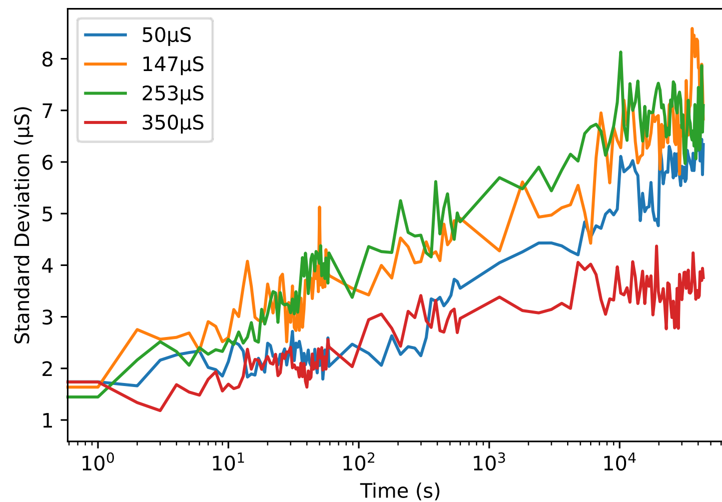

Figure 5.11: Median standard deviation for 12 hours with no biphasic pulses

Each graph shows the four curves growing with time at a similar rate. From these results, we can see that neither one nor five biphasic pulses have a strong impact on the standard deviation of the conductance. However, the pulses do seem to have an affect on the drift of the devices. Figures 5.14, 5.15, and 5.16 display the drift measurements for the same experiment described above. The drift results in Figure 5.14 follow a trend consistent with that seen in Figure 5.8. At the end of the 12 hours, the devices have drifted between 2 µS (0.57%) and 6 µS (1.71%) from the target. However, inspection of the drift after a single biphasic pulse has been applied (Figure 5.15) suggests the drift has been improved with the worst case drift being the

