Application of Resistive Random Access Memory (RRAM) For Non-Von Neumann Computing

By

Sarah Rafiq

A Dissertation
Submitted to SUNY Polytechnic Institute
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

College of Nanoscale Science and Engineering

2022
To my loving family and friends
User Authorization

The IEEE publications listed below are discussed in this dissertation, which are contributions of the research conducted in this dissertation, discussed in Sections 4.2, 5.2, 5.3.1 and 5.4.2, for which I am the primary author:


In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of College of Nanoscale Science and Engineering, SUNY Polytechnic Institute's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.
Acknowledgements

First and foremost, my sincere and deepest gratitude goes to my Ph.D. advisor Dr. Nathaniel Cady. I am extremely grateful for the immense help and support I have received from him throughout my journey as a Ph.D. student, who has helped me to foster and develop my full potential by overcoming many obstacles in my research projects. He has always given me the support I needed to make progress in my research projects. His feedback and suggestions have been invaluable in improving my technical writing and presentation skills on my research projects. I am very grateful for the opportunity to work for such a wonderful Ph.D. advisor like Dr. Nathaniel Cady.

I would like to acknowledge that the funding of my research projects has been provided by the National Science Foundation (NSF). I am also very grateful for my former and current colleagues for their help and training. I would like to thank these individuals: my colleagues for their continuous unwavering support, Dr. Josh Holt for the hours of training on the SEM and Dr. Karsten Beckmann for his immense help in fabrication efforts of 300mm wafers. I would also like to thank my collaborators at University of Texas at San Antonio (UTSA), for their contribution in developing crossbar designs for approximate Boolean functions.

Last but not least, I would like to extend my gratitude to my family, especially my husband, who has supported me relentlessly throughout my Ph.D. journey, in providing continuous support to help me achieve my dreams. I am very grateful for the support I received from my parents, parents in-law and my little sisters, in completing this remarkable journey.
Abstract

The movement of data between physically separated memory and processing units in conventional computing systems (the so-called von Neumann architecture) incurs significant costs in energy and latency. This is known as the von Neumann bottleneck. With the advent of the Internet of Things (IoT) and edge computing, computing systems are also becoming significantly power limited. In this work, hafnium oxide resistive random access memory (ReRAM) integrated with 65nm CMOS technology on a 300 mm wafer platform was assessed to carry out two novel non-von Neumann computing applications that process data within memory and avoid excessive data movement. These computing applications are based on regulating the flow of sneak path currents in memory arrays to perform computation, called flow-based computing, and detecting degree of association (correlation) between binary processes in an unsupervised manner using the ReRAM non-volatile accumulative behavior, termed as temporal correlation detection. Electrical characterization of hafnium oxide ReRAM arrays was conducted for multi-level resistance states for flow-based computing, which was then investigated for two functions, approximate edge detection and XOR Boolean logic, through both experiments and simulation. The effect of device non-idealities was also evaluated. A trade-off between the flow-based output resistance ratio and the variability of flow-based outputs was found for different patterned binary resistance $R_{on}/R_{on}$ ratios. For the second non-von Neumann application, the feasibility of ReRAM as a non-volatile candidate device was investigated with an empirical ReRAM model through simulation. Experimental ReRAM analog incremental switching data, from both SET and RESET regimes, was also evaluated on the modified temporal correlation detection algorithm, where the RESET regime resulted in better performance. The ReRAM based implementation yielded 36,000-53,000
times lower energy consumption than similar implementation with phase change memory for 25 binary processes, and a speed-up of computation time by 1,600-2,100 times than that of a CPU-based implementation using 1xPOWER8 CPU. 1xPOWER8 CPU is a CPU available on the IBM* Power* System S822LC system, the POWER8 system series, where the CPU was run for 1 thread. In summary, hafnium oxide ReRAM based on 65nm CMOS technology has been evaluated for two non-von Neumann computing applications, and the effect of device non-idealities has also been assessed. These ReRAM in-memory computing applications show the promising potential of ReRAM in overcoming the von-Neumann bottleneck.
Table of Contents

List of Acronyms ........................................................................................................................................... x

List of Tables ............................................................................................................................................... xiii

List of Figures .............................................................................................................................................. xiv

Chapter 1 Introduction ........................................................................................................................................ 1
  1.1. Introduction to Emerging Non-Volatile Memory Technologies .......................................................... 1
    1.1.1. Phase Change Memory ................................................................................................................... 3
    1.1.2. Spin Transfer Torque Random Access Memory (STTRAM) ......................................................... 5
    1.1.3. Ferroelectric Random Access Memory (FeRAM) and Ferroelectric FET (FeFET) ....................... 6
    1.1.4. Resistive Random Access Memory (ReRAM) .............................................................................. 9
  1.2 Applications of Emerging Non-Volatile Memory Technologies ........................................................... 14
    1.2.1. Storage Class Memory (SCM) ...................................................................................................... 14
    1.2.2. Hardware Security Primitives ..................................................................................................... 15
    1.2.3 In-Memory Computing .................................................................................................................. 16
  1.3. Summary ............................................................................................................................................... 24
  1.4. References ............................................................................................................................................ 25

Chapter 2 Problem Statement and Thesis Organization .................................................................................. 39
  2.1. Problem Statement ............................................................................................................................... 39
  2.2. Thesis Organization ............................................................................................................................. 42
  2.3. References ........................................................................................................................................... 44

Chapter 3: Investigation of HfO₂ ReRAM Device Characteristics for Non-Von Neumann Computing Applications by Electrical Characterization of 1T1R Arrays .................................................. 46
  3.1. Introduction to Multi-Level Resistance States ..................................................................................... 46
  3.2. Fabrication of ReRAM Devices .......................................................................................................... 47
  3.3. Electrical Characterization and Operation of ReRAM Devices ......................................................... 49
  3.4. Operation of ReRAM Devices ............................................................................................................ 53
  3.5. Multi-Level Resistance States with Current Control ............................................................................ 56
3.6. Conclusion ...........................................................................................................................65
3.7. Acknowledgment .................................................................................................................65
3.8. References ...........................................................................................................................66

Chapter 4 Temporal Correlation Detection in ReRAM Arrays................................................73
4.1. Introduction ........................................................................................................................73
4.2. Temporal Correlation Detection using ReRAM arrays .......................................................76
  4.2.1. Generation of Correlated and Uncorrelated Processes .................................................77
  4.2.2. Temporal Correlation Detection on ReRAM Arrays using the ReRAM Model .......79
  4.2.3. Temporal Correlation Detection on ReRAM Arrays using ReRAM Experimental Data ............................................................................................................................................... 89
4.3 Conclusion .............................................................................................................................111
4.4 Acknowledgment ..................................................................................................................113
4.5 References ............................................................................................................................113

Chapter 5 Flow-Based Computing on 65 nm CMOS Integrated HfO₂ based ReRAM Arrays. 117
5.1. Introduction ........................................................................................................................117
  5.1.1. Background ..................................................................................................................118
5.2. Electrical Characterization Setup .....................................................................................121
5.3. Methodology ......................................................................................................................126
  5.3.1. Methodology of Flow-Based Edge Detection ...........................................................126
  5.3.2. Methodology of Flow-Based XOR Boolean Logic ...................................................131
5.4. Results and Discussion .....................................................................................................133
  5.4.1. XOR Boolean Logic Results and Discussion ............................................................133
  5.4.2. Edge Detection Results and Discussion ....................................................................136
5.5. Conclusion ..........................................................................................................................144
5.6. Comparison to Related Works ........................................................................................146
  5.6.1. XOR Boolean Logic .................................................................................................146
  5.6.2. Approximate Edge Detection ...................................................................................151
5.7. Acknowledgment .................................................................................................................152
5.8. References ..........................................................................................................................152

