AFRL-RI-RS-TR-2021-203

THE CONTROLLER DESIGN AND INTEGRATION OF MEMRISTOR-BASED NEUROMORPHIC SYSTEM

DUKE UNIVERSITY

DECEMBER 2021

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2021-203 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
NATHAN R. MCDONALD
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
DECEMBER 2021	FINAL TECHNICAL REPORT	JUNE 2018	SEPTEMBER 2021

4. TITLE AND SUBTITLE

THE CONTROLLER DESIGN AND INTEGRATION OF MEMRISTOR-BASED NEUROMORPHIC SYSTEM

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-18-2-0121		62788F

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
T2BS	MN	DU

6. AUTHOR(S)

Hai Li and Brady Taylor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Duke University
Electrical and Computer Engineering Department
Edmund T. Pratt Jr. School of Engineering
Box 90291
Durham NC 27708

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RITB 525 Brooks Road Rome NY 13441-4505	RI	AFRL-RI-RS-TR-2021-203

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project, we designed and fabricated a neuromorphic system controller in TSMC 65 nm technology for integration with memristive crossbar arrays. The design is capable of driving 1T1R or 1S1R arrays for implementation of spiking neuromorphic systems, using a digital pulse-width modulation-based input scheme as well as an integrate-and-fire circuit connected to a digital counter for the output scheme. Furthermore, the controller design can be tiled to drive arbitrarily large arrays and flexibly assigned to different layers in a multilayer neural network design. The chips can be integrated on a PCB or in-package and utilized by an off-chip processor or FPGA.

15. SUBJECT TERMS

Memristor, resistive memory, controller, crossbar array, neuromorphic computing, spiking neural networks.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	44

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
NATHAN R. MCDONALD	N/A

PREVIOUS EDITION IS OBSOLETE.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

1.0 SUMMARY

This project aims to accelerate artificial neural networks by virtue of programmable, low-power, and high-density memristor technology. Matrix-vector computation is the core operation that has been heavily exploited in neural network applications. Its computing speed and power consumption dominants the performance of neuromorphic hardware. In the project, we implemented a programmable neural network computing engine based on the memristor crossbar technique. The peripheral circuitry was designed and fabricated in TSMC 65nm process. The system integration of the CMOS chip and resistive arrays was realized and evaluated at the PCB board level, with typical neural network applications. FPGA board works as the holistic system controller, performing chip configuration, data pre-processing, and necessary data routings for multi-chip coordination.

2.0 INTRODUCTION

In the development history of microprocessors, the continuation of Moore's law was first guaranteed by the increase of transistor integration density and then by multi-core technology [1]. Although the computing efficiency of solid-state circuits keeps improving by following technology scaling, the data transportation (communication) efficiency between CPU cores and storage systems continues drifting down and dominating the entire system's energy consumption. This phenomenon is referred to as the "memory wall" [2]. Information is generally stored in solid-state circuits as the electrical charge on capacitors, e.g., in SRAM and registers. Data transportation is realized by moving the charge from one location to another. Thus, the communication cost is determined by two factors—the amount of the charge relocated and the distance between the source and the destination. Reducing the communication cost can be realized by lowering the power supply voltage in dynamic voltage scaling and sub-threshold designs and shortening the distance between the computing cores and memories in distributed systems.

The neuromorphic computing inspired by the working mechanism of human brains effectively reduces the data communication cost and, consequently, achieves very high computation efficiency. As both the frequency of the spikes and their relative timing are carrying on the transmitted information, the spikes can be very short and sparse to minimize the amount of the relocated electrical charge. Moreover, neuromorphic computing minimizes the data communication distance by distributing the data into the memories (i.e., synapses) close to the associated computing units (i.e., neurons) throughout the entire system. For example, TrueNorth – the spike-timing-based neuromorphic hardware prototyped by IBM achieved an extremely low energy consumption of 45pJ per spike in data communication [3]. However, such conventional CMOS implementation of neuromorphic designs suffers from a large area, high power consumption, and low-level parallelism. In TrueNorth, each neurosynaptic core contains an array of 1024×256 synapses built on SRAM cells. The majority of system energy was consumed on retaining synaptic weights, programming synapses in the learning process, and reading out the stored weights in the recall function. Moreover, since the SRAM array cannot support truly parallel execution, TrueNorth runs at only 1 kHz even though the operating frequency within each neurosynaptic core is 1 MHz [4].

Since the first actual device demonstration, memristor technology has gained significant attention in neuromorphic system society [5]. Many important features of biological synapses, such as long-term potentiation (LTP), long-term depression (LTD), and spike-timing-dependent plasticity (STDP), have been realized on memristor devices [6]-[9]. More importantly, the two-terminal structure of memristors enables the cross-point array implementation with high storage density and sub-pJ access energy [10].

Previously, we studied the input-output feature of the memristor crossbar that employs a memristor at each intersection of horizontal and vertical metal wires. We found that this typical array structure not only offers a massive number of connections but also naturally

provides the capability of connection matrix storage and matrix-vector multiplication [11][12]. To evaluate the potential of memristor crossbars in complex and large-scale pattern association and classification applications, we exploited the possibility of applying memristor crossbars in neuromorphic hardware design and used the Brain-State-in-a-Box (BSB) model, an auto-associative neural network, as a testing vehicle. In the previous project, we successfully implemented the spiking-based and level-based controller designs by using GlobalFoundries 130nm technology. The circuit modules, including input decoder/driver, current amplifier, integrate-and-fire (IFC), counter, and comparator, as well as the entire controller design, has passed the functionality test.

In this three-year project, the team at Duke University (Duke) continued the effort of implementing the programmable neural network computing engine based on memristor crossbar technique and peripheral circuitry in the TSMC 65nm process. The system integration of the CMOS chip and memristor crossbar was realized at the PCB board level. Typical neural network applications, such as multi-layer perceptron and convolutional neural networks, were adapted against hardware constraints and deployed on our chip. FPGA board works as the holistic system controller, performing chip configuration, data pre-processing, and necessary data routings for multi-chip co-ordinations. During the performance period of the project, we worked closely with Dr. Qiangfei Xia at the University of Massachusetts Amherst (UMass), Dr. Jianhua Yang at the University of Southern California, and Dr. Hao Jiang at San Francisco State University (SFSU) on memristor device development and analog circuit components. With support from the device and circuit levels, the Duke team made the following accomplishments during the performance period:

- We designed and fabricated a controller chip specified for one 32×32 1T1R (one transistor one memristor) crossbar array developed by the UMass team. The use of transistors in the crossbar array provides better control on cell resistance level and, therefore, computation accuracy. We integrated the controller design with a resistor array on a PCB board to realize a one-layer neural network and demonstrate its functionality.

- Based on the testing results from the first stage, we further optimized the PCB board design to support multiple crossbar arrays. At the system level, a multi-layer perceptron network was realized. An FPGA board was integrated with the PCB board as a system controller, performing chip configuration, data pre-processing, and necessary data routing for multi-chip co-ordination.

- We experimented with simulating efficient in-situ training techniques for high-density 1S1R (one-selector-one memristor) crossbar arrays. We proposed a single-spike scheme for extremely high energy efficiency and split the STDP process into separate LTP and LTD phases to ensure functionality.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1. The Design Concept

Nearly forty years ago, Professor Leon Chua predicted the existence of the memristor—the fourth fundamental circuit element, to complete the set of passive devices that previously includes only resistor, capacitor, and inductor [13]. Memristor uniquely defines the relationship between the magnetic flux (φ) and the electric charge (q) passing through the device, or $d\varphi = M \cdot dq$. Considering that magnetic flux and electric charge are the integrals of voltage (V) and current (I) over time, respectively, the definition of memristor can be generalized as

$$\begin{cases} V = M(\omega, I) \cdot I \\ d\omega / dt = f(\omega, I) \end{cases}. \tag{1}$$

Here, ω is a state variable; $M(\omega, I)$ represents the instantaneous memristance, which varies over time. For a "pure" memristor, neither $M(\omega, I)$ nor $f(\omega, I)$ cannot be expressed as an explicit function of I. These devices were primarily intellectual curiosities until HP Lab demonstrated the first physical realization of memristor, in which the memristive effect was achieved by moving the doping front along TiO_2 thin-film device [5]. Soon, more materials with memristive properties have been reported or rediscovered including spintronic devices [14][15], polymeric thin film [16][17], magnetic tunnel junctions (MTJ) [18][19], AlAs/ GaAs/AlAs quantum-well diodes [20], etc.

Crossbar array is a typical structure in memristor-based memories. As illustrated in **Figure 1**, it employs a memristor device at each intersection of horizontal and vertical metal wires without utilizing any extra selection device. Such a simple 1R structure with only one memristor per cell offers the smallest data storage unit size, i.e., $4F^2$, where F represents the technology feature size. Our previous research demonstrated that memristor crossbar array can provide a large number of signal connections within a small footprint and conduct the weighted combination of input signals, making it naturally attractive for implementation of connection matrix in neural networks [11][12]. To improve the array controllability and device reliability, 1T1R (one transistor one memristor) and 1S1R (one selector one memristor) cell structures have also been widely explored in crossbar array development.

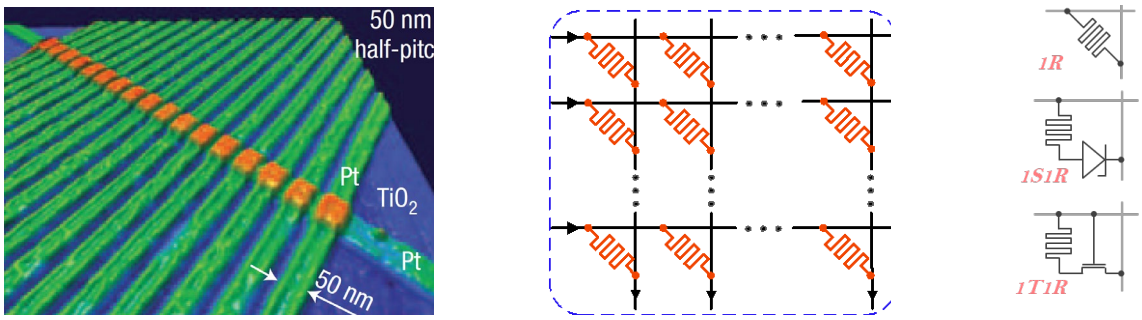

Figure 1: (a) Memristor in crossbar array (HP Labs); (b) diagram; (c) different cell structures.

Matrix computations widely exist in the applications like data signal processing, pattern recognition, weather prediction, machine learning, etc. Because of the structural similarity, reconfigurable resistive array is conceptually efficient for matrix-based operations. For instance, flash transistor array has been used to implement matrix computations. However, as a four-terminal device, such transistor-based designs generally have a large cell size. Memristor crossbar arrays have also been proved to efficiently perform common operations, e.g., matrix-vector multiplication, in neuromorphic algorithms. For example, we can realize the matrix-vector multiplication by using the crossbar array shown in **Figure 1** as follows: (1) represent the input vector as a set of input voltage signals to wordlines (WLs) of the memristor crossbar; (2) transfer the mathematical matrix to the resistance state (or more precisely, the conductance levels) of the memristors in the array; and (3) collect the currents at the bitlines (BLs) as the output vector of the matrix-vector multiplication operation.