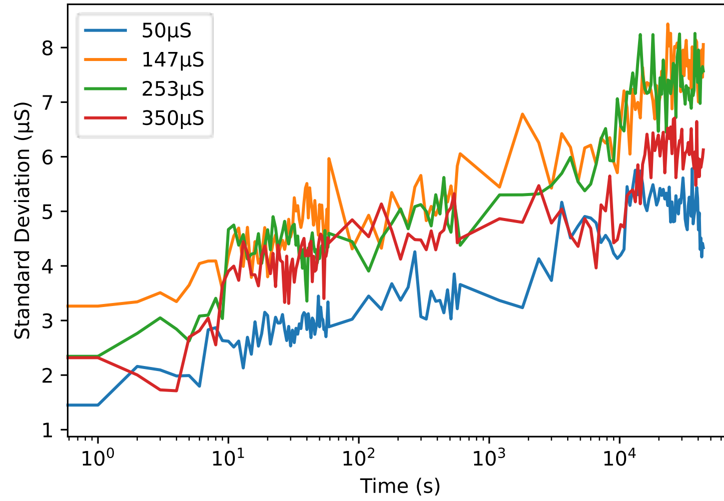

Figure 5.12: Median standard deviation for 12 hours with 1 biphasic pulse

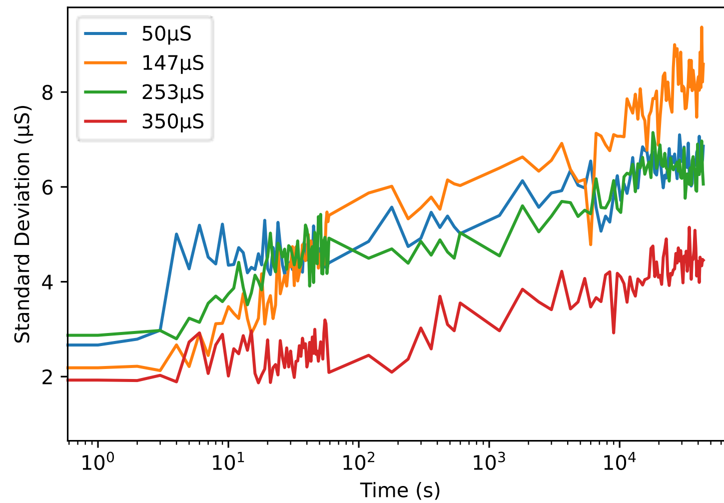

Figure 5.13: Median standard deviation for 12 hours with 5 biphasic pulses

350 µS curve with about 5 µS (1.43%) from target and the best case being the 50 µS curve with a little over 1 µS (0.29%) from the target. Additionally, the drift appears to improve even more when five biphasic pulses are applied. In Figure 5.16, the curves each stay more level and close to the target. After the 12 hours, the conductances have only drifted between about 1 µS (0.29%) and 4 µS (1.14%) from the target. This data suggests that biphasic pulses do indeed improve the drift characteristics of the ReRAM devices. However, as mentioned earlier, the main factor responsible for

degrading neural network accuracy is the noise characteristics. These results are only the initial results from the denoising experiment and there are many factors that may cause our results to vary from the Rao et al paper. Thus, there is still much work and research to do to improve these results. These results will likely improve when we adjust our denoising algorithm to follow more closely to that as seen in the Rao et al paper.