Chapter 6 Conclusion and Future Outlook ...........................................................................160
6.1. Conclusion of Dissertation .................................................................................................160
6.2. Future Outlooks .................................................................................................................163
6.3. References .........................................................................................................................165
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1T-1C</td>
<td>1-transistor 1-capacitor</td>
</tr>
<tr>
<td>1T1R</td>
<td>1-transistor 1-ReRAM</td>
</tr>
<tr>
<td>ALD</td>
<td>Atomic layer deposition</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial neural network</td>
</tr>
<tr>
<td>BE</td>
<td>Bottom electrode</td>
</tr>
<tr>
<td>BEOL</td>
<td>Back-end-of-the-line</td>
</tr>
<tr>
<td>BDD</td>
<td>Binary decision diagrams</td>
</tr>
<tr>
<td>BJT</td>
<td>Bipolar junction transistor</td>
</tr>
<tr>
<td>BL</td>
<td>Bit line</td>
</tr>
<tr>
<td>CF</td>
<td>Conductive filament</td>
</tr>
<tr>
<td>CRS</td>
<td>Complementary resistive switching</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary metal oxide semiconductor</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional neural network</td>
</tr>
<tr>
<td>DC</td>
<td>Direct current</td>
</tr>
<tr>
<td>DPC</td>
<td>Depression-potentiation cycle</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic random access memory</td>
</tr>
<tr>
<td>DT</td>
<td>Direct tunneling</td>
</tr>
<tr>
<td>DUT</td>
<td>Device under test</td>
</tr>
<tr>
<td>ECM</td>
<td>Electrochemical metallization mechanism</td>
</tr>
<tr>
<td>FNT</td>
<td>Fowler-Nordheim tunneling</td>
</tr>
<tr>
<td>FBDD</td>
<td>Free binary decision diagrams</td>
</tr>
<tr>
<td>FeFET</td>
<td>Ferroelectric field effect transistor</td>
</tr>
<tr>
<td>FeRAM</td>
<td>Ferroelectric random access memory</td>
</tr>
<tr>
<td>FTJ</td>
<td>Ferroelectric tunnel junctions</td>
</tr>
<tr>
<td>GL</td>
<td>Gate line</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>GST</td>
<td>Germanium-antimony-tellurium</td>
</tr>
<tr>
<td>IMPLY</td>
<td>Material implication</td>
</tr>
<tr>
<td>HRS</td>
<td>High resistance state</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet-of-Things</td>
</tr>
<tr>
<td>ICOMP</td>
<td>Current compliance</td>
</tr>
<tr>
<td>LRS</td>
<td>Low resistance state</td>
</tr>
<tr>
<td>MFS</td>
<td>Metal-ferroelectric-Si</td>
</tr>
<tr>
<td>MIM</td>
<td>Metal-insulator-metal</td>
</tr>
<tr>
<td>MLC</td>
<td>Multi-level-cell</td>
</tr>
<tr>
<td>MOS</td>
<td>Metal oxide semiconductor</td>
</tr>
<tr>
<td>MRAM</td>
<td>Magnetic random access memory</td>
</tr>
<tr>
<td>MRL</td>
<td>Memristor ratioed logic</td>
</tr>
<tr>
<td>MTJ</td>
<td>Magnetic tunnel junction</td>
</tr>
<tr>
<td>MW</td>
<td>Memory window</td>
</tr>
<tr>
<td>NFET</td>
<td>N-type field effect transistor</td>
</tr>
<tr>
<td>NMOS</td>
<td>N-type metal oxide semiconductor</td>
</tr>
<tr>
<td>NVM</td>
<td>Non-volatile memory</td>
</tr>
<tr>
<td>OEL</td>
<td>Oxygen exchange layer</td>
</tr>
<tr>
<td>PCM</td>
<td>Phase change memory</td>
</tr>
<tr>
<td>PCRAM</td>
<td>Phase change random access memory</td>
</tr>
<tr>
<td>PL</td>
<td>Plate line</td>
</tr>
<tr>
<td>PVD</td>
<td>Physical vapor deposition</td>
</tr>
<tr>
<td>PZT</td>
<td>Lead zirconate titanate</td>
</tr>
<tr>
<td>RF</td>
<td>Radio frequency</td>
</tr>
<tr>
<td>RIE</td>
<td>Reactive ion etch</td>
</tr>
<tr>
<td>ROBDD</td>
<td>Reduced ordered binary decision diagram</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>ReRAM</td>
<td>Resistive random access memory</td>
</tr>
<tr>
<td>SBT</td>
<td>Strontium bismuth Tantalate</td>
</tr>
<tr>
<td>SCM</td>
<td>Storage class memory</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SILC</td>
<td>Stress induced leakage current</td>
</tr>
<tr>
<td>SIMS</td>
<td>Secondary ion mass spectroscopy</td>
</tr>
<tr>
<td>SL</td>
<td>Switching layer</td>
</tr>
<tr>
<td>SMU</td>
<td>Source measurement unit</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static random access memory</td>
</tr>
<tr>
<td>SSD</td>
<td>Solid state drive</td>
</tr>
<tr>
<td>STTRAM</td>
<td>Spin-torque transfer resistive random access memory</td>
</tr>
<tr>
<td>TAT</td>
<td>Trap-assisted-tunneling</td>
</tr>
<tr>
<td>TE</td>
<td>Top electrode</td>
</tr>
<tr>
<td>TEM</td>
<td>Transmission electron microscopy / microscope</td>
</tr>
<tr>
<td>TER</td>
<td>Tunneling electroresistance</td>
</tr>
<tr>
<td>TMR</td>
<td>Tunneling magnetoresistance</td>
</tr>
<tr>
<td>VCM</td>
<td>Valence change mechanism</td>
</tr>
<tr>
<td>VMM</td>
<td>Vector-matrix multiplication</td>
</tr>
<tr>
<td>WGFMU</td>
<td>Waveform generator fast measurement unit</td>
</tr>
</tbody>
</table>
List of Tables

Table 5-1: Truth table of Boolean XOR logic, and the expected output resistance for flow-based XOR for each input configuration, A and B. The output logic 1 and logic 0 correspond to low resistance output and high resistance output, respectively. ........................................................... 134

Table 5-2: The experimental results of flow-based XOR is reported. For each input configuration, the binary logic of each cell in the 2x2 1T1R array and their respective resistance states are shown, where logic 1 and logic 0 are mapped to low resistance state (LRS) and high resistance state (HRS) of the 1T1R, respectively. The experimental results for each input configuration of XOR confirms the expected logic outputs, where the flow-based XOR gave high resistance when XOR output is logic 0 and low resistance when the XOR output is logic 1. ........................................................ 134

Table 5-3: The experimental and simulation results of flow-based XOR are reported for each input configuration. The experimental results are validated by the simulation of flow-based XOR. 136

Table 5-4: The values of pixel a and pixel b in each of the five patterns are shown below. The expected output logic of 1 corresponds to the presence of an edge between the 2 pixels and is expected to contribute to low resistance in the edge detection test. Similarly, the expected output logic of 0 implies the absence of an edge and therefore would result in high resistance in the output of edge detection test. The patterns consist of 3 low resistance patterns and 2 high resistance patterns. © 2021 IEEE, Reprinted, with permission. ........................................................................................................ 137

Table 5-5: Comparison of proposed flow-based XOR to other approaches using ReRAM array, in terms of number of ReRAM cells used and total number of sequential steps in the XOR operation. ........................................................................................................ 147
List of Figures

Figure 1- 1: The memory hierarchy, comprised of volatile and non-volatile memory. Adapted from [4]........................................................................................................................................................3

Figure 1- 2: The cross-sectional view of a phase-change memory (PCM) cell. The switching material undergoes phase transition between crystalline (green) and amorphous (yellow) phases, leading to the low resistance state and the high resistance state of the PCM device, respectively. Figure Adapted from [10]. .............................................................................................................. 4

Figure 1- 3: The schematic of a magnetic tunnel junction (MTJ) in spin transfer torque random access memory (STTRAM) cell [adapted from [3]]. In the free layer, the ferromagnetic orientation in parallel to the pinned layer (shown in green) results in low resistance of the STTMRAM and the orientation shown antiparallel (shown in red) results in high resistance of the STTRAM. 6

Figure 1- 4: The schematic of a ferroelectric random access memory (FeRAM) in the 1T-1C configuration. The polarization direction in the ferroelectric layer can be reversed through the application of electric field, by applying voltages to the bit line and plate line to write ‘0’ and ‘1’ states, respectively [adapted from [16]]........................................................................................................... 7

Figure 1- 5: Schematic of resistive random access memory (ReRAM) based on the valence change mechanism (VCM). The conductive filament (CF) in the metal-oxide switching layer is first created during an initial electroforming process. The CF is comprised of oxygen vacancies as shown by the green circles, and conducts current between the top and bottom electrodes, giving the device a low resistance state. Once the CF has been formed, it can be partially dissolved and recovered through a RESET process and a SET process, respectively, to switch the ReRAM between the high resistance state and the low resistance state. During RESET, the oxygen ions shown in red circles recombine with oxygen vacancies to disrupt the CF................................................. 13

Figure 1- 6: Schematic of cross-section of resistive random access memory (ReRAM) based on HfO2 ReRAM, that will be discussed in this work. The ReRAM top and bottom electrodes are titanium nitride (TiN), the switching oxide layer is hafnium dioxide (HfO2) and the oxygen exchange layer (OEL) is titanium (Ti) ...................................................................................................................... 14

Figure 1- 7: The vector-matrix multiplication (VMM) on a 1T1R array. The blue arrows refer to the voltages V1-V8, applied to the respective rows 1-8 for the VMM operation. The dashed red lines show individual current along rows and the solid red line shows the summed-up current. 18

Figure 1- 8: A multi-layer perceptron neural network consisting of 3 layers, one input, one hidden and one output layer. The circles represent neurons, and each connection in-between them
represents a NVM synapse. Arrays of NVM are used in conjunction with CMOS circuitry to implement neural networks. Forward propagation is shown here...........................20

Figure 1- 9: a) When the excitatory inputs from each of the five input neurons (shown in different colors) in first layer are applied to a neuron as voltages, the VMM weighs the incoming signals according to the weight of each synapse (array element) by Ohm’s law, and Kirchhoff’s current law sums up the current down the column to give a weighted sum. b) The portion of neural network where the VMM is taking place, shown are five neuron excitations being weighted by five synapses to be output to post-neuron Y. c) A schematic of the VMM output of the weighted sum, which is then processed by a function to determine the excitation of neuron Y. [adapted from 55] 21

Figure 2- 1: The von-Neumann architecture, consisting of separate processing unit and main memory unit. The processing unit contains comprises of both control unit and arithmetic logic unit (ALU). Instructions and data are stored in memory, which are fetched using the bus. For processing, instructions are fetched and decoded. Then, data is loaded from memory into cache or registers, and computed upon by the ALU. The result of computation is then stored back to the memory. Data needs to be transferred back and forth in this architecture .........................41

Figure 3- 1: A transmission electron micrograph (TEM) of the fabricated 1-transistor-1-ReRAM (1T1R) device, and the cross-section of the ReRAM device showing the device stack composition, the TiN top electrode, the Ti oxygen exchange layer (OEL), the HfO2 switching layer and the TiN bottom electrode. © 2021 IEEE, Reprinted, with permission...............................................................48