In previous collaborations with *Air Force Research Lab* (AFRL), we have successfully demonstrated the application of memristor crossbar arrays in pattern recognition through *Brain-State-in-a-Box* (BSB) model [11][12]. The mathematical model of a BSB recall function can be represented as:

$$\mathbf{x}(t+1) = S\left(\alpha \cdot \mathbf{A}\mathbf{x}(t) + \beta \cdot \mathbf{x}(t)\right). \tag{2}$$

For a given input pattern $\mathbf{x}(0)$, the recall function computes Equation (3) iteratively until *convergence*, that is, when all entries of $\mathbf{x}(t+1)$ are either '1' or '–1'. Accordingly, a feedback loop is also needed to conduct the assignment of the output based on $\mathbf{x}(t)$ to $\mathbf{x}(t+1)$. Besides the matrix-vector multiplication $\mathbf{A}\mathbf{x}(t)$, the operations of BSB model also include the scalar scaling of vectors, sum, and a "squash" function.

The conceptual diagram of such a system is illustrated in **Figure 2**. Since all of the terms in a memristance-determined matrix must be positive, the design splits the positive and

negative terms of the BSB weight matrix into two matrices $\mathbf{A}^+$ and $\mathbf{A}^-$ and map them to two memristor crossbar arrays **M+** and **M−**, respectively. Here, the normalized input vector $\mathbf{x}(t)$ is converted to a set of input voltage signals $\mathbf{V}(t)$. Then the BSB recall algorithm can be realized by a voltage feedback system represented as:

$$\mathbf{V}(t+1) = S'\left(G_1 \cdot \mathbf{V}_+(t) - G_1 \cdot \mathbf{V}_-(t) + G_2 \cdot \mathbf{V}(t)\right). \tag{3}$$

Here, $S'(v)$ is the modified "squash" function defined as

$$S'(v) = \begin{cases} V_{op-} & (V \le V_{op-}) \\ V & (V_{op-} < V < V_{op+}) \\ V_{op+} & (V \ge V_{op+}) \end{cases}. \tag{4}$$

The design is an analog system consisting of three major components. (1) *Memristor arrays*, the key components of the overall design, are used to realize the matrix-vector multiplication function. (2) The input signals $\mathbf{V}(t)$ along with the corresponding voltage outputs of two memristor arrays, $\mathbf{V}_+(t)$ and $\mathbf{V}_-(t)$ are fed into *summing amplifiers* to realize Eq. (3). G_1 stands for the Gain of amplifiers for output signals from the memristor crossbars, and G_2 stands for the Gain of amplifiers for input signals. They reflect the learning rate in the recall function. The modified 'squash' function $S'(v)$ in Eq. (4) is naturally realized with the amplifiers since the output signal of an amplifier is bounded by the positive/negative power voltage signals, V_{op+} and V_{op-}. (3) *Comparators* are used to check for convergence of the BSB recall operation and give the decision signal V_{conv}.

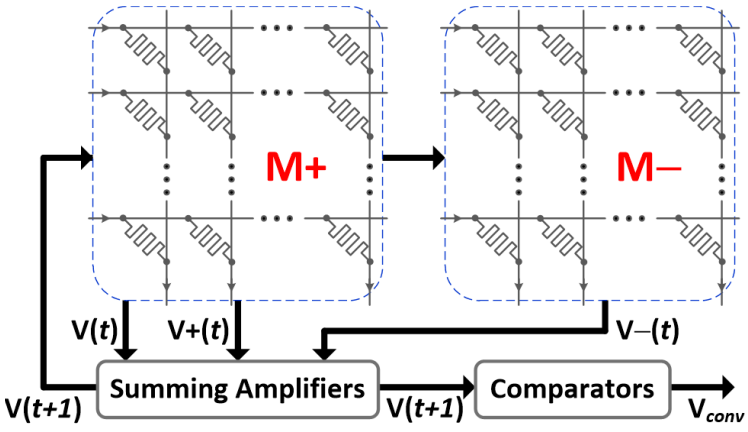

Figure 2: The conceptual diagram of the BSB recall circuit

3.2. The Controller Design

The controller chip design consists of three separate logic blocks: the word-line (WL) logic controls the gate-lines of transistors in a 1T1R array; the bit-line (BL) logic implements the converting of digital data and select signals into analog pulses to the top elements of memristors in the array; finally, the source-line (SL) logic processes the output current signals from the bottom elements of the memristor array and converts the current into a spiking voltage signal of proportional rate that is counted by a digital output circuit. These circuits can operate in memory-mode, in which the selected WL is turned on and the selected array cell is activated where the selected BL and selected SL intersect. Conversely, the circuits can operate in compute-mode, where all WLs are turned on, the BLs produce voltage pulses X clock-cycles long (where X is the value of the latched digital value for each BL), and the SLs count the number of spikes produced by the resulting current during the compute phase. Each BL has a latch that can be written before the compute phase, and each SL has a latch that can be read after the compute phase.

In the BL logic block, the digital compute-mode flag (CM) is the selection signal for a multiplexor. Each BL is attached to the voltage source BL_EXT0 when the output of its multiplexor is 0 and BL_EXT1 when the output of its multiplexor is 1. When in memory-mode, CM = 0 and the output of the multiplexor is the decoded select signal for each BL, which is a logical 1 for the selected BL only and a logical 0 otherwise. When in compute-mode, CM = 1, and the output of the multiplexor is the PWM (pulse-width modulation) signal from the BL's compute block. This block can be seen in **Figure 3** below.

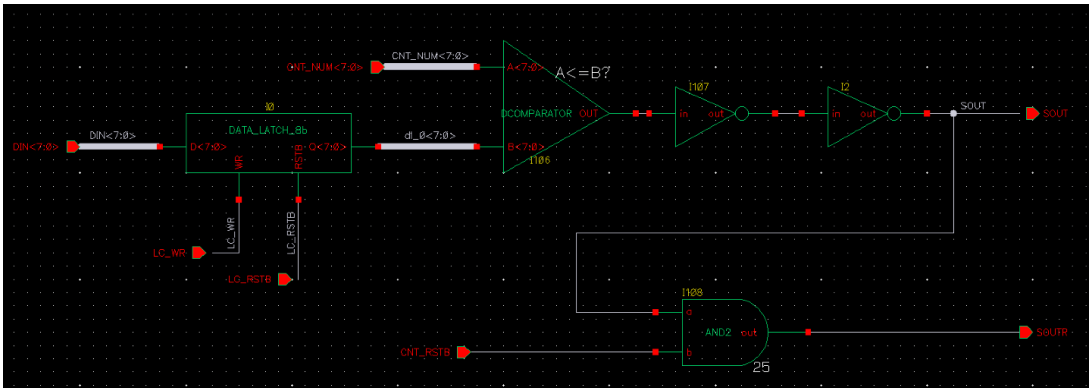

Figure 3: Digital-to-PWM compute block for BL logic. The value stored in the latch is compared to the counter. The output is filtered by the CNTR_RSTB_BL signal

A digital value can be latched to a selected BL by passing in the 8-bit value to be stored, the 5-bit select signal for the desired BL, and enabling LC_WR (latch write enable) and LC_RSTB (latch reset bar). When initiating the compute phase, the CNTR_EN_BL (counter enable for BL) and CNTR_RSTB_BL (counter reset bar for BL) signals are enabled, and the counter begins counting clock cycles. The PWM signal of a BL is high until the digital counter reaches the value stored in that BL's latch, at which point the PWM

swings low again. To ensure that the BL is only high starting at the beginning of the compute phase and ending when the counter reaches the value of the latch, the output for the BL is ANDed with CNTR_RSTB_BL.

For memory-mode in the SL logic block, the decoded select signal is ANDed with ~CM, which results in only the selected SL having an output of logical 1 and all other SL's having an output of logical 0. In compute-mode, CM = 1 activates a pass-gate that connects each SL to the current amplifier of the SL compute block. The current amplifier receives the output current from the memristor array while holding the voltage of the SL at CA_VREF. The next block of the logic, the integrate-and-fire circuit (IFC), is shown in **Figure 4.**

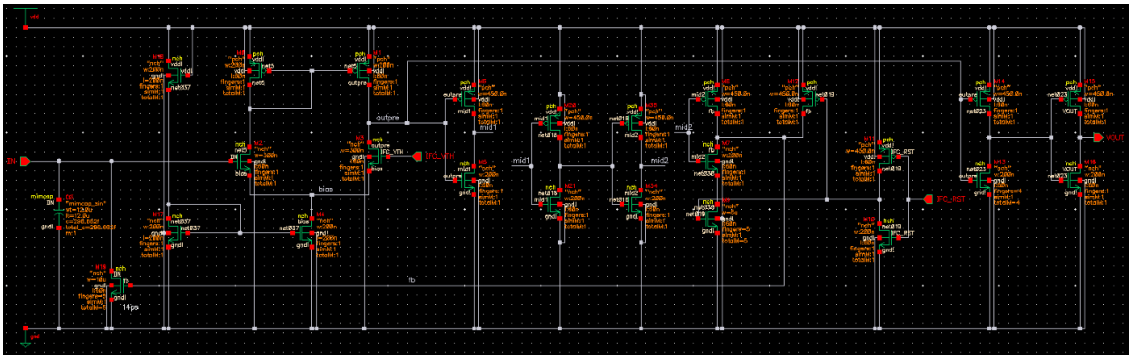

Figure 4: IFC converts current from amplifier into digital spikes. A capacitor is compared to a threshold by a comparator, and the resulting oscillatory signal is amplified by inverters to generate a spiking waveform

The current amplifier can multiply the current by a shrinkage coefficient (x1, x10, x20, or x40) based on digital flags and charges the capacitor of the IFC block. The comparator in the IFC compares the voltage on the capacitor to IFC_VTH, swinging high when the capacitor reaches this point. The output of the comparator is attached to the gate of a transistor that discharges the capacitor and causes the voltage on the capacitor to dip below the threshold again. When the transistor is shut off again, the current from the amplifier once again begins to charge the capacitor. These dynamics create an oscillating voltage signal that is buffered by inverters into a digital, spiking voltage signal. An example of these signals is shown in **Figure 5.** The spikes are counted by a digital counter identical to the counter block used in the BL logic. After the compute phase, the IFC_RST signal should be enabled to prevent subsequent spikes from being counted. Each counter can then be read by cycling the select signal through each SL.