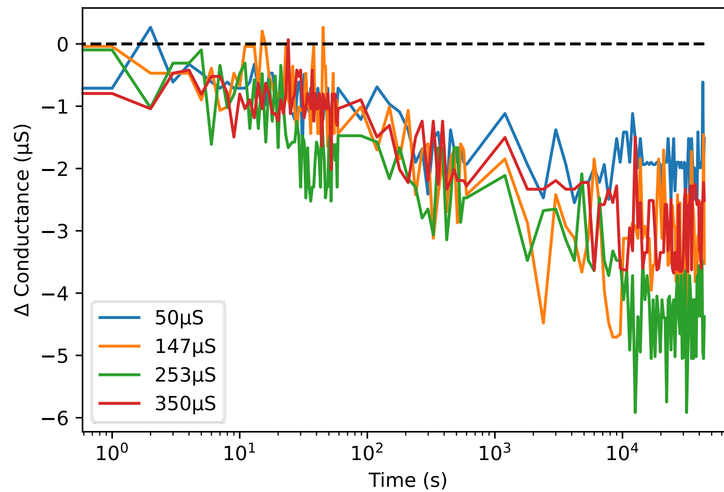

Figure 5.14: Median drift for 12 hours with no biphasic pulses

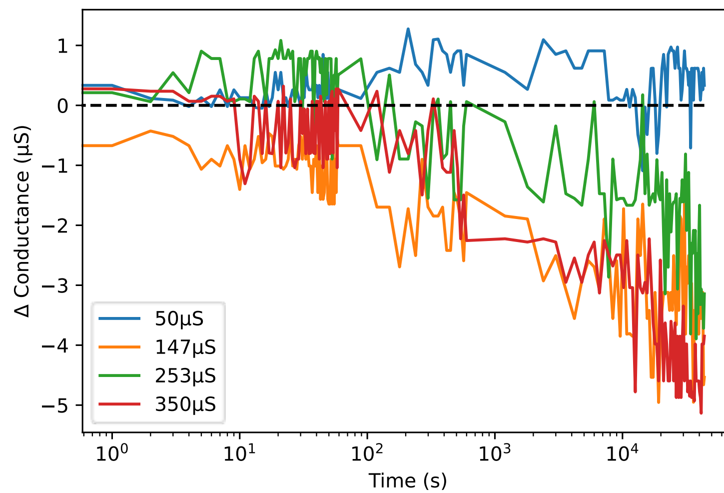

Figure 5.15: Median drift for 12 hours with 1 biphasic pulse

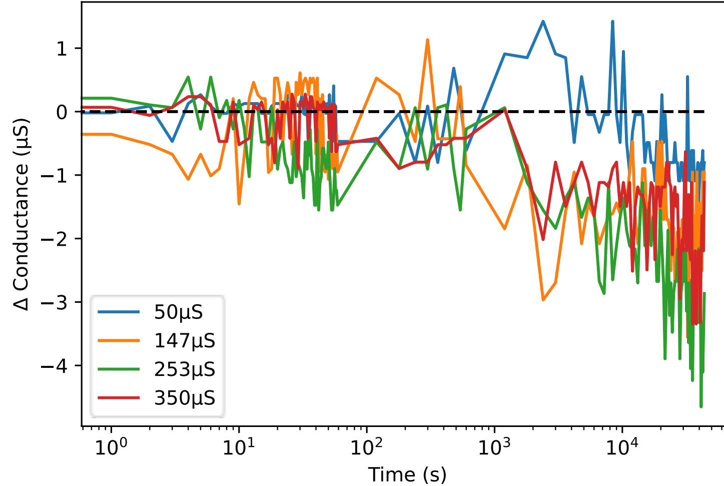

Figure 5.16: Median drift for 12 hours with 5 biphasic pulses

5.3 CIFAR-10 Accuracy Modeling

The data described above was used to model the impact of device drift and noise on inference accuracy for an IMC accelerator based on the TaOx ReRAM array. CrossSim [16] was used to model the ResNet-56 network [17] classifying 1000 images from the CIFAR-10 data set [18]. Target weights were modified using a normal distribution accounting for the device drift and variability data obtained from Cairn (Figures 5.8-5.10) as a function of time and target conductance. The accuracy as a function of time is shown in Figure 5.17. Due to the variability inherent in randomly modifying the weights, each data point was averaged over 10 runs. Additionally, the same experiment was performed with the 5-pulse denoised data modeled in CrossSim and this data is shown in Figure 5.18. The red dashed line in both figures plots a polynomial fit. For both cases, this fit shows the accuracy is beginning to stabilize following 300s.

In a digital baseline system with floating point precision, the network can accurately classify the images into the correct category with 91.5% accuracy using the same ResNet-56 model. The time zero accuracy with ReRAM and no denoise pulses was

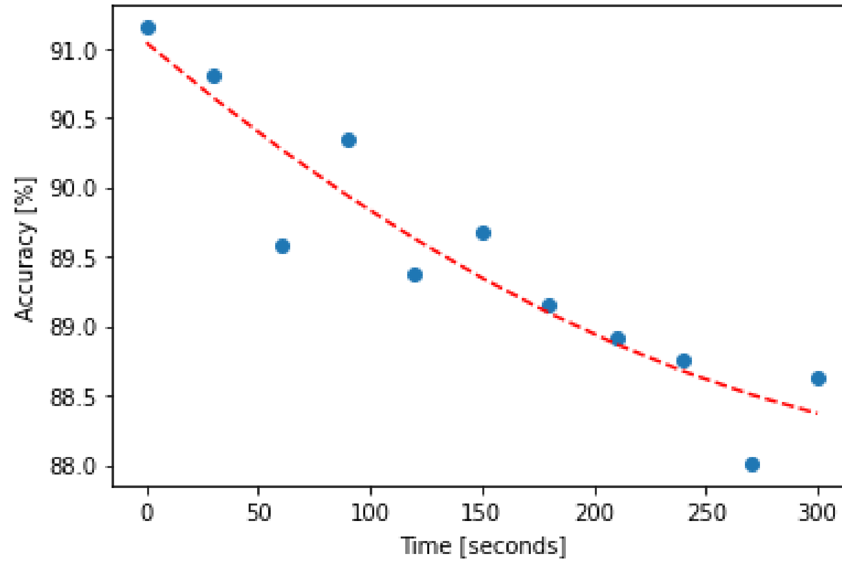

Figure 5.17: CIFAR-10 classification accuracy vs ReRAM drift time with no denoise