Figure 3- 2: A typical pulse train of FORMING, RESET, SET, READ voltage pulses applied for bipolar hafnium-oxide ReRAM operation. The FORMING pulse is applied once to pristine device, and the switching cycle consisting of RESET, READ, SET, READ are applied repeatedly for switching the ReRAM between the low resistance state (LRS) and the high resistance state (HRS). The READ pulse is applied to read the programmed resistance after FORM, SET and RESET operations. ......................................................................................................................................51

Figure 3- 3: a) Voltage pulses for FORM, RESET, SET and READ operations are applied to the top electrode of ReRAM. The ReRAM is connected in series with the drain of the transistor, the source of the transistor is grounded and the gate voltage is applied to control the current compliance b) drain current vs. drain voltage, I_d -V_d characteristics of integrated NFET transistor for drain voltage sweeps at various gate voltages, yielding a range of saturation currents that can provide the current compliance needed during the 1T1R switching. ........................................................................................................52

Figure 3- 4: The valence change mechanism (VCM) of bipolar metal-oxide ReRAM. During FORMING, a conductive filament of oxygen vacancies is formed in the metal-oxide layer. During RESET operation, negative voltage is applied to top electrode to push oxygen ions and undergo
recombination to partially dissolve the conductive filaments. During SET operation, positive voltage re-connects the conductive filaments to conduct current. ..................................................................................54

Figure 3- 5: The pulsed I-V measurements of ReRAM FORMING, RESET and SET. The ReRAM forms at a voltage of 2.3V, and undergoes RESET and SET operations at 0.75V and -0.85V, respectively. The applied FORMING, RESET and SET voltages were 3V, -1.5V and 2.5V respectively..........................................................................................................................56

Figure 3- 6: The schematic of the 8x8 1-transistor-1-ReRAM (1T1R) array structures. Each 1T1R array element can be accessed through its drain, source and gate lines. The 1T1R FORM, SET, RESET, READ operations are the same as mentioned previously. © 2021 IEEE, Reprinted, with permission. ..........................................................................................................................57

Figure 3- 7: The pulsed I-V measurements of ten ReRAM RESET and SET cycles at six different current compliances, ranging from 50µA to 400 µA. The current compliance regulates the maximum SET current for each switching cycle. The pulse IV switching cycles are uniform for each set of conditions. The applied RESET and SET voltages were -1.5V and 2.5V respectively. Higher current compliance therefore results in lower LRS resistance states .......................58

Figure 3- 8: The evolution of the conductive filament consisting of oxygen vacancies, with respect to increasing current compliance during SET operations, shown from left to right. The conductive filament widens laterally with respect to an increase in current compliance, and results in further reduction of the low resistance state R_{LRS}. The R_{LRS} can be controlled via modulation of current compliance .....................................................................................................................................59

Figure 3- 9: Cumulative frequency distributions of a) intra-array LRS resistance values within an 8x8 1T1R array and b) inter-array LRS resistance values for 8x8 1T1R arrays across a die with respect to current compliance. The current compliance was controlled by varying gate voltage of the integrated transistor. ..........................................................................................................................59

Figure 3- 10: Cumulative frequency distributions of inter-array LRS resistance values for 8x8 1T1R arrays across a die with respect to current compliance. The current compliance was controlled by varying the gate voltage of the integrated transistor..................................................................................................................63

Figure 3- 11: Box-and-whisker plot for the die-to-die resistance variation in 8x8 1T1R array devices across a 300mm wafer, with respect to current compliance (by varying gate voltage of the control transistor). The standard deviation of the LRS resistance decreased with increasing current compliance .....................................................................................................................................63

Figure 3- 12: a) NSF logo programmed with binary LRS resistance values, using current compliance of 60 µA and 400 µA. d) A greyscale image was programmed into 8x8 1T1R array
using current compliance from 175 μA to 400 μA. Each LRS resistance value corresponds to intensity of pixel in image .................................................................................................................................64

Figure 4- 1: Schematic illustration of an all-memristive computational primitive, adapted from [19]. PCM devices are used for both the neuronal membrane potential and the synapses. The weight of synapse represents the strength of the connection between two neurons. The incoming input spikes to the synapses (orange dots), the array of PCM used as synapses (orange box), the integrate-and-fire neuron (green box), and the postsynaptic spikes (blue dots) are shown. Insets show (a) the neuronal membrane potential of primary neuron and (b) the spike-timing dependent plasticity (STDP) learning rule, where $\Delta w$ (y-axis) represents the change in the weight of STDP synapse based on the timing of the presynaptic spike time and the postsynaptic spike time (x-axis). The $\Delta t$ in the x-axis represents the $(t_{PRE} - t_{POST})$ of the presynaptic and postsynaptic spike events. The LTP (green) and LTD (purple) are the long term potentiation and depression curves respectively, where potentiation refers to increase in weight of synapse and depression refers to decrease in weight of synapse. Adapted from [5]. .............................................................................................................75

Figure 4- 2: Algorithm of the simulated temporal correlation detection with ReRAM model. © 2020 IEEE, Reprinted, with permission. .........................................................................................................................82

Figure 4- 3: The conductance of the 5x5 ReRAM array at time instant k=5. © 2020 IEEE, Reprinted, with permission. ..................................................................................................................................................85

Figure 4- 4: The conductance of the 5x5 ReRAM array at time instant k=20. © 2020 IEEE, Reprinted, with permission. ..................................................................................................................................................85

Figure 4- 5: The conductance of the 5x5 ReRAM array at time instant k=50. © 2020 IEEE, Reprinted, with permission. ..............................................................................................................................................86

Figure 4- 6: The conductance of the 5x5 ReRAM array at time instant k=70. © 2020 IEEE, Reprinted, with permission. ..............................................................................................................................................86

Figure 4- 7: Block diagram of the simulation of the algorithm on a 5x5 ReRAM array in Python. Each module is a class of specific attributes and functions, to generate the processes, to control the algorithm (assign the processes, compute momentum, compute $V(t)$ according to momentum, etc.), the 5x5 ReRAM array, the ReRAM model used and an interface to provide communication between the ReRAM array module and the module implementing the algorithm. .........................................................................................89

Figure 4- 8: Analog switching data using ultra-short 300ps pulses of 200 RESET pulses followed by 200 SET pulses, for four cycles. The four cycles for RESET and SET were denoted by R1-R4 and S1-S4, respectively, using different colors. The SET and RESET voltages used are 1V and -0.75V, respectively, of 300ps duration. After each SET/RESET pulse, a read-verify approach was used to measure the intermediate resistance states. ..........................................................................................92
Figure 4-9: Algorithm of the simulated temporal correlation detection with reset cycles R1-R4 of the ReRAM analog switching data using 200 RESET pulses of 300ps pulse width.

Figure 4-10: Algorithm of the simulated temporal correlation detection with set cycles S1-S4 of the ReRAM analog switching data using 200 SET pulses of 300ps pulse width.

Figure 4-11: The ideal results with temporal correlation detection algorithm on a 5x5 ReRAM array using RESET analog switching data. Correlated processes are assigned to the first two rows of the array. The rest of the array are assigned uncorrelated processes. The ReRAM devices at these correlated processes, represented by the blue squares, are expected to go from high conductance to low conductance, while the rest of the array, represented by yellow squares, are expected to have higher conductance than the first two rows of the array at the end of the algorithm. In other words, the blue and yellow boxes represent devices at low and high conductance, respectively, at final time instance kfinal.

Figure 4-12: Conductance of 5x5 ReRAM array at k=500 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data, the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes; d) Conductance of 5x5 ReRAM array at k=500 using analog R4 reset cycle data.

Figure 4-13: Conductance of 5x5 ReRAM array at k=600 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data d) R4 reset cycle data. For (a) and (d), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R1 and R4 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.

Figure 4-14: Conductance of 5x5 ReRAM array at k=900 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data and d) R4 reset cycle data. For (b), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R2 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.

Figure 4-15: Conductance of 5x5 ReRAM array at k=1200 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data and d) R4 reset cycle data. For (b), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R2 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.
Figure 4-16: Conductance of 5x5 ReRAM array at k=1300 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data and d) R4 reset cycle data. For (c), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R3 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.

Figure 4-17: The ideal results with temporal correlation detection algorithm on a 5x5 ReRAM array using SET analog switching data. Ten correlated processes are assigned to the first two rows of the array, and the rest of the array are assigned uncorrelated processes. The ReRAM devices at these correlated processes, represented by the yellow boxes, are expected to go from low conductance to high conductance, while the rest of the array, represented by blue boxes, are expected to have lower conductance than the first two rows of the array at the end of the algorithm. In other words, the blue and yellow boxes represent devices at low and high conductance, respectively, at final time instance kfinal.

Figure 4-18: Conductance of 5x5 ReRAM array at k=130 using analog a) S1 SET cycle data b) S2 SET cycle data c) S3 SET cycle data and d) S4 SET cycle data. For (a), the conductance of 9 correlated processes are lower than the conductance of the uncorrelated processes, since SET data was used. Thus, the temporal correlation detection algorithm worked 90% with S1, 50% with S2 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.

Figure 4-19: Conductance of 5x5 ReRAM array at k=317 using analog a) S1 SET cycle data b) S2 SET cycle data c) S3 SET cycle data and d) S4 SET cycle data. For (b), the conductance of all correlated processes are lower than the conductance of the uncorrelated processes, since SET data was used. Thus, the temporal correlation detection algorithm worked with S2 analog data as it resulted in greater absolute change in conductance for all the correlated processes compared to the uncorrelated processes.

Figure 4-20: Conductance of 5x5 ReRAM array at k=1200 using analog a) S1 SET cycle data b) S2 SET cycle data c) S3 SET cycle data and d) S4 SET cycle data. For (c), the conductance of all correlated processes are lower than the conductance of the uncorrelated processes, since SET data was used. Thus, the temporal correlation detection algorithm worked with S3 analog data as it resulted in greater absolute change in conductance for all the correlated processes compared to the uncorrelated processes.