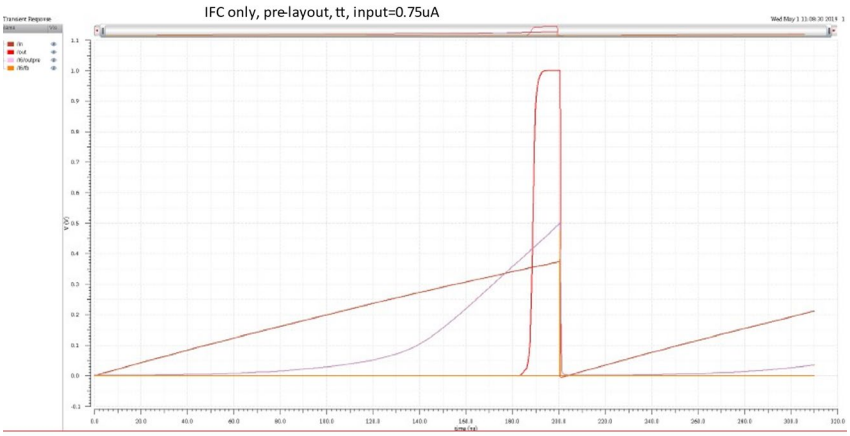

Figure 5: Wave forms converting analog current to digital spikes. The voltage across the capacitor (brown) increases linearly until the comparator spikes (red) and discharges the capacitor

Through simulation, we predicted the expected spiking frequency of the combined current-amplifier and IFC blocks (CA+IFC) given a range of input current magnitudes, which is shown in **Figure 6.** This data can be used to constrain the resistance ranges of memristor arrays in a given application. Finally, the WL logic behaves identically to the SL logic while in memory-mode, but all WLs are activated during compute mode. This is implemented by ORing the CM signal with the decoded WL select signal. When CM = 0, only the selected WL swings high, but when CM = 1, all WLs swing high. The schemes for multiplexing signals to the crossbar is illustrated in **Figure 7.**

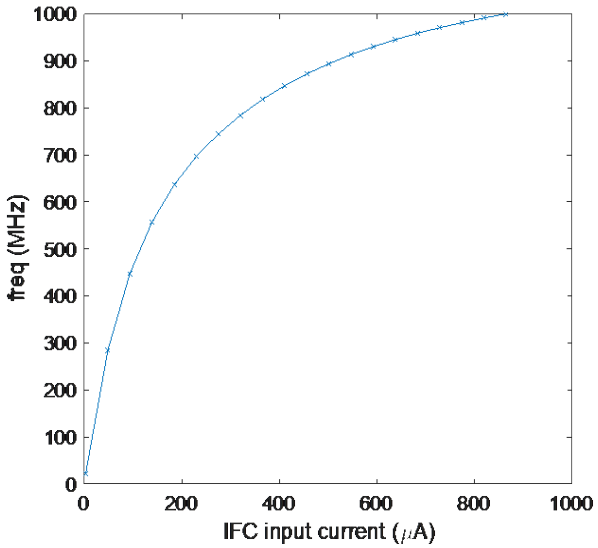

Figure 6: CA+IFC characteristics

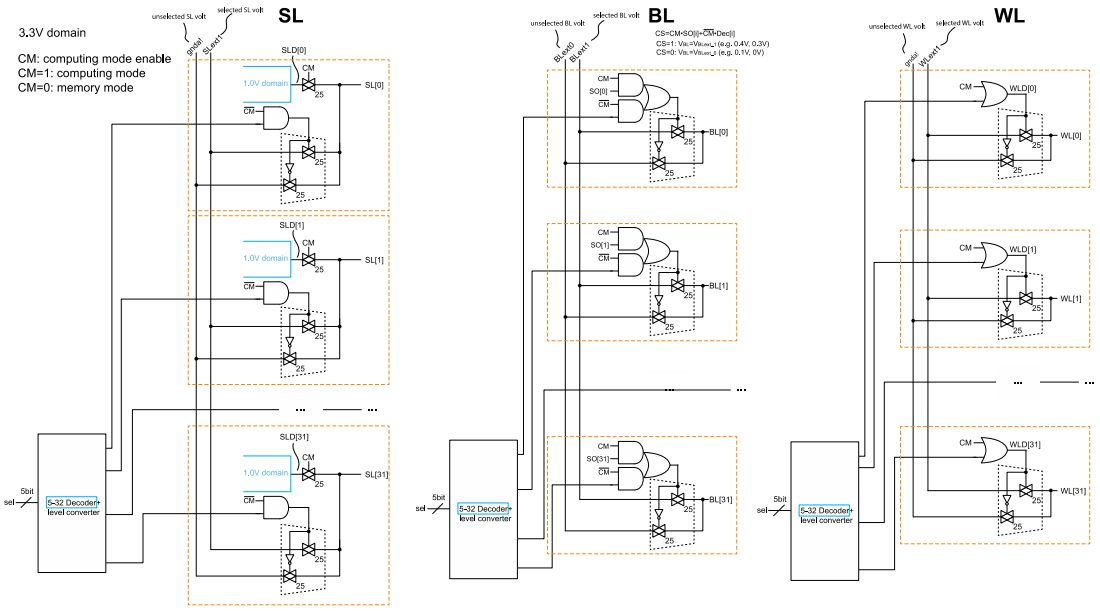

Figure 7: Multiplexer designs for SL/BL/WL

The design layout and the fabricated chip under microscope are both shown in **Figure 8** below. The final area of the chip, including the pad-frame, is 1.712 mm×1.770 mm, implemented in TSMC 65nm technology. It has 162 I/O including power and testkey, 96 of which connect to a 32 by 32 memristor crossbar. The chip was manufactured by MOSIS and placed in a 181-pin ceramic PGA package.

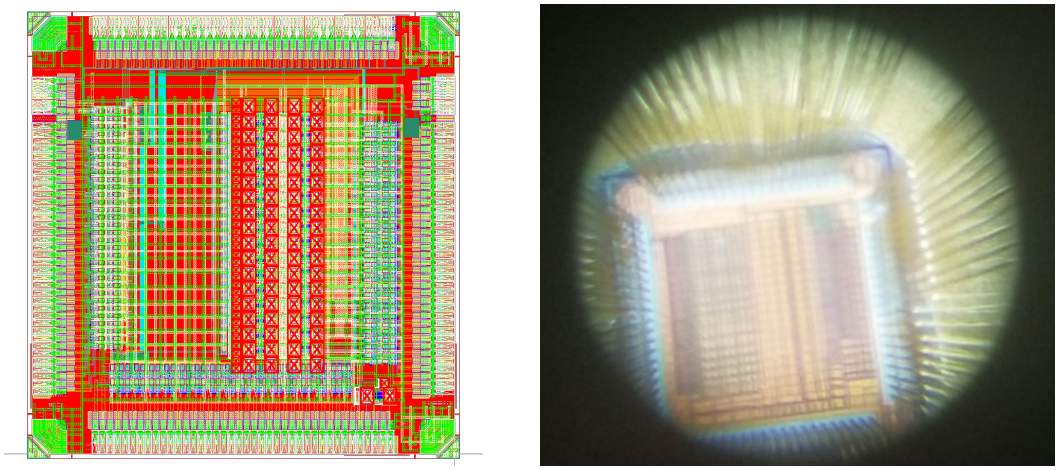

Figure 8: Chip layout (left) and the corresponding die photo (right) of the fabricated controller chip

3.3. 1S1R-Enabled Single-Spike-Encoded STDP

During the course of this project, we also experimented with simulating efficient in-situ training techniques for high-density 1S1R arrays. Dr. Yang's group at the University of Massachusetts has previously demonstrated that diffusive memristors (selector) and drift memristors that comprise 1S1R arrays can implement STDP dynamics. Rate-encoded STDP can be implemented in a 1S1R synapse by pre-synaptic and post-synaptic spikes that are a short, high-magnitude negative spike followed by a long, low-magnitude positive pulse. When a synapse with an inactive selector receives a spike from either direction, the resistance of the selector is too high for the short, high-magnitude spike to affect the memristor; i.e. the voltage drop across the memristor is below its threshold. The long, low-magnitude pulse then activates the selector in the synapse to its lowest resistive state (LRS). Between the time of this spike and the next spike that the synapse experiences, the selector's resistance begins to increase back to highest resistive state (HRS). When the second spike arrives at the synapse, if the selector's resistance is still sufficiently small, the short, high-magnitude spike changes the resistance of the memristor to a value dependent on the voltage across the memristor at that time and the memristor's current resistance. The direction of the change in resistance depends on the polarity of the spike across the synapse; i.e. which direction the second spike came from. An example of long-term potentiation (pre-synaptic before post-synaptic, LTP) and long-term depression (pre-synaptic after post-synaptic, LTD) are seen in **Figure 9(a)(b)** below.

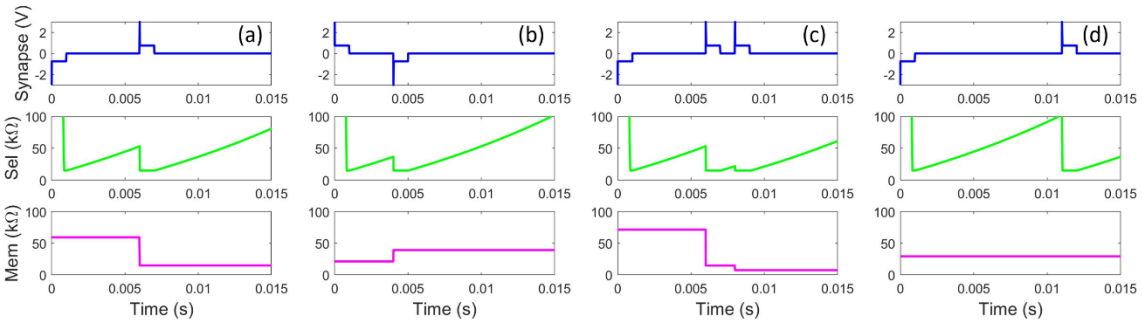

Figure 9: (a) A pre-synaptic spike followed by a post-synaptic spike induces LTP. (b) A post-synaptic spike followed by a pre-synaptic spike induces LTD. (c) A neuron with a high spike-rate causes unintentional potentiation. (d) Highly uncorrelated spikes do not induce depression

However, while attempting to implement this technique for a simple supervised learning architecture, an issue became clear that can severely limit the spike-rate of a neuron. If, for example, a pre-synaptic neuron spikes and is followed by a post-synaptic spike, the connecting synapse is potentiated as expected. However, because of how the spike waveforms are designed, the post-synaptic spike also activates the synapse's selector back to LRS. Therefore, if the post-synaptic neuron spikes again before the selector's resistance increases sufficiently, the synapse will potentiate again based on the timing between post-synaptic spikes. This unintentional potentiation can cause all synapses connected to

neurons with high spiking rates to turn completely on and disrupt learning. An example of this scenario can be seen in **Figure 9(c)**. The neurons are therefore rate-limited and cannot represent data with large precision.

Encoding data as the timing of a spike within a window is a much more efficient option with the potential for large amount of data precision. A single-spike representation of input data will obviously require less voltage spikes, and if we represent output data as a spike-time within a window as well, this would have higher precision than rate-encoded systems that simply count output spikes within a window. In this single-spike system, we can represent input data with pre-synaptic spikes and training data as post-synaptic spikes with waveforms similar to those in the rate-encoded system previously mentioned. However, we must address the case in which a pre-synaptic spike and post-synaptic spike occur on opposite ends of the time window. In this case, the neurons are uncorrelated, and we would expect the connecting synapse to be depressed in an STDP synapse. This is not the case, as the selector's resistance increases too much in the time between spikes, and the second spike has no affect on the memristor resistance. An example of this situation is shown in **Figure 9(d)**. While this situation would be fine in a rate-encoded system where the first neuron would spike again just after the second spike, depressing the synapse, we need to address this issue explicitly in the single-spike system.

We therefore proposed a system in which the STDP process was split into separate LTP and LTD phases. The LTP phase uses a very similar waveform and methodology. An input spike involves a long, low-magnitude pulse to the word-line, which activates each selector in the bit-line. A training spike involves a short, high-magnitude negative pulse to the bit-line, which decreases the resistance of synapses with previously activated selectors. The time between an input spike and a training spike controls how much the memristor resistance is decremented. Furthermore, if the training spike occurs before the input spike at a synpase, the synapse experiences no change. An example of an input spike followed by a training spike and a training spike followed by an input spike can be seen in the first and second column of **Figure 10(c),** respectively.