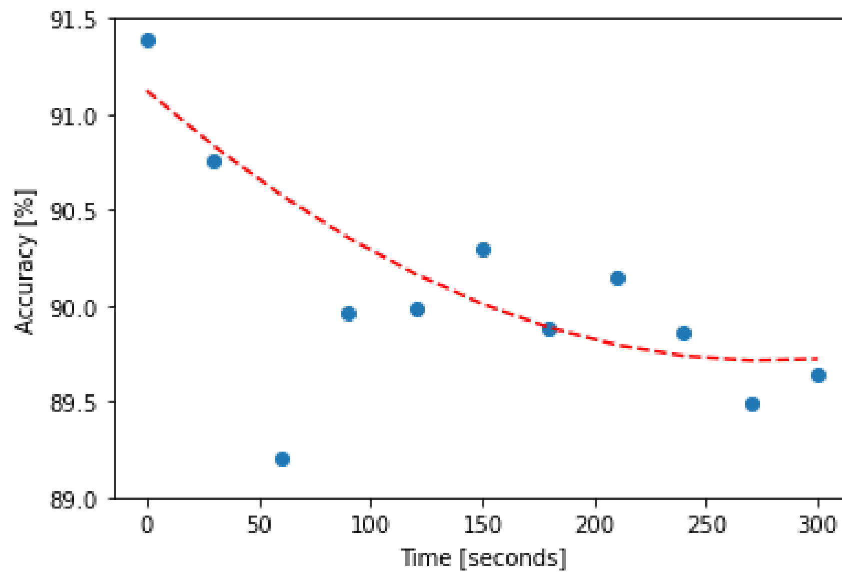

Figure 5.18: CIFAR-10 classification accuracy vs ReRAM drift time with denoise

91.2%, representing only a 0.3% departure from ideal due to programming error. This is within the bounds required for a production IMC accelerator. However, following 300s, the accuracy is degraded to 88.6%, representing a 2.9% accuracy degradation versus ideal. Additionally, in the case of denoised ReRAM, the accuracy at time zero was 91.4%, which is almost ideal. By the end of the 300s, the accuracy falls to 89.6%,

which is only a 1.9% degradation from ideal, suggesting a 1% improvement from the non-denoised data. This data is tabulated in Table 5.1.

Time	Non-denoised	Denoised	Ideal
0s	91.2%	91.4%	91.5%
300s	88.6%	89.6%	91.5%

Table 5.1: CIFAR-10 classification accuracy

From Figures 5.17 and 5.18, we can see that the denoised accuracy begins to level off sooner than non-denoised accuracy. This data suggests that the denoised pulses do, in fact, help improve the accuracy. Given our previous results, this accuracy improvement is likely reflected from the drift stability improvements. However, the majority of the degradation seen in this data is likely due to noise rather than drift. In the future, these results may be improved by improving the noise characteristics of these devices and perfecting the denoising technique described earlier.

Chapter 6

CONCLUSION

The exponential rise in energy costs caused by machine learning advancement calls for more energy efficient methods of accelerating neuromophic computing. In-memory computing using ReRAM in analog crossbar arrays offers a valuable solution to help decrease these energy costs by removing the memory-transfer requirements and improving the energy efficiency of matrix operations. The reprogrammability and non-volatile storage characteristics of ReRAM makes them an ideal tool for in-memory computing with low power. Researches have begun experimenting with in-situ learning for analog IMC accelerators and have shown that there is only a small decrease in accuracy from an ideal digital model. However, in-situ training trains on the non-idealities inherent in analog in-memory computing and thus crossbar arrays cannot be used to implement previously trained networks without retraining. It then becomes important to remove the AIMC non-idealities using methods such as the denoising technique mentioned by Rao et al [14]. In addition, the effectiveness of techniques such as these must be tested on many different types of emerging ReRAM.