Figure 5-1: The read current flowing through desired ReRAM cell is shown in blue while the sneak path current flowing in parallel to device being read is shown in red on the memory crossbar array. Sneak path currents would contribute to the read current of desired read cell, giving an incorrect readout. Hence, they are an issue in passive crossbar arrays.
Figure 5-2: Illustration of how a 1-transistor-1-ReRAM (1T1R) array can be used to both accurately program 1T1R device and accommodate sneak path currents. The blue line shows the programming current flowing though device when its transistor is biased but the other transistors are turned off, so that transistor provides isolation and enables accurate programming. The red line shows the sneak path current flowing in the 1T1R array when all the gates of its transistors are biased simultaneously to give passive array. This is useful for flow-based computing.

Figure 5-3: a) The schematic and b) the layout of the 8x8 1-transistor-1-ReRAM HfO2 array. The array has 24 pads (2x12), consisting of eight source lines S1-S8, drain lines D1-D8 and gate lines G1-G8. For flow-based computing, the number of gates corresponding to flow-based nanoscale crossbar design were biased to give passive array and accommodate the flow of sneak paths.

Figure 5-4: a) Electrical characterization setup with the SUSS Microtech probe station, the E5270A parametric analyzer, Keithley 707A switching matrix and their communication controlled by the computer. The probe card is installed onto the probe station in the highlighted area. b) The 2x12 Celadon Ultra High Performance probe card and device under test (DUT), probing all the 24 pads of our 8x8 1T1R HfO2 array on the 300mm wafer. The inset shows the schematic and the transmission electron micrograph (TEM) of the 1T1R array element, ReRAM integrated with NFET at metal 1/via 1 interface, and the TiN/Ti/HfO2/TiN ReRAM device stack composition.

Figure 5-5: The 8x8 ReRAM crossbar design for flow-based edge detection, to detect edges between 8-bit pixels. Each circle represents one ReRAM at the intersection of row and column. The variables of edge detection consist of the bits of the two pixels, bits A0-A7 and B0-B7, of pixels A and B, respectively. The binary values of the bits are mapped to the ReRAM resistances, shown by blue circles, while other ReRAM are fixed at either logic 0 or logic 1, shown by gray and green circles respectively. The “!” represents the negation of the variables of pixels A and B. The logic 1 is mapped to Ron resistance state while the logic 0 is mapped to Roff resistance state. © 2021 IEEE, Reprinted, with permission.

Figure 5-6: The flow chart of the steps required to implement the flow-based edge detection. © 2021 IEEE, Reprinted, with permission.

Figure 5-7: The mapping scheme of the variables of pixels A and B, namely bits A0-A7 and B0-B7, onto the ReRAM array. Each bit of pixel is a Boolean logic 1 or 0, and depending on the crossbar design, the respective ReRAM of that variable will either hold the input bit or its negation if the variable is preceded by “!” . The rest of the array remains fixed at logic 1 and logic 0. Based on this scheme, the ReRAM assigned logic 1 and logic 0 are programmed to Ron resistance states (shown by green circles) and Roff resistance states (shown by gray circles), respectively. Once the array has been programmed, flow-based computing is performed by applying a read pulse to bottom row (input nanowire) and the output current is observed along last column (output...
Figure 5-8: a) The 2x2 crossbar design for flow-based XOR consisting of literals and negations of the input variables of Boolean XOR logic. b) Methodology of implementing Boolean XOR logic on a 2x2 1T1R array. After the array is configured according to input variables of XOR, A and B, the sneak path currents computing the flow-based XOR are shown in red. The input read pulse is applied to top row and output current is measured along last row of the 2x2 array.

Figure 5-9: The resistance states of the 2x2 1T1R cells mapped according to flow-based crossbar design, and the corresponding flow-based XOR computation for input configurations a) A=0, B=0 b) A=1, B=1 c) A=0, B=1 and d) A=1, B=0. The 1T1R devices at LRS and HRS are shown by the white and green circles respectively, and the flow-based current is shown in red. The read pulse is applied to top row and flow-based current is measured along bottom row. The flow-based output current is expected to be low for (a)-(b) as R_{off} states in each sneak path lowers current measured at bottom row. In contrast, flow-based output current is expected to be high for (c)-(d) as the R_{on} states are present in the same sneak path, promoting current from row 1 to column 1, and column 1 to row 2, where output current is measured.

Figure 5-10: Comparison of experimental edge detection outputs of 5 pixel-pairs (each pattern denotes a pixel-pair) with their ideal case LTSpice simulation using HRS and LRS patterned arrays. The result confirms that an edge is detected in the first three pixel pairs, and no edge is present between the last 2 pixel-pairs. The results were validated by LTSpice simulation assuming HRS of 100 kΩ and LRS of 3.5 kΩ, with no resistance variation. © 2021 IEEE, Reprinted, with permission.

Figure 5-11: The effect of three patterned R_{off}/R_{on} resistance ratios, 1.5:1, 2.5:1 and 28.6:1, shown left to right, on the five flow-based edge detection outputs for over 50 cycles. The five flow-based edge detection outputs consist of three low outputs (E.D. Patterns 1-3) and two high outputs (E.D Patterns 4-5). With increasing patterned R_{off} state, the ratio between the flow-based binary outputs improves by three-fold from 1.16:1 to 3.08:1, when HRS is used for patterned logic 0 resistance, as opposed to LRS states of 5.6 kΩ and 9 kΩ used for R_{off} state. Also, the variability of flow-based edge detection outputs increases with increasing variability of patterned R_{off} state (logic 0). Hence, there is a trade-off between the binary flow-based output ratio and the variability of the flow-based outputs. One-way ANOVA shows significant difference between the high and low flow-based edge detection outputs for all three patterned R_{off}/R_{on} resistance ratios, and hence flow-based edge detection worked for the three conditions.

Figure 5-12: The LTSpice Gaussian simulation results for effect of three patterned R_{off}/R_{on} resistance ratios, 1.5:1, 2.5:1 and 28.6:1, shown left to right, on the five flow-based edge detection outputs for over 200 cycles. The five flow-based edge detection outputs consist of three low outputs (E.D. Patterns 1-3) and two high outputs (E.D Patterns 4-5). With increasing patterned R_{off}
state, the high flow-based outputs increase, giving higher ratio between the binary flow-based outputs. Also, the variability of flow-based outputs increase with variability of patterned resistance. Hence, a trade-off exists between the binary flow-based output ratio and the flow-based outputs, and simulation conform to experimental results. © 2021 IEEE, Reprinted, with permission. ................................................................................................................................... 142

Figure 5-13: The Pearson correlation coefficient of the experimental edge detection outputs for 5 patterns and their LTSpice Gaussian simulation for the three patterned R_{off}/R_{on} ratios of 1.5:1, 2.5:1 and 28.6:1. The coefficient r of 0.9547 shows a strong positive correlation between the experimental data and the simulation. © 2021 IEEE, Reprinted, with permission.........................144
Chapter 1: Introduction

1.1. Introduction to Emerging Non-Volatile Memory Technologies

Memory is a fundamental component in today’s computing systems, where it is critical for both short-term or permanent (e.g. Read-Only Memory (ROM)) data storage. The memory hierarchy is shown in Figure 1-1. Memory can be classified into two types: the volatile memory and the non-volatile memory. For the former type of memory, the memory holds the data as long as power is supplied to the memory device, and loses the data when the power is switched off. The major types of volatile memory in use today are the static random access memory (SRAM) and dynamic random access memory (DRAM). SRAM has the fastest operating speed (about 1ns) among memory technologies, but it is very expensive and consumes a large footprint due to its six-transistor (6T) structure, making it lower density data storage. It is used in the cache [1]. DRAM, on the other hand, has smaller footprint than SRAM due to its series 1-transistor-1-capacitor structure, has considerable read/write speed but requires large programming voltages and is volatile. In addition, capacitor leakage issues [2] pose a limit on the data storage capacity. As DRAM is scaled to increase its density, it incurs more power consumption to refresh the data [3].

The latter form of memory, non-volatile memory, is a memory that retains its data even after the power supply is switched off. Both volatile and non-volatile memory can be written into, and be read from. Examples of non-volatile memory includes Flash NAND and NOR, where the building block consists of a floating gate transistor. In Flash NAND, the transistors are arranged in series where the NAND output is low only when all its inputs are logic 1. In Flash NOR, the transistors are arranged in parallel. The charge stored in the floating gate represents the data and the transistor acts as a selector, both are in one device [2]. Flash memory can be stacked in 3D to
form high density data storage, but requires long programming/read times (about milliseconds), which increases its programming power [4]. SRAM and DRAM have much lower density than flash technology. All SRAM, DRAM and Flash NAND are based on complementary metal oxide semiconductor (CMOS) technology, which is hitting its scalability limits due to the end of Moore’s law [1]. Nowadays, the market trends demand high speed (high performance), high density and low cost for data storage. Additionally, data-intensive applications, mobile systems and Internet-of-Things (IoT) demand ultra-low-power systems. These requirements cannot be fulfilled by the current memory technologies (SRAM, DRAM, Flash NAND) alone. This paves the way for emerging non-volatile memory technology that can achieve ultra-low programming power, high speed, high density, low cost and good scalability. The major emerging non-volatile memory technologies are phase change memory (PCM), spin transfer torque magnetic random access memory (STTRAM), resistive random access memory (ReRAM), ferroelectric random access memory (FeRAM) and ferroelectric field effect transistor (FeFET) memory. The following sections will review each of these memory technologies.
1.1.1. **Phase Change Memory**

Phase change memory (PCM), also known as phase change random access memory (PCRAM), is the most mature of the emerging memory technologies. It uses the phase transition properties of chalcogenides. For example, an alloy of germanium, antimony, and tellurium can be used to store data as resistance [5]. The Ge-Sb-Te (GST) material is intensively studied as a material for PCM [4]. The chalcogenide material is sandwiched between two metal electrodes. The phase transition is induced by Joule heating, where the material’s phase transition from amorphous to crystalline
and from crystalline to amorphous, alters its resistance from high resistance to low resistance and low resistance to high resistance, respectively. PCM devices switch between the low resistance and high resistance using the same bias direction, (i.e., they are unipolar devices). The reset transition (to high resistance) requires larger currents than set transition (to low resistance) and is, therefore, high in power consumption during RESET. To RESET PCM, a large current pulse with small pulse width is applied to melt the crystalline material and is abruptly stopped to rapidly quench the switching material to room temperature to make the amorphous phase, which gives the PCM a high resistance [6]. To SET the PCM device, a long low amplitude pulse is applied to recrystallize the GST material to the crystalline phase, giving the low resistance of PCM. This is the speed-determining step [2]. During fabrication, the pristine PCM device is at a crystalline phase and has low resistance due to high temperature processing conditions [7].