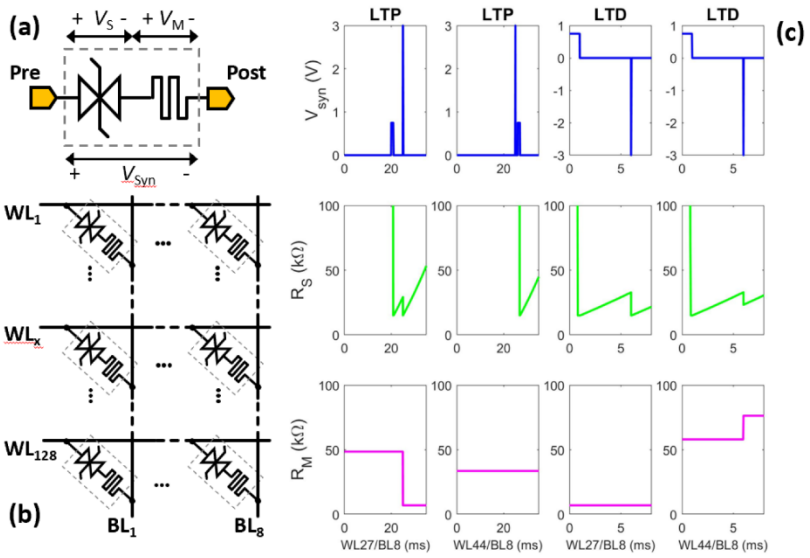

Figure 10: (a) Synapse circuit. (b) 1S1R crossbar topology. (c) LTP and LTD phases for 2 different synapses

The LTD phase involves passive weight decay (PWD). In PWD, each word-line receives a long, low-magnitude pulse simultaneously to activate each selector in the array. The selectors' resistances increase over time, and after about 5ms, we apply a short, high-magnitude positive pulse at each bit-line, increasing each memristor's resistance by an amount independent of the temporal relationship of that synapse's input and training spikes. Over the course of training, highly-correlated synapses will have a net potentiation, while highly-uncorrelated synapses will have a net depression. An example of two synapses being depressed, one with a much lower starting resistance, can be seen in the third and fourth column of **Figure 10(c),**. Note in the third column of **Figure 10(c)**, this synapse experiences no change in resistance during the LTD phase. To put it simply, during LTD, strong synapses stay strong and weak synapses stay weak. This can be attributed to the fact that the synapse is a voltage divider, as seen in **Figure 10(a),**. If the memristor in the synapse has a low resistance, less voltage drops across the memristor during the high-magnitude spike, and if the voltage drop is less than the memristor's threshold, it experiences no change in resistance. Conversely, if a memristor has a high resistance, more voltage drops across the memristor and increases the resistance significantly. This is actually a benefit of 1S1R synapses and is referred to as weight-dependent LTP/LTD. Highly-correlated synapse weights will remain high, and highly-uncorrelated synapse weights will be quickly pulled downward.

In order to test our training algorithm, we created two sample patterns, each with a set of input spikes and a set of training spikes. We trained a 1S1R array with the described algorithm and compared the results to that of a classic STDP algorithm, which doesn't have the weight-dependence described in the previous paragraph. Both patterns were used to train models, one trained directly in 1S1R, and one trained off-line with the classic STDP and then mapped to a resistive array. The outputs spikes when each input pattern is

presented to the trained arrays are shown in **Figure 11(a).** As expected, both models are capable of associating the input patterns with the training patterns. The trained synapse distributions can also be seen in **Figure 11(b).** Notably, the 1S1R-trained array has a much sparser distribution of synapses as a result of the weight-dependence. This sparse distribution of synapse weights means that average cell resistance is higher and will correspond to lower power consumption. The average cell resistance is 4.75x higher than the classic STDP-trained array and should correspond to 4.75x lower power consumption. The sparse distribution is also beneficial for higher noise resilience. As only the most correlated synapses have large weights, the 1S1R-array is able to retain the spatio-temporal relationship of the output pattern, even when a Gaussian noise vector is multiplied by an increasing parameter p and added to the vector of input spikes. The RMSE vs. p is shown in **Figure 11(c),** demonstrating that the 1S1R-trained array is approximately twice as resilient to noise.

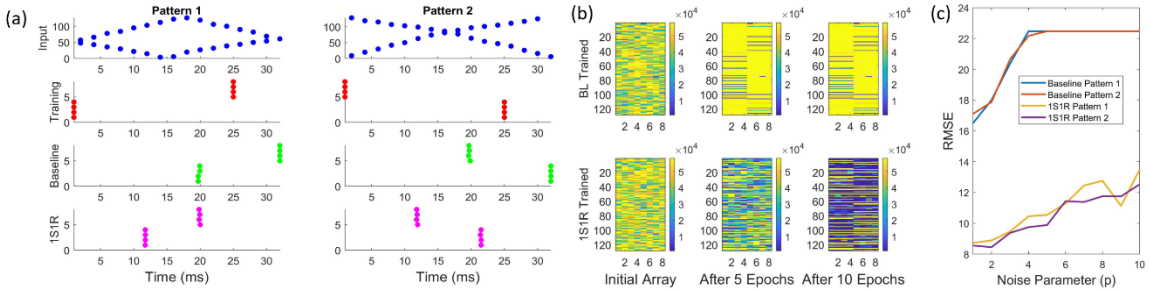

Figure 11: (a) Input and training spikes for two different patterns, as well as the output spikes for both 1S1R and baseline classic STDP-trained models. (b) Synapse distributions for both trained models. (c) RMSE vs. noise parameter *p* for both models

4.0 RESULTS AND DISCUSSION

4.1.1 Testing Fabricated Controller

After designing our first PCB for chip testing, our first test involved powering up our 1T1R controller chip and passing an input into the test inverter. Subsequent tests involved testing the three different logical blocks (BL, SL, and WL) in the two different modes (memory-mode and compute-mode). To complete the first test, we provided the analog and pad-frame power rails with 2.5V and the digital power rails with 1V. After powering up the chip, we provided the test inverter with a clock signal from an off-board FGPA and measured the test inverter output. The test inverter was functioning properly, and can be seen in **Figure 12** below.

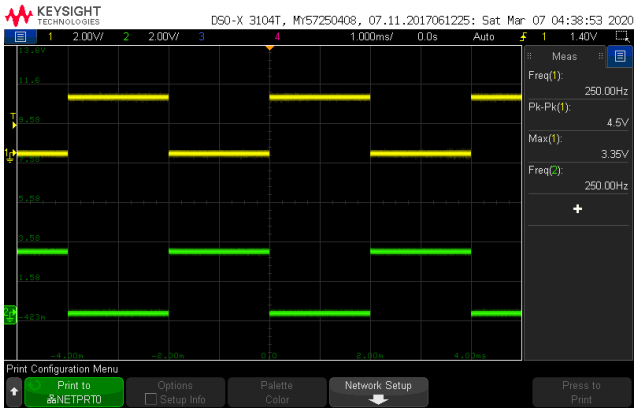

Figure 12: Test inverter input (yellow) and output (green) signals

We then moved on to memory-mode tests of all three logic blocks, as this mode is easier to test for functionality. In memory-mode, the BL, WL, and SL blocks should take the 5-bit select signal and two analog voltage sources, xL_EXT1 and xL_EXT0 (x, here, representing B, W, or S). Whichever BL, WL, or SL is selected by the select signal should be connected through the multiplex circuit to their respective xL_EXT1, and all other lines should be connected through the multiplex circuit to their respective xL_EXT0 (as described previously). We provided the WL block with the inputs SEL_WL = 5b00000, WL_EXT1 = 0.36V, and WL_EXT0 = 0.1V. Only the 0^{th} word-line received WL_EXT1 as expected. When SEL_WL was changed to 5b00010, the 0^{th} word-line then received WL_EXT0 and only the 8^{th} word-line received WL_EXT1 as expected. These results are communicated in **Figure 13** below.

SEL_WL	5b00000	5b00010
WL0	0.36	0.1
WL4	0.1	0.1
WL8	0.1	0.36
WL12	0.1	0.1
WL16	0.1	0.1
WL20	0.1	0.1
WL24	0.1	0.1
WL28	0.1	0.1

Figure 13: Analog outputs of WL blocks in memory-mode

The next test was the SL memory-mode test. This test was identical to the previous test. We set SEL_SL = 5b00000, SL_EXT1 = 0.36V, and SL_EXT0 = 0.1V, and found that the 0^{th} source-line was the only source-line measured at 0.36V as expected. Changing SEL_SL to 5b00010, only the 8^{th} source-line measured at 0.36V as expected. This output mirrors exactly the outputs shown in **Figure 13**. The operation of the BL logic in memory-mode is identical to the operation of the SL and WL logic in memory-mode, with the exception that several of the digital signals for the BL logic are inverted by the network of buffers in the chip. The buffers in the chip are inverting, so any digital signal using an odd number of buffers inverts the input signal. The inverted signals include: SEL_BL, D_OUT, CS_BLORSL, IFC_RST, CA_MR, CNTR_EN_BL, CNTR_RSTB_BL, LC_WR, LC_RSTB, and SW_FUNC_SPIKE. A few of these signals have not yet been explained. CS_BLORSL is a digital signal that indicates whether the data port is in input-mode or output-mode. Due to the inversion, the port is an output port (D_OUT, not D_IN) when CS_BLORSL is low, and an input port when CS_BLORSL is high. CA_MR is a two-bit signal that indicates which shrinkage coefficient the current amplifier will use, with 2b00 meaning x1, 2b01 meaning x10, 2b10 meaning x20, and 2b11 meaning x40 after inversion. SW_FUNC_SPIKE is the input for the test inverter, as well as indicates whether to AND the output of the comparator with CNTR_RSTB_BL.

Furthermore, due to inversion SEL_BL and D_OUT are inverted, IFC_RST resets the IFC when low, CNTR_EN_BL enables the counter when low, CNTR_RSTB_BL resets the counter when high, LC_WR enables the latch when low, and LC_RSTB resets the latch when high. Therefore, when testing the BL logic in memory mode, SEL_BL = 5b11111 when BL_EXT0 = 0.1V and BL_EXT1 = 0.36V made the 0^{th} BL equal to 0.36V and all other BLs equal to 0.1V. Changing to SEL_BL = 5b11101 flipped the 0^{th} BL to 0.1V and the 8^{th} BL to 0.36V, as in **Figure 13**.

In compute-mode, the simplest block to test is the WL block. When in compute-mode, all WLs should be attached to WL_EXT1. This is to turn on all transistors in a 1T1R array.

Indeed, in compute-mode, the chip sets all WLs to WL_EXT1, regardless of the SEL_WL input. In order to test the compute-mode of the SL logic, we set CA_MR = 2b00, SL_EXT0/CA_VREF = 200mV, and SL_EXT1/IFC_VTH = 600mV. Using a discrete resistor, we connected one end to a SL and the other end to a power source. The SL was held at 200mV by the current amplifier, and as we increased the voltage supplied by the power source, we controlled the amount of current that the CA+IFC block received. We varied the input current from 133uA to 533uA and measured the 7^{th}-bit of the counter for that SL. When given a constant current source, the counter would count the spikes from the IFC, and the period of the 7^{th}-bit waveform would tell us how long it would take for the IFC to generate 256 spikes given the input current. From this information, we could calculate the spiking frequency of the IFC, which is shown plotted against the input current in **Figure 14**.