As analog in-memory computing moves closer to reality, it has become increasingly important to develop large scale methods of statistically characterizing candidate memory devices. This work has introduced the Cairn 32x32 crossbar array platform, which is capable of electroforming, programming, and collecting detailed statistics for resistive memories. Cairn has been designed with a 2.5D architecture capable of supporting multiple types of ReRAM for characterization. In addition, an interfacing technique using an FPGA and custom PCB has been introduced. This interface is controlled through an intricate interface program used for abstracting low-level

functionalities to a high-level for a user.

Additionally, this thesis covers some of the debug challenges that were encountered during testing of Cairn. An issue was discovered that causes the ADC current polarity bit to occasionally result in a misread when the current is negative. This error was caused by a design flaw which resulted in the polarity bit being latched in at the wrong time. The problem only occurs when the current is negative and thus this research avoids the negative current domain. This thesis also points out skewed results caused by parasitic resistance seen in the crossbar structure as well as a problem with a short on column 31. All three of these problems are being fixed or improved upon in a revised version of the Cairn chip.

This thesis has explained and demonstrated the characterization abilities with Sandia's TiN/TaOx/Ta/TiN ReRAM cell. This research suggests the most optimal forming pulse for these devices is a reverse-biased pulse lasting for several microseconds with a close-to 5V magnitude. The platform has enabled the discovery of optimized steps for a device burn-in procedure after forming. This burn-in procedure has allowed for an 100% yield of working device. To demonstrate the reprogrammability of these ReRAM devices, a write-verify programming procedure is employed, treating the device conductances as pixels on a heat map to create an image.

The characterization potential of this platform was demonstrated by characterizing the drift and noise of the 32x32 ReRAM array and experimenting with a denoising routine. All cells in the array were set to 32 different conductance levels and measured over time. The median standard deviation and median delta drift characteristics were extracted from this data and compared against similar data that had biphasic denoise pulses applied to the cells after programming. This experiment finds that the denoise pulses provide no improvement for the standard deviation but do provide a considerable improvement of device drift. The data was then used with CrossSim to model

the accuracy of an analog IMC inference accelerator based on these ReRAM devices. Inference of this model shows only a small decrease in accuracy at time zero. Random noise rather than conductance drift over the first 300s appears to be the dominant factor degrading the inference accuracy from 91.2% (as programmed) to 88.6%. However, stabilizing the drift with the denoise pulses does result in a slight improvement of inference accuracy. For the stabilized models, the accuracy only degrades to 89.6% and begins to level off quicker than pre-stabilization (Figures 5.17 and 5.18).

In the future, Cairn will continue to be used for array and device characterization. The denoise procedure used in this research is still unoptimized. This procedure can be improved by checking the standard deviation on a device before determining if a biphasic pulse should be issued. The pulse duration and magnitude can also be optimized to better match that seen in the Rao et al paper. In addition, there is a noticeable amount of quantization noise from the ADC that adds to programming error and makes it difficult to get accurate conductance readings. The precision of a least significant bit (LSB) from the ADC can be improved by experimenting with built-in current ranging settings, thus reducing the quantization noise. There are also offsets inherent in the ADC circuitry that should be extracted and accounted for. In the future, code will be adjusted to compensate for these inherit offsets to improve read accuracy from the ADC. The two-pulse programming procedure introduced will also be attempted in the future. This two-pulse procedure allows for quick and accurate programming of ReRAM. Problems such as the polarity bit error, row resistance problem, and column 31 issue will all be fixed or improved upon in future iterations of the chip. Work will continue to improve the software and FPGA interfacing code to better optimize and allow for new testing procedures.

REFERENCES

[1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," 2019.

[2] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[3] R. Hammeed *et al.*, "Understanding sources of inefficiency in general-purpose chips," *SIGARCH Comput. Archit. News*, vol. 38, p. 37–47, jun 2010.