Figure 1-2: The cross-sectional view of a phase-change memory (PCM) cell. The switching material undergoes phase transition between crystalline (green) and amorphous (yellow) phases, leading to the low resistance state and the high resistance state of the PCM device, respectively. Figure Adapted from [10].
In general, PCM devices have exceeded the flash NAND in terms of programming voltages, programming/read times and endurance [8]. PCM can achieve up to \(10^9\) switching endurance, and 100ns write/erase times. However, it suffers from high RESET currents, high switching power, high write latency [2, 9], and cost of fabrication [10]. It is difficult to achieve both high speed switching and stable retention at high temperature for PCM [9], where higher programming time is required for better data retention at high temperature for PCM devices. There are ongoing efforts to optimize the power efficiency by overcoming the short-comings mentioned [2]. The scalability of the size of PCM is also limited by the selector devices used for PCM, such as bipolar junction transistors (BJTs) and diodes [2], [4].

1.1.2. Spin Transfer Torque Random Access Memory (STTRAM)

Spin transfer torque random access memory (STTRAM) is a category of magnetoresistive random access memory (MRAM), which consists of a magnetic tunnel junction (MTJ) in series with a transistor in 1T-1MTJ configuration [2]. The structure of STTRAM is depicted in Figure 1-3. Due to small size of nanopillar MTJ, the STTRAM uses low spin-polarized current (~100\(\mu\)A) in the WRITE operation to switch its state, which gives better performance than MRAM, in terms of scalability and efficiency [11]. The MTJ has two ferromagnetic layers, where one has a fixed magnetic orientation and the other’s magnetic orientation is free to rotate. The two layers are separated by an ultra-thin (1-2nm) layer of tunnel oxide barrier, usually MgO [4]. When the two ferromagnetic layers are aligned, the STTRAM is at a low resistance state. Meanwhile, when the two ferromagnetic layers are anti-parallel, the STTRAM is at a high resistance state. The dipoles can be switched through the application of spin polarized current [12].
STTRAM has the highest speed (about 10ns) among the emerging non-volatile memory technologies, which is faster than DRAM, flash memory and PCM technology. It also has an excellent endurance of up to $10^{12}$ switching cycles, which exceeds PCM and flash memory [2]. It has a higher write energy than read energy, and with scaling, its read current does not scale with lower write currents, which gives rise to read disturbance errors [3]. Furthermore, its density is not higher than DRAM and has a higher write energy than DRAM [11]. It is also very sensitive to fabrication process parameters [4]. On the other hand, its tunnel magnetoresistance ratio, which is defined by $(R_{oFF}-R_{on})/R_{on}$, is very small ($\sim 1.5$) and is sensitive to parametric process variations [13].

![Figure 1- 3: The schematic of a magnetic tunnel junction (MTJ) in spin transfer torque random access memory (STTRAM) cell [adapted from [3]]. In the free layer, the ferromagnetic orientation in parallel to the pinned layer (shown in green) results in low resistance of the STTMRAM and the orientation shown antiparallel (shown in red) results in high resistance of the STTRAM.](image)

1.1.3. Ferroelectric Random Access Memory (FeRAM) and Ferroelectric FET (FeFET)

Ferroelectric RAM (FeRAM or FRAM) is a non-volatile memory, where the dielectric of the capacitor has ferroelectric properties. This means that the dielectric has spontaneous
polarization (defined as electric dipole moment per unit volume), where the direction of polarization can be reversed by applied electric field. FeRAM has a structure of a CMOS transistor connected in series with a capacitor with a ferroelectric film, in a 1T-1C configuration. The structure of FeRAM is shown in Fig. 1-4. It has three lines, gate line (GL), bit line (BL) and plate line (PL). The transistor switches help prevent disturbing states of unselected cells in crossbar architectures similar to that of DRAM [14]. FeRAM is a non-volatile memory with a spontaneous polarization, where the polarization has two equilibrium states called remanent polarization, of equal but opposite (up and down) directions, hence giving its “binary states”. The polarization direction can be reversed by the application of an electric field, and the polarization state is maintained after the electric field is removed. To read the memory state, polarization reversal current is detected for ON state and OFF state [15]. FeRAM is among the first emerging non-volatile memory technology to be commercialized [16]. It is a very promising technology, with lower power consumption and endurance exceeding that of flash memory, PCM and STTRAM [17]. It also has fast read/write speeds (about 100 ns) and good retention (about 10 years) [18].

**Figure 1- 4:** The schematic of a ferroelectric random access memory (FeRAM) in the 1T-1C configuration. The polarization direction in the ferroelectric layer can be reversed through the application of electric field, by applying voltages to the bit line and plate line to write ‘0’ and ‘1’ states, respectively [adapted from [16]].
However, it has some major drawbacks such as a destructive read process and not being CMOS compatible [18]. The former drawback means that the non-volatile memory loses its data when the device is read out. This requires data to be rewritten after the data is sensed from the memory, which lowers the actual endurance of FeRAM. The current technology for FeRAM are based on lead zirconate titanate (PZT) and strontium bismuth tantalate (SBT) materials, which suffer from unstable ferroelectric properties at thin film thicknesses [19]. The scalability of FeRAM memories based on PZT and SBT is limited to 130 nm node [19]. Therefore, ferroelectric tunnel junctions (FTJs) are proposed to overcome these challenges of destructive readout process, and hafnia-based FTJs are also promising due to their good CMOS compatibility [20]. FTJ devices have a thin ferroelectric layer of a few nanometers sandwiched between two metal electrodes. There are still challenges to be overcome in the fabrication of FTJ devices, such as achieving the ultra-thin thickness of the FTJ film and the reproducibility of their performance [16].

The ferroelectric field effect transistor (FeFET) is another type of memory where a ferroelectric material layer is incorporated into the gate of the field effect transistor (FET). The value or degree of polarization of the ferroelectric film in the gate of FeFET determines the threshold voltage of the FeFET, and is used to store information. When positive and negative voltage is applied to the gate of n-channel FeFET, the device is in ON-state and OFF-state, respectively [19]. The polarization pointing downwards attracts minority carriers into channel and increases conductivity of FeFET, whereas polarization pointed upwards repels minority carriers and decreases conductivity of FeFET. The information is sensed through the drain current of the FET [15]. FeFET has a non-destructive read and is scalable, however, it suffers from poor data retention properties after the gate substrate has been improved to mitigate the chemical reaction between ferroelectric film and Si substrate [16, 21]. For FeFET based on Bi₄Ti₃O₁₂, the gate consisting of
metal-ferroelectric-Si (MFS) results in a chemical reaction between the ferroelectric film and the Si during fabrication and degrades the FeFET [16]. The constituent elements of both diffuse into each other during crystallization annealing when ferroelectric film is deposited directly on Si substrate. In the ferroelectric-Si substrate interface, to prevent inter-diffusion of constituent elements in both Si substrate and ferroelectric film, a buffer insulating layer is introduced. [15]. Improvements have been sought by introducing metal-ferroelectric-insulator-Si substrates and metal-ferroelectric-metal-insulator-Si substrates, which resulted in poor data retention properties. This is due to depolarization field of the ferroelectric layer when the FET is switched off (this electric field is in the opposite direction of the polarization field), which affects data retention time [15].

1.1.4. Resistive Random Access Memory (ReRAM)

Resistive Random Access Memory (ReRAM), also known as memristors, were first theoretically hypothesized to exist by Leon Chua in 1971 as the fourth fundamental element, apart from resistors, capacitors and inductors [22]. The first working memristor was fabricated by Hewlett Packard (HP) labs in 2008 [23]. For ReRAM, there are two types of switching mechanisms, the electrochemical metallization mechanism (ECM) and the valence change mechanism (VCM). ReRAM based on VCM and ECM are discussed in the following paragraphs. The primary difference between VCM and ECM is the composition of their conductive filaments, where VCM has anionic oxygen vacancies while ECM has cationic metal ions in the filaments of the ReRAM, responsible for resistive switching.
1.1.4.1. *Electrochemical Metallization Memory (ECM)*

ReRAM based on the electrochemical metallization switching mechanism are also known as conductive bridge random access memory (CBRAM). In ECM, the dissolution of an active electrode (e.g. Cu or Ag) causes the migration of metal ions through a solid electrolyte [1, 24]. These metal ions undergo reduction at the other electrode which is inert (e.g. Pt or W), to form the conductive filament during the SET process, where a positive voltage is applied to top electrode. This results in the low resistance state of the device. In RESET, this filament is broken by Joule heating through the application of a negative bias to the top electrode [25] and results in the high resistance state. ECM-based ReRAM typically have much lower endurance of $10^6$ switching cycles, and lower retention time as opposed to VCM-based ReRAM (endurance up to $10^{12}$). The rest of this dissertation will discuss the VCM-based ReRAM.