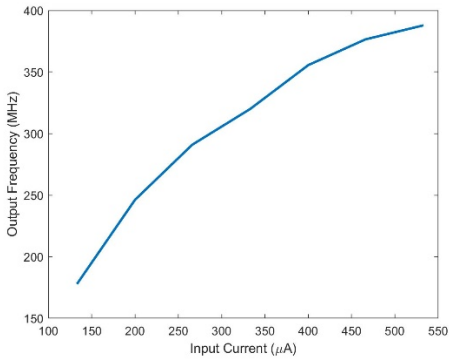

Figure 14: Measured output frequency vs. input current for a single SL

The curve has a similar shape to the curve expected in **Figure 6**. To test the BL logic in compute-mode, we could latch data into certain BLs and let the input counter continuously overflow and repeat the compute phase. If, for example, we latched the value 64 into BL0, we would expect BL0 to generate a waveform that looks like a square wave with 25% duty cycle. If we latched the value 16 into BL7, we would expect BL7 to generate a waveform that looks like a square wave with a 6.25% duty cycle. We attempted these examples, as shown in **Figure 15 (left),** where the top waveform (green) shows a BL whose latch was written with 64, and the bottom waveform (yellow) shows a BL whose latch was written with 16. These waveforms were generated with a clock signal of 50MHz, which contributed to the noisy nature of the waveforms. We rewrote the latches for these BLs before each compute-phase by changing SEL_BL and D_IN to the desired BL and input data and then flipping LC_WR low for a cycle. The LC_WR waveform is shown in **Figure 15 (right)** to show that the BLs were being rewritten each time.

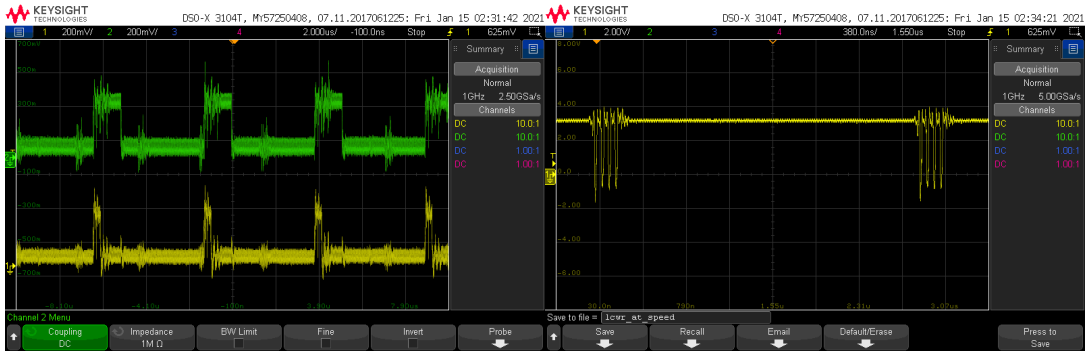

Figure 15: Left: Two BLs shown repeatedly in compute-phase with input data of 64 (top, green) and 16 (bottom, yellow). Right: LC_WR for a single BL, showing that the BL's latch is being written before each compute phase

To ensure that the signals for the BL logic and SL logic are lining up properly during compute-mode, we measured the output of a BL and the signals that initiate a compute phase for both blocks simultaneously. The CA+IFC blocks in the SL logic can only generate spikes and a digital output if the IFC_RST signal is high (and therefore, not discharging the IFC capacitor). To show that the SL and BL logic are in synch, we measured the IFC_RST signal and the output of two BLs with input values of 32 and 128, both in **Figure 16.** For both the BL with input of 128 and BL with input of 32, the beginning of the waveform coincides with the IFC_RST going high and enabling the IFC. With these tests completed, we developed a simple testbench using discrete resistors to implement a small perceptron.

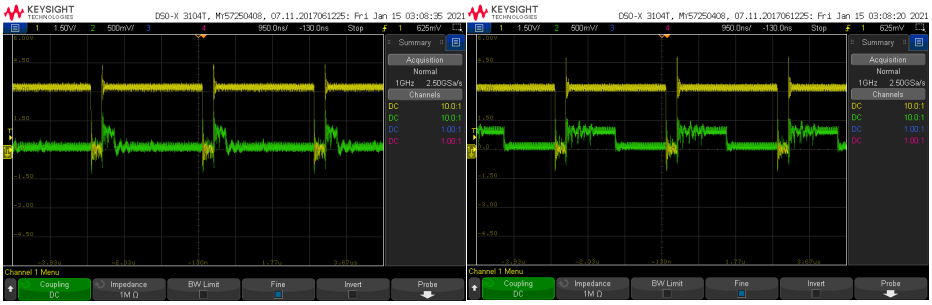

Figure 16: Two BL outputs measured against the IFC_RST signal for the SL logic

We utilized four discrete resistors for the perceptron, using 2 BLs and 2 SLs of the chip that operated as expected. The first BL was connected to the first SL by a 1K resistor and to the second SL by a 2K resistor. The second BL was connected to the first SL by a 2K resistor and to the second SL by a 1K resistor. The input clock was running at 50MHz, so we did a compute phase of 16 cycles, or 320ns. Based on our previous measurements, we expected an output of [~113, ~79] for an input of [16, 0] and an output of [~79, ~113] for an input of [0, 16]. We ran our testbench with these resistors and input vectors two times each. For an input vector of [16, 0] the measured outputs were [72, 62] and [65, 43]. For

an input vector of [0, 16], the measured outputs were [61, 78] and [55, 69]. While these outputs did not generate the exact output values expected, the outputs did follow the expected trends, with SLs that receive larger currents generating larger digital output values. It is also worth noting that the digital values measured seemed to get smaller the longer that the chip was on. We decided to next attempt a simple perceptron with multiple layers by integrating the chips on a PCB.

4.1.2 Multilayer Perceptron Integration

We designed a PCB on which multiple controller chips and memristor arrays can be used to implement larger layers or multiple layers of a neural network. This PCB can be seen in **Figure 17** below. An off-board FPGA (the Narvi Spartan 7 FPGA module) can be attached to the board to manage the flow of data. Data is received by the FPGA via UART (from Matlab, for instance) and then is written to BL latches of the controller chip (or chips) for the first layer. These chips interact with the memristor arrays and generate digital outputs that the FPGA then reads and processes before writing the latches for the next layer. After the digital outputs of the last layer are processed by the FPGA, the module can then send the results back to the main system via UART.

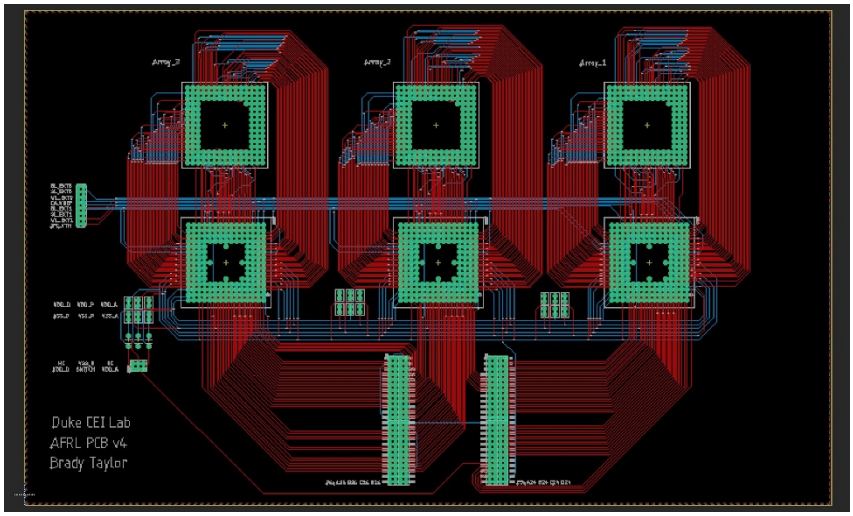

Figure 17: PCB for integrating controller chips with memristor arrays and FPGA module

Due to the pandemic, the University of Massachusetts lab was unable to get us a packaged array in time for our experiments. Therefore, we demonstrated the flow of data in a multilayer perceptron using discrete resistors to represent weights. The flow for this simple demonstration is: Matlab → UART → FPGA → controller chip 1 → resistors → controller chip 1 → FPGA → controller chip 2 → resistors → controller chip 2 → FPGA → UART → Matlab. Matlab has built-in commands for accessing COM ports on the host computer as well as for sending and receiving data via UART. We wrote Verilog modules for implementing UART on FPGA that are called by our main control module, and the Narvi

Spartan 7 module has an on-board FT2232H chip that can configure the USB port for UART communication. For the sake of the demonstration, the activations will be binary; each BL latch will be written with a value of 8 or 0. We wrote the main control module to first receive 8 bytes of data from Matlab, where each bit represents a single BL of a chip, and the bit is written to its corresponding latch. The compute-phase is 8 clock cycles long (assuming a clock of 10MHz), and BLs with a latched 1 are on for all 8 cycles while BLs with a latched 0 are on for a single cycle of the compute-phase.

To represent a single weight in this demonstration, we use two resistors from a single BL to two separate SLs; in our case, an 820-Ohm and a 2.2K resistor. The resistance values were selected to match the memristance range of the devices developed by the UMass team. The digital output of the second SL is subtracted from the digital output of the first SL to represent a single weight. When the first resistor is smaller, the first SL will experience a larger current and will produce a larger number of spikes than the second SL, resulting in a positive weight. When the second resistor is smaller, the first SL will experience a smaller current and will produce a smaller number of spikes than the second SL, resulting in a negative weight. When the outputs of these two SLs from the first layer are read by the FPGA, the FPGA writes an 8 to the corresponding BL latch of the second layer if the first SL output is larger than the second SL output and a 0 if the first SL output is smaller than the second SL output. This effectively implements a unit-step activation function on the output of the first layer and passes the results to the input of the second layer.

The BL waveforms for a logical 1 and a logical 0 in the compute phase are shown in **Figure 18** below. The ringing of the signal is due to the large clock frequency of 10MHz. For the first layer of the demonstration, we used BL31 and SL4/SL5 of the center chip of our PCB. For the first scenario, SL4 has a smaller 820-Ohm resistor and SL5 has a larger 2.2K resistor. The SLs are read by the FPGA sequentially, and we can view the output of the first layer by probing the data pins while the FPGA reads them. In **Figure 19** below, the 7th-bit (yellow) and 6th-bit (green) of the data pins for the first layer chip are probed. Keeping in mind that the output is inverted by the buffer network, we can see that SL4 and SL5 have the only nonzero outputs (the low points in the waveforms), with SL4 having a larger value than SL5. This resulted in the corresponding BL of the layer 2 chip producing a logical 1 waveform as expected.

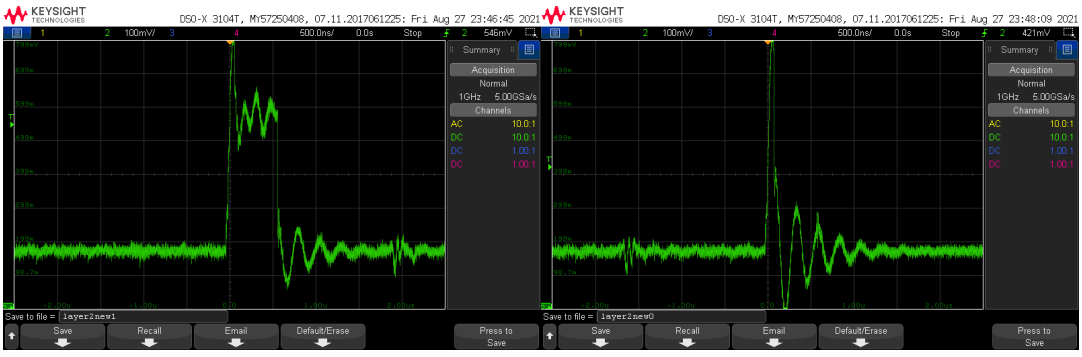

Figure 18: The BL waveform for a logical 1 (left) and a logical 0 (right).