[4] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.

[5] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," 2016.

[6] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[7] D. Blalock and J. Guttag, "Multiplying matrices without multiplying," 2021.

[8] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "Gpus and the future of parallel computing," *IEEE Micro*, vol. 31, no. 5, pp. 7–17, 2011.

[9] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," 2017.

[10] T. P. Xiao, C. H. Bennett, B. Feinberg, S. Agarwal, and M. J. Marinella, "Analog architectures for neural network acceleration based on non-volatile memory," *Applied Physics Reviews*, vol. 7, p. 031301, 07 2020.

[11] J.-M. Hung, X. Li, J. Wu, and M.-F. Chang, "Challenges and trends indeveloping nonvolatile memory-enabled computing chips for intelligent edge devices," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1444–1453, 2020.

[12] C. Li *et al.*, "In-memory computing with memristor arrays," in *2018 IEEE International Memory Workshop (IMW)*, pp. 1–4, 2018.

[13] Z. Wang *et al.*, "Reinforcement learning with analogue memristor arrays," *Nature Electronics*, vol. 2, pp. 115–124, Mar 2019.

[14] M. Rao *et al.*, "Thousands of conductance levels in memristors integrated on cmos," *Nature*, vol. 615, pp. 823–829, Mar 2023.

[15] T. Ninomiya *et al.*, "Conductive filament scaling of tao_x bipolar reram for improving data retention under low operation current," *IEEE Transactions on Electron Devices*, vol. 60, no. 4, pp. 1384–1389, 2013.

[16] T. P. Xiao *et al.*, "Crosssim: accuracy simulation of analog in-memory computing." Available: https://github.com/sandialabs/cross-sim.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[18] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.

[19] B. Yan *et al.*, "Resistive memory-based in-memory computing: From device and large-scale integration system perspectives," *Advanced Intelligent Systems*, vol. 1, 2019.

[20] Z. Wei *et al.*, "Highly reliable taox reram and direct evidence of redox reaction mechanism," in *2008 IEEE International Electron Devices Meeting*, pp. 1–4, 2008.

[21] Y. Yang, P. Sheridan, and W. Lu, "Complementary resistive switching in tantalum oxide-based resistive memory devices," *Applied Physics Letters*, vol. 100, p. 203112, 05 2012.

[22] M. J. Marinella, "Radiation effects in advanced and emerging nonvolatile memories," *IEEE Transactions on Nuclear Science*, vol. 68, no. 5, pp. 546–572, 2021.

[23] J. Short *et al.*, "Array-scale characterization of reram arrays for analog in-memory computing," in *2023 IEEE International Conference on Rebooting Computing (ICRC)*, 2023. Forthcoming.

APPENDIX A

STATEMENT FOR 'ARRAY-SCALE CHARACTERIZATION OF RERAM ARRAYS FOR ANALOG IN-MEMORY COMPUTING'

In this thesis, I would like to acknowledge that certain portions of the text and figures included here have previously appeared in a paper, authored by me and others, titled 'Array-Scale Characterization of ReRAM Arrays for Analog In-Memory Computing,' [23] that will be officially published in *2023 IEEE International Conference on Rebooting Computing (ICRC)* in December 2023.

Certain parts of the introduction section in chapter 1 are explained using the same or similar wording as seen in the ICRC paper. Likewise, some of Cairn's functionality explained in chapter 2 also appears in the ICRC paper. Some results and wording in chapter 5 appear in the ICRC paper as well. Finally, the conclusion in chapter 6 also overlaps with the conclusion in the ICRC paper.

Figures 1.1, 2.2, 3.2, 5.1, 5.5, 5.10, 5.9, 5.17, and a simplified version of figure 2.3 all appear in the ICRC paper. These figures are in chapters 1, 2, 3, and 5.

I would also like to acknowledge that I have received permission from the co-authors to use the previously published work in this thesis.