1.1.4.2. *Valence Change Mechanism (VCM)*

Resistive Random Access Memory (ReRAM) primarily consists of a metal-insulator-metal (MIM) structure, where the resistive switching takes place via anion movement in the insulating oxide layer through application of electric field, forming a metal-rich filament (or filament of oxygen vacancies due to absence of oxygen ions) to conduct current between the top and bottom electrodes [26]. This is called the Valence Change Mechanism (VCM), as it alters the valence state of the metal and produces the oxygen vacancies as the main species responsible for switching. The switching material for VCM-based ReRAM has been investigated with perovskites and transition metal oxides such as titanium oxide (TiO$_x$) [23, 27], tantalum oxide (TaO$_x$) [28], hafnium oxide (HfO$_x$)[29], magnesium oxide (MgO) [30], nickel oxide (NiO) [31], zirconium oxide (ZrO$_2$), etc. [26, 32]. Likewise, various metals have been investigated for the top and bottom electrodes of ReRAM, including W [33], Pt [34], nitrides like titanium nitride (TiN) [29], tantalum nitride (TaN)
Among these transition metal oxides, HfO$_x$ has garnered great attention due to hafnium having a mature fabrication process and being CMOS compatible, and demonstrating low operating voltages for low power consumption. The morphology of the HfO$_x$ film, which can be either crystalline or amorphous, has an effect on switching performance. Capulong et al. has shown that amorphous hafnium-oxide based ReRAM has lower forming voltage than crystalline hafnium-oxide based ReRAM, with better switching endurance [32]. The switching performance of ReRAM can also be affected by the choice of top and bottom electrodes. For instance, the effect of W bottom electrode on HfO$_x$ ReRAM was investigated at various annealing temperatures for different time periods in [33], where they found that higher annealing temperatures oxidized the inert W BE, which caused lower forming and switching voltages, and resulted in device failure. In addition to this, Z. Yong et al. have shown that the bottom electrode of HfO$_2$-based ReRAM needs to be inert for better switching performance, where they demonstrated that PVD deposited TiN BE resulted in greater oxygen vacancy concentration near the HfO$_2$/TiN (BE) interface than ALD deposited TiN BE [29]. PVD deposited TiN BE had more grain boundaries and is less crystalline compared to ALD deposited TiN BE, so it scavenged oxygen from the HfO$_2$ switching layer and created more defects that widened the filament at the base of the ReRAM. This therefore resulted in reduced memory window (Roff/Ron) and lower forming voltages of ReRAM, due to wider conductive filament formation [29].

For VCM-based ReRAM, the devices are operated in bipolar mode, where voltages of opposite polarities are applied to the top electrode of the device to switch its resistance between the low resistance state (LRS) and the high resistance state (HRS). Before any switching event can take place, the pristine ReRAM device needs to undergo an electroforming process. In electroforming, a high positive voltage called the forming voltage, is applied to the top electrode
of the ReRAM, giving high electric field to induce a soft-breakdown of the dielectric in the MIM stack. During this process, the high electric field changes the dielectric properties to conduct current, due to the formation of oxygen vacancies and movement of ions in the metal oxide layer, which are then enhanced by Joule heating, to create the conductive filament through oxide layer [36-37]. This process is required only once and is irreversible. Once the ReRAM device has been formed, it can be switched repeatedly by partially rupturing and recovering the conductive filament through the RESET and SET processes, respectively. The schematic of the cross-section of VCM-based ReRAM device during the forming, the RESET and the SET processes, is shown in Figure 1-5. To RESET the ReRAM, a negative voltage is applied to the top electrode of the device for oxygen ions to recombine with oxygen vacancies, switching it to HRS state by partially dissolving the conductive filament. Similarly, a positive voltage is applied during SET process to push oxygen ions towards the top electrode, recovering the filament of oxygen vacancies and switching the device to LRS state.

ReRAM is a very promising candidate to replace Flash non-volatile memory due to its excellent CMOS compatibility, simple MIM structure, high switching speed (on the order of nanoseconds) and low energy consumption (on the order of 0.1pJ), making it suitable for both embedded data storage and low-power applications like IoT [38]. In addition to this, ReRAM also has good endurance (up to $10^9$ switching cycles) [39], exceeding that of Flash ($\sim 10^6$) and retention properties (>10 years). An endurance of $10^{12}$ has been demonstrated for ReRAM based on TaO$_x$/TiO$_2$ [39]. It is also possible to stack ReRAM in 3D integration due to its simple structure [39-40], forming high density data storage. However, the stochastic switching of ReRAM due to the nature of the formation and rupture of conductive filaments gives rise to intrinsic cycle-to-cycle and device-to-device variability observed in ReRAM. There has been on-going research for
the development of ReRAM since its first fabrication in 2008 to reduce the stochasticity and optimize ReRAM stack through investigation of different electrode and switching materials, interface engineering, programming algorithms and optimal control of ReRAM operating parameters (e.g. SET and RESET voltages, current compliance, etc.) [41]. Programming algorithms refer to the application of electrical pulses to modulate the resistance states of ReRAM, where the amplitude and pulse width of the voltage pulses may be varied. Optimization of device operating parameters through programming algorithms have also been employed in literature to reduce variability of ReRAM [42]. In our research lab, different materials including hafnium, tantalum and niobium have been investigated for the switching oxide layer, with various types of

Figure 1- 5: Schematic of resistive random access memory (ReRAM) based on the valence change mechanism (VCM). The conductive filament (CF) in the metal-oxide switching layer is first created during an initial electroforming process. The CF is comprised of oxygen vacancies as shown by the green circles, and conducts current between the top and bottom electrodes, giving the device a low resistance state. Once the CF has been formed, it can be partially dissolved and recovered through a RESET process and a SET process, respectively, to switch the ReRAM between the high resistance state and the low resistance state. During RESET, the oxygen ions shown in red circles recombine with oxygen vacancies to disrupt the CF.
electrodes including tungsten and titanium nitride, and different types of OEL layers such as hafnium and titanium [32-33,41-61]. The schematic of the hafnium-oxide based ReRAM focused in this work and discussed in later chapters is shown in Figure 1-6. It consists of titanium nitride (TiN) top and bottom electrodes, hafnium oxide (HfO2) switching layer and a titanium (Ti) OEL layer. The ReRAM material stack of TiN/Ti/HfO2/TiN is shown from top to bottom.

Figure 1-6: Schematic of cross-section of resistive random access memory (ReRAM) based on HfO2 ReRAM, that will be discussed in this work. The ReRAM top and bottom electrodes are titanium nitride (TiN), the switching oxide layer is hafnium dioxide (HfO2) and the oxygen exchange layer (OEL) is titanium (Ti).

1.2 Applications of Emerging Non-Volatile Memory Technologies

1.2.1. Storage Class Memory (SCM)

Emerging memory technology brings the advantages of fast access speed of volatile memory like SRAM and DRAM, and the high density data storage capability of the Flash memory, which makes them suitable to serve as storage class memory (SCM) [4]. The gap in access speed of SRAM and Flash memory varies by more than six orders of magnitude, and that of DRAM and Flash by three orders of magnitude [10]. SCM can fill the gap in speed (1 ns -100 ns versus $10^4$ ns – $10^7$ ns) and data storage density between the cache and the external hard disk drives, and result in performance improvement and lower cost [4, 38, 62]. However, for memory systems, the NVM
needs to have good endurance to replace DRAM as it is frequently accessed, while for storage systems, NVM requires good retention of data. Both memory and storage systems would benefit from high \( R_{\text{off}} / R_{\text{on}} \) ratio.

1.2.2. Hardware Security Primitives

Data encryption, random telegraph noise (RTN), the random fluctuations in current signals between 2 or more levels under constant voltage stress, owing to vacancy/ion movement on atomic level, is useful for entropy source for true random number generators (TRNG) [63]. In addition, the probabilistic switching of ReRAM and STTRAM through control of programming pulse amplitude and pulse duration, to yield a probability of 50%, has also been used for TRNG [64]. The variation in write speed of VCM based ReRAM has also been proposed for TRNG for IoT security applications [65].

The intrinsic variation present in ReRAM is also attractive for physical unclonable functions (PUFs). Conventional PUFs rely on the fixed process variation from IC fabrication, that is unique to each IC but uncontrollable [66]. The variation in ReRAM resistance, forming and switching voltages, device-to-device resistance, can be harnessed as a source of entropy for PUF applications, which may be possibly reconfigurable depending on operating conditions [4]. The PUF is a circuit that generates a n-bit response (output) to a n-bit challenge (input), and has its own unique challenge-response pair (CRP) such that it is unclonable. Therefore, ideally, the same PUF will always generate the same response for its unique challenge n-bit, but this response cannot be produced by other PUFs for the given challenge [4, 60]. This is measured using Hamming distance between different PUFs for uniqueness and within a PUF for reliability [67]. The reliability may be improved through re-programmability of NVM in innovative PUF designs. Typically, a set of CRP are produced by a PUF and stored in a database. For authentication, n-bit
challenges are applied and the responses are compared against the CRP stored in database. Strong PUF requires a large number of CRPs while weak PUF utilizes fewer CRPs with increased variability [68].

1.2.3 In-Memory Computing

Apart from high density memory storage, emerging non-volatile memory are attractive candidates for bridging the gap between memory and processing units, where their properties can be used to compute on the data where it is stored. This is different from the traditional von-Neumann architecture where memory and processing units are physically separated. This section gives a brief overview of different types of in-memory computing using ReRAM.