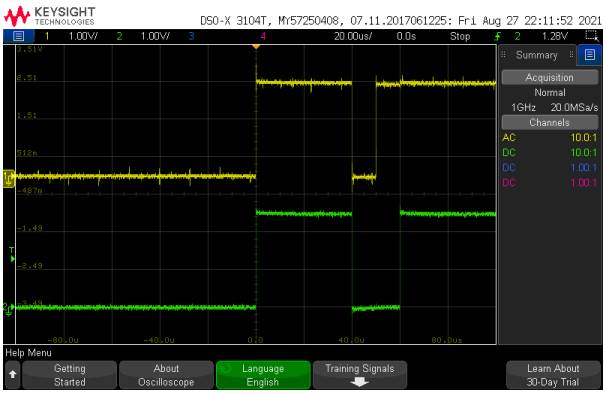

Figure 19: The 7th (top, yellow) and 6th (bottom, green) bits of SL outputs for the first layer, which are read sequentially. In this scenario, SL4 has a smaller resistor than SL5

For the second scenario, SL4 has a larger 2.2K resistor and SL5 has a smaller 820-Ohm resistor. Again, we probe the 7th and 6th bits of the data pins for the first layer chip and see that SL4 has a smaller value than SL5 this time. As a result, the corresponding BL of the layer 2 chip produced a logical 0 waveform as expected. This is seen in **Figure 20** below.

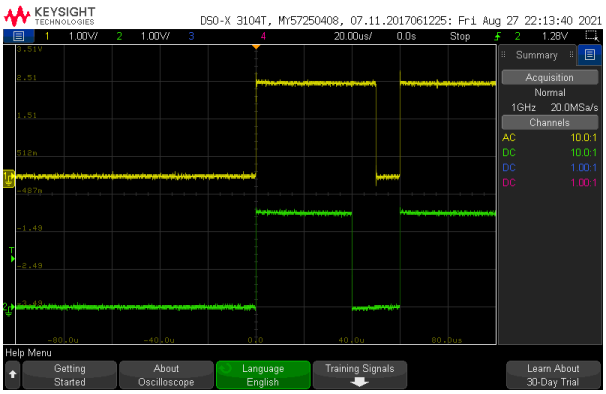

Figure 20: The 7th (top, yellow) and 6th (bottom, green) bits of SL outputs for the first layer, read sequentially. In this scenario, SL4 has a larger resistor than SL5

With resistors connected from the second layer BL to its SLs, we can read the digital output of the second layer chip and send them back to Matlab via UART. This demonstration therefore shows that we can transmit data from our host computer system to a router (modelled by the FPGA in this case) that can integrate multiple controller chip/memristor array pairs into a flexible neural network.

4.2. Progress Towards Multilayer Integration of Controller and 1S1R Array

Our current controller circuit was designed with the intention of controlling both 1T1R and 1S1R arrays without major modifications. To show the 1S1R driving capabilities of our current design, we ran simulations for both compute and memory-modes. Due to the size of the controller circuit and slow speed of its simulations, we separated these simulations from 1S1R simulations; i.e. we simulated the controller on a timescale of a few hundred nanoseconds and used the output voltage waveforms as inputs to our 1S1R simulations on a timescale of a few milliseconds. This gives us a general idea of how a 1S1R array will react to controller outputs.

The first simulation involved memory-mode operations; specifically, setting and resetting 1S1R cells within an array. The digital select signals (SEL_BL and SEL_SL) were set to 5b00000 for 55ns of the simulation, followed by 5b00100 for another 55ns of the simulation. This allowed cell (0, 0) to be set and reset first, followed by cell (4, 4). The analog signals BL_EXT1/SL_EXT1 were 2V/0V for the first 25ns, 0V/2.5V for the next 25ns, 0V/0V for 5ns as the select signals changed, and then repeated 2V/0V for 25ns followed again by 0V/2.5V for 25ns. BL_EXT0/SL_EXT0 were kept at 0V for the entire simulation. These signals were chosen to set and reset cell (0, 0), and then set and reset cell (4, 4). Notice that all analog signals were briefly 0V while the digital signals changed to prevent any unintentional voltage glitches from appearing across a 1S1R cell. Furthermore, actual pulse durations will depend on the selector technology implemented. The models used for 1S1R-based simulations were created by the lab at the University of Massachusetts. The output of the circuit given these digital and analog inputs can be seen in **Figure 21** below.

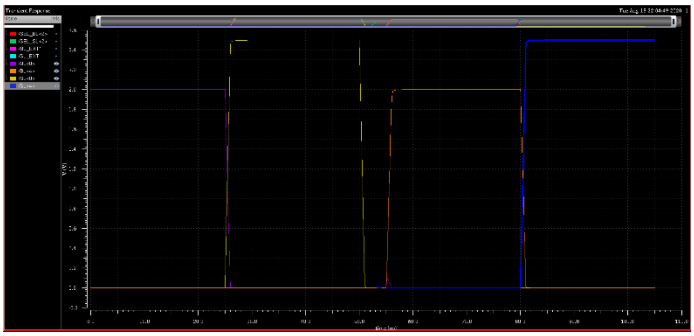

Figure 21: Controller output during memory-mode for 1S1R array

The selected cell experiences 2V when being set and -2.5V when being reset. Note that unselected cells in the same row and column as the selected cell experience 1V when being set and -1.25V when being reset. These voltage magnitudes are not large enough to disturb the state of unselected cells. For compute-mode with a 1S1R array, all selectors must first be activated and kept activated. We first activate the selectors by pulsing each BL with a long, low-magnitude voltage pulse (above the selector threshold, but below the memristor threshold; usually about 0.75V). In order to keep the selectors activated, BL_EXT0 during

compute-mode must be slightly above the selector threshold (about 0.2V). As can be seen in **Figure 22** below, keeping the "resting" voltage slightly above the selector threshold prevents the selector conductance from decaying over time.

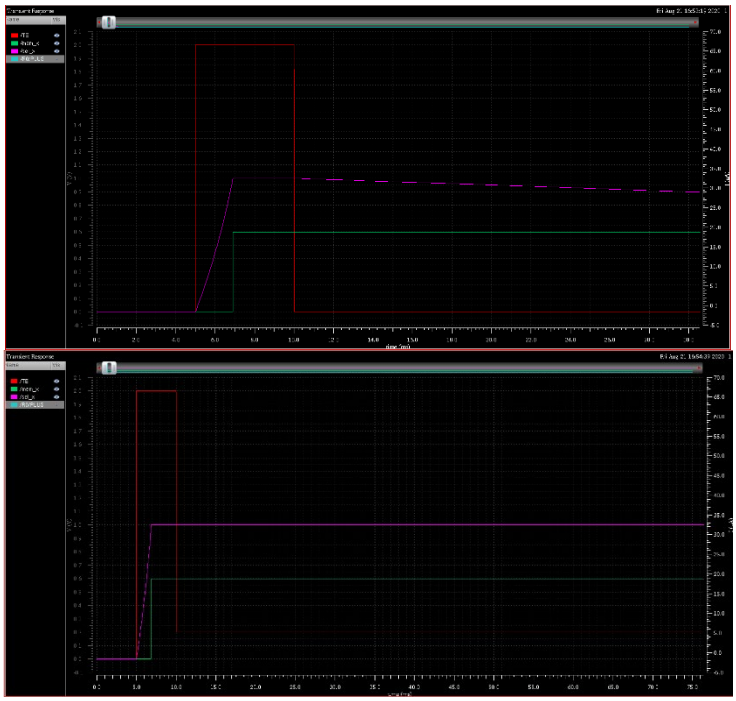

Figure 22: When BL_EXT0 is left as 0V, the selector state (pink) decays over time (top image). However, when BL_EXT0 is kept slightly over threshold (0.21V) the selector state remains high, and consequently, the conductance does not decay (bottom image)

Activating all selectors can occur by first switching into memory-mode and selecting each bit-line for a sufficient amount of time for the selectors to switch (while SL_EXT1/ SL_EXT0 are both set to 0V). Then, we can switch back to compute-mode, and continue as normal. The simulated output of the circuit throughout this process is seen in **Figure 23**. Each bit-line is pulsed for a short period of time, the latches are set with the appropriate data, and then the computation begins and each bit-line pulses a voltage for a period of time proportional to their digital input. Notice each bit-line never drops below 0.2V.

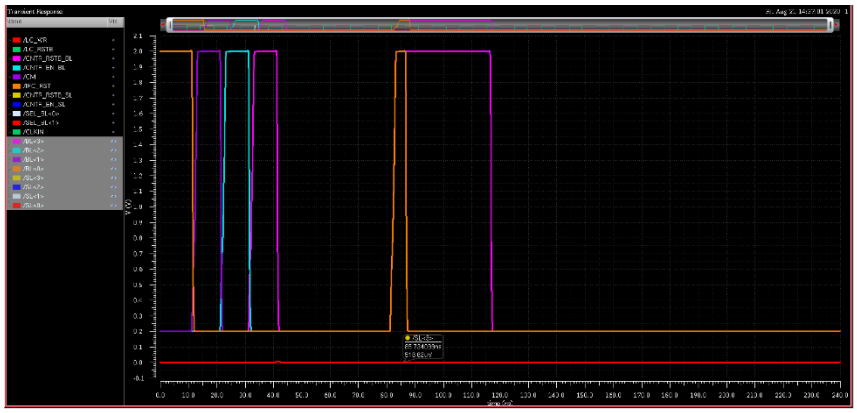

Figure 23: Simulation of compute-mode for a 1S1R array

Finally, we ran a simulation of 32 1S1R cells (representing one column of an array) feeding into our neuron circuit, which consists of a current-amplifier and integrate-and-fire circuit (IFC). After first activating all 32 selectors in the column with a long pulse (not shown), each cell received one of four pulse lengths (5ns, 12ns, 25ns, or 35ns). The current amplifier held the column's source-line at 0V, and the IFC responded with 75 spikes in a 50ns time-period. The results can be seen in **Figure 24** below. These spike rate is slightly higher than what we would expect in our compute-mode matrix-vector operations, but the results can be tuned with careful choice of input voltage magnitude, IFC threshold, and current-amplification setting.

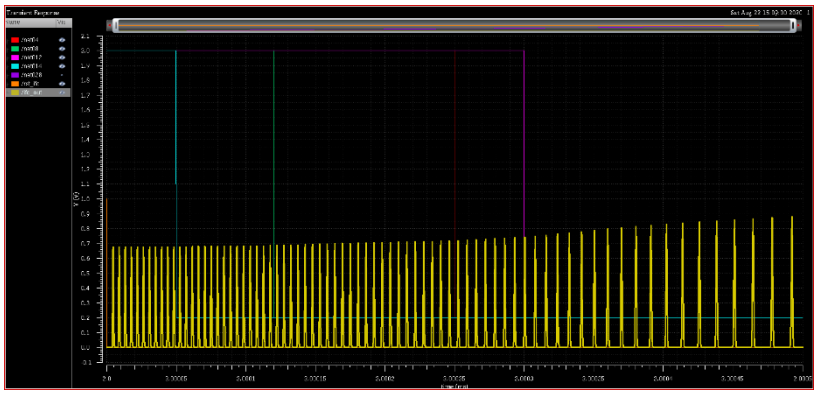

Figure 24: Simulation of CA-IFC output given an operations of a column of 32 1S1R cells

These simulations demonstrate that controlling a 1S1R array with our current circuit design should be possible with little or no hardware modifications required, and a simple change in protocol for programming the array and computing matrix-vector operations.