1.2.3.1 Vector-Matrix Multiplication (VMM)

One of the most computation-intensive tasks based on complementary metal-oxide semiconductor (CMOS) technology and the von-Neumann architecture is the vector-matrix multiplication (VMM). This is performed with graphical processing units (GPUs) to enable parallel computations, however, is very power-hungry [69]. One of the remarkable aspect of ReRAM is its ability to form crossbar array structures, which increases the density of memory. In crossbar structures, the row and column nanowires are shared by the top and bottom electrodes of ReRAM, respectively. The physical laws of Ohm’s law and Kirchhoff’s current law can be applied on the crossbar structure to perform the VMM operation in a single time step, thereby accelerating the mathematical operation of VMM. Therefore, the crossbar structures of memory arrays have been explored to speed up VMM, by co-locating memory and processor. VMM requires the multiplication of matrix with a vector. The matrix is programmed into the crossbar structure or 1T1R array using the multiple non-volatile memory states. The Ohm’s law and the Kirchhoff’s
current law are then used for the multiplication and summation respectively, irrespective of the matrix size [70-71]. VMM on a 1T1R array is demonstrated in Figure 1-7. The matrix is first written onto the resistance states of the crossbar or 1T1R array. The gate line (G8) is enabled to select the entire column. Then, the vector is translated into the amplitude/duration of a series of voltage pulses. The pulses are applied simultaneously to the array elements, and the resulting currents from Ohm’s law (in red dashed lines) is summed up along the column nanowire (in red solid line) using the Kirchhoff’s current law. The equation in Figure 1-7 shows the relation between the current along that column, $I_j$, and the applied voltages ($V_1 - V_8$) to the rows and conductance ($G$ – the reciprocal of resistance) of cells along that column. This can carry out the VMM in a parallel fashion in the analog domain at low power and is therefore extensively investigated as hardware accelerators [70-71].

This speed up of the VMM operations by NVM crossbar array structures is attractive for a variety of applications, including neuromorphic computing, discussed briefly in the sub-sections below.
In neuromorphic computing, the memristors serve as ideal electronic devices to mimic biological connections between neurons, known as synapses. The brain is able to efficiently process data in parallel at ultra-low power using $10^{14}$ synapses [72] and $\sim 10^{11}$ neurons on the timescale of a millisecond. As such, many neural networks have been developed using a combination of CMOS circuits and memristors to implement this to achieve classification tasks at lower power. As a synapse represents the connection between two neurons, the conductance (reciprocal of resistance), a.k.a. the weight of the synapse can be increased (potentiated) and decreased (depressed) depending on the timing difference between spiking events from pre-neurons and post-neurons, a learning rule known as Spike Timing Dependent Plasticity (STDP).
Hence, the conductance of the synapse represents the strength of the connections between two neurons [74].

Deep Neural Networks (DNNs) with backpropagation algorithms have proven to solve classification tasks with great success compared to Spiking Neural Networks (SNN) due to its processing of neuron outputs at synchronous time-steps [71]. One particular type of DNN, the Convolutional Neural Network (CNN), has had great impact on large-scale analysis and classification tasks such as speech and image recognition [75]. DNN relies heavily on VMM operations as a hardware accelerator for forward-inference and training stages of NN [70-71]. A multi-layer perceptron neural network for forward propagation, another example of DNN, is shown in Figure 1-8. It consists of an input layer, a hidden layer and an output layer of neurons. The neurons are represented by circles. Each of the connections between the neurons are synapses, which are represented by NVM devices in an array in this Figure. Figure 1-9 demonstrates how the VMM is applied in a neural network for forward propagation to one neuron in next layer. The synaptic weights of synapses are represented by conductance of NVM in an array. The five neuron activations from input layer are first converted to voltages $V_1 - V_5$, which are then applied to the rows of the array, and are multiplied by the weights (conductance) of NVM cells in a column. These are then simultaneously accumulated down the column to serve as weighted sum of inputs to a post-neuron in next layer for forward-propagation of neuron excitations, as shown in Figure 1-9(a) [73, 76]. This weighted sum of current is then input into a function that determines the excitation of post-neuron $Y$, as shown in Figure 1-9 (c). The portion of neural network where the VMM is taking place is depicted in Figure 1-9 (b). The neuron circuit may be implemented by mixed-analog digital designs or NVM devices [10, 71]. Therefore, dense arrays of resistive memory have been proposed to accelerate DNN algorithms by processing data in parallel inside
of memory in an analog manner [69, 71, 72, 74, 75] and result in high energy efficiency compared to that of traditional CPU and GPU hardware and advanced CMOS-ASIC accelerators [77].

**Figure 1-8:** A multi-layer perceptron neural network consisting of 3 layers, one input, one hidden and one output layer. The circles represent neurons, and each connection in-between them represents a NVM synapse. Arrays of NVM are used in conjunction with CMOS circuitry to implement neural networks. Forward propagation is shown here.
1.2.3.1.2. Arithmetic Operations based on VMM

VMM is also useful for accelerating other arithmetic applications such as linear transformation and signal processing; image processing, error-correcting codes [61], compressed sensing [78], sparse coding [79] and solution of linear systems of equations [80-81]. For instance, Li et al. performed a discrete cosine transformation (DCT) on ReRAM crossbar, which is commonly used in digital signal processing [82]. VMM on crossbars was also demonstrated for spectrum analyzer to convert input signal from time-based to its frequency components [83]. In addition, 2D convolution, similar to that of CNNs, was also used for image filtering [82]. In [61], message bits
were encoded using Hamming code (7,4) through VMM, where three parity bits were computed by the VMM of message bits with Hamming code generator matrix.

1.2.3.1.2. Closed-Loop Circuits

A modification to the crossbar is to integrate additional negative feedback circuits, between each row and individual columns, such that the applied input currents along rows can be feedback to the columns, making analog closed-loop crossbar array [70]. This enables solving systems of linear algebraic equations \( Ax=b \) and matrix eigenvectors without digital iterations (in just one step), when output of each row is connected to its respective column through the operational amplifiers [70, 84]. For instance, in the solution to \( Ax=b \), where \( A \) is array conductance and \( b \) is the vector of input currents, the analog closed-loop circuit outputs voltage along columns representing \( x=A^{-1}b \) [70] using Kirchhoff’s law. On the other hand, integrating negative feedback circuitry between each row and columns of multiple crossbars have been proposed to accelerate solutions of linear regression problems [85].

1.2.3.2. Search Operations / Content Addressable Memory

Ternary content addressable memory (TCAM) circuits are used for search of large datasets within specified ranges and hence enable complex routing [86]. Instead of conventional approach where data is read from the address it is stored in, TCAM searches the data using its content and not address. The TCAM functions by pre-charging a match line (ML). During the search operation, when the TCAM search word is compared with content of memory, the ML will discharge quickly if there is a mismatch. In the case of a match, the ML discharges slowly [86-87]. ReRAM-based TCAMs are attractive as opposed to SRAM-TCAMs made of transistors that consume large area on silicon and also lowers the search latency. The non-volatility of ReRAM would also reduce the standby power of SRAM-TCAMs. ReRAM-based CAM is therefore attractive for a wide range of
applications such as genomics and reconfigurable computing, and has shown excellent performance for pattern matching and approximate computing [87]. Another data matching method was proposed in [88] where a look-up table (LUT) for inputs and output of Boolean function are stored in crossbar array, where each row holds specific inputs and its output. Search data is applied in the form of voltages to the columns and current is sensed along each row and compared against a threshold for matched data [88]. However, the memory windows of programmed content and the magnitude of read voltage applied for search operations using ReRAM is a challenge [86].

1.2.3.3. Logic Operations

Logic operations have been proposed on NVM devices using either resistance of devices or voltage pulses applied to devices, or a combination of both, by few different methods as a potential in-memory computation. There are various methods reported in the literature for performing logic functions on Resistive Random Access Memory (ReRAM). As described above, ReRAM has been regarded as a very promising non-volatile memory, with the advantages of low power consumption, high scalability, good endurance (greater than $10^{10}$ cycles) and high switching speed. This makes it very attractive for logic functions.

For instance, a method called material implication (IMP) and FALSE, has been proposed to implement a variety of Boolean logic operations on ReRAM [89]. IMP involves the conditional switching of ReRAM, is based on two variables $p$ and $q$, where the logical states of $p$ and $q$ are stored in terms of the resistance states of two memristors. The two memristors share a common horizontal wire, which is connected to a load resistor. For material implication, IMP, the pulses $V_{\text{cond}}$ ($V_{\text{cond}}$) and $V_{\text{set}}$ are applied simultaneously to the memristors holding $p$ and $q$ respectively. The state of memristor $q$ will be switched conditionally based on the resistance states...
of the memristors p and q. For ‘q \leftarrow p\text{IMP}q’, the output of the IMP operation is stored in resistance state of memristor q, where HRS and LRS represent logic 0 and logic 1 outputs, respectively. The truth table of “p\text{IMP}q” is the same as (NOTp)ORq [89]. This method is also used with additional memristors, called functional memristors, and a sequence of steps, to implement various Boolean logic operations. Borghetti et al. also used 3 binary resistive switches and 7 sequential steps to realize NAND operation [89]. A full addition based on this method is proposed in [90]. A method did an improvement over the IMP method by demonstrating multiple logic operations on a single 1T1R device, but it involves a sequence of steps [91]. On the other hand, logic operations has also been proposed within crossbars such as the memristor-aided logic (MAGIC). This uses a configuration of loading inputs to states of input memristors that share a common electrode, and the output from common electrode is connected to an output memristor. A carefully designed voltage Vo is applied for MAGIC operation, to yield switching of output memristor resistance state based on states of input memristors, which yields the computation result [92]. Many different approaches for Boolean logic on ReRAM will be discussed in Chapter 5.