4.3. Complications and Potential Future Work

We encountered several complications in the project that should be addressed in future work. A major issue with the design is the relatively large variation in behavior experienced by the system. One source of this variation is the differences in responses to a given input current by each CA+IFC block. When SLs on the same chip receive the same input current, not all SLs spike at the same frequency. It is likely that the difference comes down to device variation in implementing the capacitor in the IFC or scaling the current between the CA and IFC. Differences in the capacitance of the circuit or the modulation of the current can cause the capacitors to reach IFC_VTH at different rates and produce different spike rates for given input currents. The IFCs were also designed for memristors with higher resistance ranges, meaning the operation ranges of the IFCs might be small and would not overlap very well with small-valued resistors.

The variation between SLs of the same chip seems to be related in part to the integration of the IFC and the counter. While the MSBs of the output data (bits 7, 6, 5, and 4) seem to switch at an almost constant rate, the LSBs of the output data (bits 3, 2, 1, and 0) seem to switch randomly, as if the IFC is either not spiking at a steady rate or if the counter is not picking up every spike. We hypothesize that the spikes are not activating the counter reliably. One potential solution for this issue is increasing the amplification of the spikes between the IFC and the counter to ensure that the spikes consistently have the driving power needed to switch the 0^{th} bit of the counter. Another solution would be to somehow increase the duration of each spike to ensure the spikes are sufficiently long enough to reliably switch the 0^{th} bit of the counter.

The demonstration of our chip was also further complicated by the high frequency of the signals used. The intended clock rate of our chip is 100MHz, which requires careful PCB design to reduce crosstalk between signals and prevent signal degradation due to trace capacitance and resistance. For example, the high-density routing of signals to the array pads on our demonstration PCB made it difficult to reduce crosstalk, and it was therefore very difficult to produce the correct waveform at each BL. We transmitted an alternating bit signal ("01010101…" repeated 4 times) to all 32 BLs of one chip, but when measured at the array pins at the top of the PCB, we found that we only measured the correct results consistently when the clock rate was reduced to 100kHz or below, and at that speed, only the first and last byte of data were completely correct. The center BL pins were more susceptible to crosstalk and less likely to have the correct waveform. Furthermore, integrating the controller chip and array at the PCB level can mean different SLs have different length traces connecting them to the chip. Traces of different lengths can have different resistances that modulate their current more or less than other traces. By integrating the controller chip and array on a single die or package, future designs could potentially mitigate the degradation of signals caused by the PCB.

5.0 CONCLUSIONS

In this project, the team at Duke University (Duke) worked closely with Dr. Qiangfei Xia at the University of Massachusetts Amherst (UMass), Dr. Jianhua Yang at the University of Southern California, and Dr. Hao Jiang at San Francisco State University (SFSU) on memristor device development and analog circuit components. More specifically, the Duke team implemented a programmable neural network computing engine based on memristor crossbar techniques and peripheral circuitry in the TSMC 65nm process. System integration of the CMOS chip and memristor crossbar was realized at the PCB board level. Typical neural network applications, such as multi-layer perceptron and convolutional neural networks, were adapted against hardware constraints and deployed on our chip. An FPGA module works as the holistic system controller, performing chip configuration, data pre-processing, and necessary data routing for multi-chip co-ordination. Furthermore, we experimented with simulating efficient in-situ training techniques for high-density 1S1R crossbar arrays and proposed a single-spike scheme for extremely high energy efficiency and split the STDP process into separate LTP and LTD phases to ensure functionality.

6.0 REFERENCES

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics*, vol. 38, no. 4, 1965.

[2] W. A. Wulf and S. A. McKee, "Hitting the Memory Wall: Implications of the Obvious," in *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[3] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A Digital Neurosynaptic Core Using Embedded Crossbar Memory with 45pJ per Spike in 45nm," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2011, pp. 1–4.

[4] J.-S. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, and D. S. Modha, "A 45nm CMOS Neuromorphic Chip with A Scalable Architecture for Learning in Networks of Spiking Neurons," in *Custom Integrated Circuits Conference (CICC)*, 2011, pp. 1–4.

[5] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The Missing Memristor Found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[6] G. S. Snider, "Spike-timing-dependent Learning in Memristive Nano Devices," in *IEEE International Symposium on Nanoscale Architectures (NANOARCH)*, 2008, pp. 85–92.

[7] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[8] S. Ambrogio, S. Balatti, F. Nardi, S. Facchinetti, and D. Ielmini, "Spike Timing Dependent Plasticity in a Transistor-Selected Resistive Switching Memory," *Nanotechnology*, vol. 24, no. 38, p. 384012, 2013.

[9] A. Thomas, "Memristor-based Neural Networks," *Journal of Physics D: Applied Physics*, vol. 46, no. 9, p. 093001, 2013.

[10] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A Neuromorphic Visual System Using RRAM Synaptic Devices with sub-pJ Energy and Tolerance to Variability: Experimental Characterization and Large-scale Modeling," in *IEEE International Electron Devices Meeting (IEDM)*, 2012, pp. 4–10.

[11] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware Realization of BSB Recall Function Using Memristor crossbar Arrays," in *ACM Proceedings of the 49th Annual Design Automation Conference (DAC)*, 2012, pp. 498–503.

[12] M. Hu, H. Li, Y. Chen, Q. Wu, G. Rose, and W. Linderman, "Memristor Crossbar Based Neuromorphic Computing System: A Case Study," *IEEE Transactions on Neural Network and Learning System (TNNLS)*, vol. 25, no 10, pp. 1864-1878, Oct. 2014.

[13] H. Shayani, P. Bentley, and A. Tyrrell, "Hardware Implementation of A Bioplausible Neuron Model for Evolution and Growth of Spiking Neural Networks on FPGA," in *NASA/ESA Conference on Adaptive Hardware and Systems*, 2008, pp. 236–243.

[14] X. Wang, et al., "Spintronic memristor through spin-torque-induced magnetization motion," *IEEE Electron Device Letters (EDL)*, vol. 30, pp. 294-297, 2009.

[15] Y. V. Pershin and M. Di Ventra, "Spin Memristive systems: Spin Memory Effects in Semiconductor Spintronics," *Physical Review B*, vol. 78, p. 113309, 2008.

[16] V. Erokhin and M. P. Fontana, "Electrochemically Controlled Polymeric Device: A Memristor (and more) Found Two Years Ago," 2008.

[17] J. H. Kriegerand and S. M. Spitzer, "Non-traditional, Non-volatile Memory based on Switching and Retention Phenomena in Polymeric thin Films," in *Non-Volatile Memory Technology Symposium*, 2004, pp. 121-124.

[18] Y. Huai, "Spin-Transfer Torque MRAM (STT-MRAM): Challenges and Prospects," *AAPPS Bulletin*, vol. 18, p. 33, December 2008.

[19] P. Krzysteczko, G. Reiss, and A. Thomas, "Memristive Switching of MgO based Magnetic Tunnel Junctions," *Applied Physics Letters*, vol. 95, pp. 112508-3, 2009.

[20] K. K. Gullapalli, A. J. Tsao, and D. P. Neikirk, "Multiple Self-consistent Solutions At Zero Bias and Multiple Conduction Curves in Quantum Tunneling Diodes Incorporating N^--N^+-N^- Spacer Layers," *Applied Physics Letters*, vol. 62, pp. 2971-2973, 1993.

[21] Y. Wang, W. Wen, L. Song, and H. Li, "Classification Accuracy Improvement for Neuromorphic Computing Systems with One-level Precision Synapses," *Asia and South Pacific Design Automation Conference (ASPDAC)*, January 2017.

[22] Y. Wang, W. Wen, B. Liu, D. Chiarulli, and H. Li, "Group Scissor: Scaling Neuromorphic Computing Design to Big Neural Networks," *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, June 2017.

[23] C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Q. Wu, H. Jiang, Y. Chen, and H. Li, "A Spiking Neuromorphic Design with Memristor Crossbar," *Proceedings of the 51st Annual Design Automation Conference (DAC)*, June 2015, article no. 14.

APPENDIX A – PUBLICATIONS

1) Z. Fan, Z. Li, B. Li, Y. Chen and H. Li, "RED: A ReRAM-based Deconvolution Accelerator," Design, Automation & Test in Europe (DATE), March 2019, pp. 1763-1768. (DOI: 10.23919/DATE.2019.8715103)

2) B. Li, B. Yan, and H. Li, "An Overview of In-memory Processing with Emerging Non-volatile Memory for Data-intensive Applications," in Proceedings of the 26th Great Lakes Symposium on VLSI (GLSVLSI), May 2019, pages 381-386. (DOI: 10.1145/3299874.3319452)

3) F. Chen, L. Song, and Hai Li, "Efficient Process-in-Memory Architecture Design for Unsupervised GAN-based Deep Learning using ReRAM," in Proceedings of the 26th Great Lakes Symposium on VLSI (GLSVLSI), May 2019, pages 423-428. (DOI: 10.1145/3299874.3319482)

4) B. Yan, Q. Yang, W.-H. Chen, K.-T. Chang, J.-W. Su, C.-H. Hsu, S.-H. Li, H.-Y. Lee, S.-S. Sheu, M.-S. Ho, Q. Wu, M.-F. Chang, Y. Chen, and H. Li, "RRAM-based Spiking Nonvolatile Computing-In-Memory Processing Engine with Precision-Configurable In Situ Nonlinear Activation," 2019 Symposium on VLSI Technology, Kyoto, Japan, June 2019, pp. T86-T87. (DOI: 10.23919/VLSIT.2019.8776485)

5) F. Chen, L. Song, H. Li, and Y. Chen, "ZARA: A Novel Zero-free Dataflow Accelerator for Generative Adversarial Networks in 3D ReRAM," Design Automation Conference (DAC), Jun. 2019, article no. 133. (DOI: 10.1145/3316781.3317936)

6) B. Yan, B. Li, X. Qiao, C.-X. Xue, M.-F. Chang, Y. Chen, and H. Li, "RRAM based In-Memory Computing: From Device and Large-Scale Integration System Perspectives," Advanced Intelligent Systems, volume 1, issue 7, article no. 1900068, November 2019. (DOI:10.1002/aisy.201900068)

7) B. Yan, M. Liu, Y. Chen, K. Chakrabarty and H. Li, "On Designing Efficient and Reliable Nonvolatile Memory-Based Computing-In-Memory Accelerators," IEEE International Electron Device Meeting (IEDM), December 2019. (DOI: 10.1109/IEDM19573.2019.8993562)

8) F. Chen, L. Song, H. Li, and Y. Chen, "PARC: A Processing-in-CAM Architecture for Genomic Long Read Pairwise Alignment using ReRAM," Asia and South Pacific Design Automation Conference (ASP-DAC), January 2020, pages 175-180. (DOI: 10.1109/ASP-DAC47756.2020.9045555)

9) B. Li, M. Mao, Z. Liu, X. Liu, T. Liu, W. Wen, Y. Chen, and H. Li, "Thread Batching for High Performance Energy-efficient GPU Memory Design," ACM Journal on Emerging Technologies in Computing Systems (JETC), Article No. 39, December 2019. (DOI: 10.1145/3330152; arXiv:1906.06603)

10) Z. Li, B. Li, and H. Li, "RED: an ReRAM-based Efficient Accelerator for Deconvolutional Computation," IEEE Transactions on CAD of Integrated Circuits and Systems (TCAD),

volume 39, issue 12, pages 4736-4747, December 2020. (DOI: 10.1109/TCAD.2020.2981055)

11) S. Zhang, L. Zhang, B. Li, H. Li, and U. Schlichtmann, "Lifetime Enhancement for RRAM-based Computing-In-Memory Engine Considering Aging and Thermal Effects," International Conference on Artificial Intelligence Circuits and Systems (AICAS), March 2020, pages 11-15. (DOI: 10.1109/AICAS48895.2020.9073995)

12) Y. Wang, F. Chen, C. Song, C.-J. Shi, H. Li, and Y. Chen, "ReBoc: Accelerating Block-Circulant Neural Networks in ReRAM," Design, Automation & Test in Europe (DATE), March 2020, pages 1472-1477. (DOI: 10.23919/DATE48585.2020.9116422)

13) S. Zhang, B. Li, H. Li and U. Schlichtmann, "A Pulse Width Neuron with Continuous Activation for Processing-In-Memory Engines," Design, Automation & Test in Europe (DATE), March 2020, pages 1426-1431. (DOI: 10.23919/DATE48585.2020.9116323)

14) Z. Li, B. Yan, and H. Li, "ReSiPE: ReRAM-based Single-Spiking Processing-In-Memory Engine," Design Automation Conference (DAC), July 2020, article no. 102. (DOI: 10.1109/DAC18072.2020.9218578)

15) S. Li, E. Hanson, H. Li, and Y. Chen, "PENNI: Pruned Kernel Sharing for Efficient CNN Inference," International Conference on Machine Learning (ICML), July 2020. (arXiv:2005.07133)

16) B. Kim and H. Li, "Leveraging 3D Vertical RRAM to Developing Neuromorphic Architecture for Pattern Classification," IEEE Computer Society Annual Symposium on VLSI (ISVLSI), July 2020, pages. (DOI: 10.1109/ISVLSI.2018.00092)

17) C. Song, H.-P. Cheng, H. Yang, S. Li, C. Wu, Q. Wu, and H. Li, "Adversarial Attack: A New Threat to Smart Devices and How to Defend It," IEEE Consumer Electronics Magazines, volume 9, issue 4, pp. 49-55, July 2020. (DOI: 10.1109/MCE.2020. 2969150) **(Feature Article)**

18) L. Zhang, B. Li, Y. Zhu, S. Zhang, T. Wang, Y. Shi, T.-Y. Ho, H. Li, and U. Schlichtmann, "Reliable and Robust RRAM-based Neuromorphic Computing," in Proceedings of the 27th Great Lakes Symposium on VLSI, pages 33-38, September 2020. (DOI: 10.1145/3386263.3407579)

19) Q. Zheng, X. Li, Z. Wang, Y. Cai, G. Sun, R. Huang, Y. Chen, and H. Li, "MobiLattice: A Depth-wise DCNN Accelerator with Hybrid Digital/Analog Nonvolatile Processing-In-Memory Block," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), November 2020, article no. 104. (DOI: 10.1145/3400302.3415666)

20) X. Yang, B. Yan, H. Li, and Y. Chen, "ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), November 2020, article no. 92. (DOI: 10.1145/3400302.3415640)

21) B. Taylor, A. Shrestha, Q. Qiu, and H. Li, “1S1R-Based Stable Learning Through Single-Spike-Encoded Spike-Timing-Dependent Plasticity”, accepted to IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2021. (DOI: 10.1109/ISCAS51556.2021.9401644)

22) Bokyung Kim, Edward Hanson, and Hai Li, “An Efficient 3D ReRAM Convolution Processor Design for Binarized Weight Networks,” IEEE Transactions on Circuits and Systems II (TCAS-II), volume 68, issue 5, pages 1600-1604, 2021. (DOI: 10.1109/TCSII.2021.3067840)

23) Eric Yeast, Yiran Chen, and Hai Li, “Improving Gradient Regularization using Complex-Valued Neural Networks,” Proceedings of the 38th International Conference on Machine Learning, PMLR 139:11953-11963, 2021.

APPENDIX B – PRESENTATIONS

1) Meeting name: Design, Automation & Test in Europe (DATE)
Location: Florence, Italy
Date: March 28, 2019
Attendees from this project: Hai Li
Presentation: RED: A ReRAM-based Deconvolution Accelerator

2) Meeting name: Dagstuhl Seminar on Emerging Hardware Techniques and EDA Methodologies for Neuromorphic Computing
Location: Dagstuhl, Germany
Date: April 8-10, 2019
Attendees from this project: Hai Li
Presentation: General discussion

3) Meeting name: Design Automation Conference (DAC)
Location: Las Vegas, USA
Date: June 3, 2019
Attendees from this project: Fan Chen, Hai Li
Presentation: ZARA: A Novel Zero-free Dataflow Accelerator for Generative Adversarial Networks in 3D ReRAM

4) Meeting name: 2019 Symposium on VLSI Technology
Location: Kyoto, Japan
Date: June 10, 2019
Attendees from this project: Bonan Yan, Hai Li
Presentation: RRAM-based Spiking Nonvolatile Computing-In-Memory Processing Engine with Precision-Configurable In Situ Nonlinear Activation

5) Meeting name: the Ming Hsieh Institute Seminar Series on Integrated Systems
Location: University Southern California, Los Angeles, CA, USA
Date: September 3, 2019
Attendees from this project: Hai Li
Presentation: Highly Efficient Neuromorphic Computing Systems with Emerging Nonvolatile Memories

6) Meeting name: Cadence Distinguished Speaker Series Talk
Location: San Jose, USA
Date: October 9, 2019
Attendees from this project: Hai Li
Presentation: Neural Network Acceleration on Conventional and Neuromorphic Fabric

7) Meeting name: IEEE Circuit and System Society (CASS) Distinguish Lecture Program (DLP) Talks
Location: Philadelphia, PA
Date: October 24, 2019

Attendees from this project: Hai Li
Presentation: Brain Inspired Computing: The Extraordinary Voyages in Known and Unknown Worlds

8) Meeting name: IEEE International Electron Device Meeting (IEDM)
Location: San Francisco, CA, USA
Date: December 9-11, 2019
Attendees from this project: Bonan Yan, Hai Li
Presentation: On Designing Efficient and Reliable Nonvolatile Memory-Based Computing-In-Memory Accelerators

9) Meeting name: Asia and South Pacific Design Automation Conference (ASP-DAC)
Location: Beijing, China
Date: January 13-15, 2020
Attendees from this project: Fan Chen
Presentation: PARC: A Processing-in-CAM Architecture for Genomic Long Read Pairwise Alignment using ReRAM

10) Meeting name: Design, Automation & Test in Europe (DATE)
Location: Virtual
Date: April 2020
Attendees from this project: Fan Chen; Hai Li
Presentation: ReBoc: Accelerating Block-Circulant Neural Networks in ReRAM; A Pulse Width Neuron with Continuous Activation for Processing-In-Memory Engines;

11) Meeting name: IEEE Computer Society Annual Symposium on VLSI (ISVLSI)
Location: Virtual
Date: July 2020
Attendees from this project: Bokyung Kim
Presentation: Leveraging 3D Vertical RRAM to Developing Neuromorphic Architecture for Pattern Classification

12) Meeting name: International Conference on Machine Learning (ICML)
Location: Virtual
Date: July 2020
Attendees from this project: Shiyu Li
Presentation: PENNI: Pruned Kernel Sharing for Efficient CNN Inference

13) Meeting name: Design Automation Conference (DAC)
Location: Virtual
Date: July 2020
Attendees from this project: Ziru Li; Hai Li
Presentation: ReSiPE: ReRAM-based Single-Spiking Processing-In-Memory Engine;

14) Meeting name: NSF Workshop on Machine Learning Hardware Breakthroughs Towards Green AI and Ubiquitous On-Device Intelligence

Location: Virtual
Date: November 10, 2020
Attendees from this project: Hai Li
Presentation: moderator for the Panel on “Enabling technologies”

15) Meeting name: IEEE/ACM International Conference on Computer-Aided Design (ICCAD)
Location: Virtual
Date: November 2020
Attendees from this project: Xiaoxuan Yuang; Hai Li
Presentation: ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration

16) Meeting name: IEEE Swiss CAS chapter and University of Zurich (ZH)
Location: Virtual
Date: October 30, 2020
Attendees from this project: Hai Li
Presentation: Efficient Machine Learning: A Cross-layer Co-design Approach

17) Meeting name: DOE Energy Frontier Research Center (EFRC) on “Quantum Materials for Energy Efficient Neuromorphic Computing” (Q-MEEN-C) seminar series, University of California, San Diego
Location: Virtual
Date: October 8, 2020
Attendees from this project: Hai Li
Presentation: Neuromorphic Computing - Technology, Hardware and Implementation

18) Meeting name: Grace Hopper Celebration
Location: Virtual
Date: September 29, 2020
Attendees from this project: Hai Li
Presentation: Panelist speaker for the Panel on “Brain-Inspired Computing for the Edge”

19) Meeting name: ECE Distinguished Speaker Series Talk, Rice University
Location: Virtual
Date: September 5, 2020
Attendees from this project:
Presentation: Efficient Deep Learning at Scale

20) Meeting name: IEEE International Electron Device Meeting (IEDM)
Location: Virtual
Date: December 13, 2020
Attendees from this project: Hai Li
Presentation: Short course on “Alternate Technologies for SRAM”

21) Meeting name: IEEE International Solid-State Circuits Conference (ISSCC)
Location: Virtual

Date: February 21, 2021
Attendees from this project: Hai Li
Presentation: Forum talk on “Efficient Machine Learning: Algorithms-Circuits-Devices Co-design”

22) Meeting name: IEEE International Symposium on Circuits and Systems 2021
Location: Virtual
Date: May 25, 2021
Attendees from this project: Bokyung Kim
Presentation: Lecture on “An Efficient 3D ReRAM Convolution Processor Design For Binarized Weight Networks”

23) Meeting name: IEEE International Symposium on Circuits and Systems 2021
Location: Virtual
Date: May 27, 2021
Attendees from this project: Brady Taylor
Presentation: Lecture on “1S1R-Based Stable Learning Through Single-Spike-Encoded Spike-Timing-Dependent Plasticity”

24) Meeting name: Proceedings of the 38th International Conference on Machine Learning (ICML)
Location: Virtual
Date: July 20, 2021
Attendees from this project: Eric Yeats
Presentation: Lecture on “Improving Gradient Regularization using Complex-Valued Neural Networks”

APPENDIX C – ABSTRACT

In this project, we designed and fabricated a neuromorphic system controller in TSMC 65 nm technology for integration with memristive crossbar arrays. The design is capable of driving 1T1R or 1S1R arrays for implementation of spiking neuromorphic systems, using a digital pulse-width modulation-based input scheme as well as an integrate-and-fire circuit connected to a digital counter for the output scheme. Furthermore, the controller design can be tiled to drive arbitrarily large arrays and flexibly assigned to different layers in a multilayer neural network design. The chips can be integrated on a PCB or in-package and utilized by an off-chip processor or FPGA.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

1S1R	One Selector One Resistor
1T1R	One Transistor One Resistor
AFRL	Air Force Research Lab
BL	Bitline
BSB	Brain-State-in-a-Box
CA	Current Amplifier
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
FPGA	Field Programmable Gate Array
HSR	Highest Resistive State
IFC	Integrate-and-Fire Circuit
LSB	Least Significant Bits
LSR	Lowest Resistive State
LTD	Long Term Depression
LTP	Long Term Potentiation
MSB	Most Significant Bits
MTJ	Magnetic Tunnel Junctions
PCB	Printed Circuit Board
PGA	Pin Grid Array
PWD	Passive Weight Decay

PWM	Pulse Width Modulation
ReRAM	Resistive Random Access Memory
RMSE	Root Mean Square Error
SL	Source-line
SRAM	Static Random Access Memory
STDP	Spike Timing Dependent Plasticity
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
WL	Wordline