Non-volatile memory characteristics of NVM, such as the crystallization dynamics of phase-change memory that lead to accumulation behavior, have also been proposed for calculating complicated arithmetic operations such as parallel factorization and fractional division. Base-10 additions and base-6 subtractions have been carried out through the accumulation behavior of PCM cells. This approach was further extended for parallel factorization and fractional division [93].

1.3. Summary

Emerging non-volatile memory technology has gained attention due to its wide variety of performance metrics (such as speed, endurance, retention, etc.) and particularly scaling potential and low power consumption. The pros and cons of major types of emerging memory technology,
PCM, STTMRAM, FeRAM, FeFET and ReRAM, have been discussed. The performance of emerging NVM is attractive for storage class memory (SCM), which bridges the gap between high performance and high-density storage. ReRAM is advantageous in terms of its high scalability, good CMOS compatibility and low power consumption, as well as high switching speed, high density, multi-level resistances, good endurance, etc. The range of applications that could benefit from the properties of ReRAM as a viable candidate, such as in-memory computations and neuromorphic computing has also been presented. The next chapter highlights the importance of in-memory computations and outlines the research conducted in this work.

1.4. References


Chapter 2: Problem Statement and Thesis Organization

2.1. Problem Statement

For the past five decades, Moore’s law, the doubling of transistors in the same area every two years, has driven a boost in performance of digital electronics while maintaining the cost and area. Dennard’s scaling, where device design parameters were optimized to support Moore’s law, has already come to its end in 2004 [1]. The ITRS (International Technology Roadmap for Semiconductors), which tracked the historical progress of Moore’s law, has already disbanded due to reached atomic scales of few atoms across critical device features, upon which they projected no further improvement. In addition, the excessive heat generated by packaging large numbers of transistors in small areas for faster speed and lower power consumption increased the challenges faced in CMOS scaling. For instance, CMOS technology nodes below 20 nm incurred very costly fabrication masks, complicated manufacturing fabrication processes, high static power consumption, etc. [2]. Furthermore, due to the end of Dennard scaling, the power density remained the same with smaller transistors, giving rise to the so-called “Power Wall”, which has limited the maximum clock speed at 4 Gigahertz since 2004. To overcome this, manufacturers have tried to keep up with Moore’s law by limiting the maximum clock speed as mentioned, and instead increased the number of processor cores on a chip and designed dedicated accelerators to comply with expected increase in performance [3, 4]. Hence, it has become increasingly difficult to comply with Moore’s law.

Conventional computing systems are based on the von Neumann architecture, which was named for John von Neumann, who described this basic computing architecture in the 1940s. The von Neumann computing architecture is shown in Figure 2-1, for which improvements in
performance were driven by the now declining Moore’s law. In the standard von Neumann architecture of current computing systems, the memory and processor are at physically separate locations. This incurs significant energy dissipation and latency in data movement between memory and processor, leading to the von Neumann bottleneck, especially for data-intensive applications and power stringent systems such as mobile and server systems [5].

In the von Neumann architecture, the data and instructions are stored in main memory unit. Instructions are fetched from main memory to controller unit and decoded. Then, data is fetched from memory and loaded into cache or registers to be computed upon by the arithmetic logic unit (ALU). The computation results are then stored back into the memory unit. Instructions and data are transferred to processor unit using the same bus, which gives limited bandwidth. Although transistors have improved their energy efficiency by scaling, wires and interconnects did not gain such high energy efficiency due to end of Dennard scaling [1]. Hence, the cost of data movement across a distance of greater than 5mm is on the order of 4,000 pJ on the system level at the 11 nm node, which is much greater than the floating point operation performed on the operands (~10 pJ) [1].

The extensive amount of data that is being processed today has aggravated the von Neumann bottleneck. In addition, the execution time for communication and memory access is much greater than that of computation in the von Neumann architecture [6], which is significantly worse for data-intensive applications. This leads to the processor remaining idle while waiting for data transfer, which hinders reaching the maximum performance [7,8,9,10,11]. Moreover, the difference in speed gap between the memory and processor requires separate embedded L1 cache assigned to each core, which increases the area and leakage power overhead, leading to greater power consumption and limited system scalability [6]. Hence, the von Neumann architecture
suffers from limited scalability, limited bandwidth, high latency and energy inefficiency. Therefore, alternative non-von Neumann computing approaches are needed to improve the energy efficiency and overcome this bottleneck.

Figure 2-1: The von-Neumann architecture, consisting of separate processing unit and main memory unit. The processing unit contains comprises of both control unit and arithmetic logic unit (ALU). Instructions and data are stored in memory, which are fetched using the bus. For processing, instructions are fetched and decoded. Then, data is loaded from memory into cache or registers, and computed upon by the ALU. The result of computation is then stored back to the memory. Data needs to be transferred back and forth in this architecture.

One approach looks at computing inside the memory, called in-memory computing or processing in-memory (PIM), which can lower the power consumption due to data transfer between separated memory and processor units in current chip architecture, thereby finding a way to boost chip performance [5, 12]. This approach focuses on processing the data within or near to the memory, to avoid the need of data transfer [5, 12]. As such, novel approaches to high density,
non-volatile memory are needed to support unique in-memory computing approaches. ReRAM is a promising non-volatile memory that has low power consumption, high switching speed, simple structure, high density, good scalability and good compatibility with CMOS [13]. These benefits render ReRAM emerging memory technology as a potential candidate for non-von Neumann computing applications, to overcome the von-Neumann bottleneck. Many such applications have been presented in Section 1.2.3. of Chapter 1, and these efforts of computing on ReRAM memory arrays are beneficial for low power computing, artificial intelligence related tasks and the semiconductor industry. As such, this work aims to investigate two types of ReRAM applications, and the effect of ReRAM device non-idealities on these non-von Neumann computing applications.

2.2. Thesis Organization

The background of memory technology and possible applications of ReRAM is presented in Chapter 1. Chapter 2 (this chapter) presents the problem statement and the scope of this research work. Subsequently, Chapter 3 presents the HfO$_2$-based 1-transistor-1-ReRAM (1T1R) array device fabrication, operation, and the electrical characterization of multi-level resistance states attained by changing the current compliance via gate voltage modulation of integrated transistor. These multi-level resistance states will be assessed for repeatability across the 300mm wafer, for 1T1R devices within an array (intra-array), between arrays on a die (inter-array) and die-to-die resistance variation over an endurance of 10,000 switching cycles. The resulting data aids subsequent implementation of non-von Neumann computing and will inform the simulation of flow-based computing on Chapter 5.
Chapter 4 investigates the feasibility of temporal correlation detection using HfO$_2$ ReRAM arrays as a potential candidate through simulation and experiments. This uses the ReRAM as a computational memory, where the degree of association between discrete-time binary processes will be detected by the change in resistance/conductance of ReRAM memory, in an unsupervised manner. The temporal correlation detection algorithm will be simulated to detect correlated discrete-time binary processes in Python environment with an empirical ReRAM model of fabricated devices to check the feasibility of the implementation of this algorithm with ReRAM non-volatile memory. The algorithm utilizes pulse amplitude modulation of SET voltage for the empirical ReRAM model. Next, due to limitations of available testing equipment, the effect of variability of HfO$_2$ ReRAM on the correlation detection algorithm/this algorithm, will be investigated through simulation using extracted resistance from experimental ReRAM analog data. In these experiments, incremental resistance changes is achieved in a monotonic manner using pulses in the sub-nanosecond regime. Incremental resistance changes, both in the SET and RESET regimes, will be investigated for the temporal correlation detection algorithm using experimental analog data. The algorithm for the experimental data is adjusted to rely on the number of programming pulses rather than pulse amplitude modulation, due to fixed pulse amplitude and pulse width of data to be used. The experiments consist of incremental resistance changes using sub-nanosecond pulses, in either SET or RESET regimes, to give the non-volatile accumulative behavior of device that is required for this type of non-von Neumann application. This application can benefit from the ultra-low power consumption of ReRAM.

In Chapter 5, I aim to demonstrate the approximate edge detection between two 8-bit pixels on 8x8 1T1R hafnium-oxide arrays. Furthermore, the impact of R$_{off}$/R$_{on}$ resistance ratios and their respective ReRAM resistance state variability will be evaluated on the implementation of flow-
based edge detection computing, through both endurance measurements / experiments and simulation. The work on flow-based computing will be extended to include the investigation of flow-based XOR operation on the 1T1R arrays.

The final chapter, Chapter 6, provides a summary and conclusions of my PhD dissertation work and gives some perspective on the future directions of this work.

2.3. References


3.1. Introduction to Multi-Level Resistance States

ReRAM devices are known to exhibit greater variability in the HRS states than the LRS states. This is due to the various conduction mechanisms that occur during the partial disruption of the conductive filament in the RESET operation [1]. Multiple conduction mechanisms have been reported in the literature including space charge limited current (SCLC) [1], Frenkel-Poole [2], Schottky [3], Trap Assisted Tunneling (TAT) [4-5], etc., which have been suggested for the conduction in metal-oxide ReRAM and hence the variability observed. However, the research on the conduction mechanism responsible for ReRAM variability is inconclusive and still remains open. It is also thought that there may be more than one conduction mechanism at play during conduction in the HRS state [1]. Meanwhile, the LRS states are dominated by Ohmic conduction, where it is dictated by the size of the conductive filament [6].

The programming of binary resistance states and multi-level resistance states is required for non-von Neumann computing applications. To achieve reasonable accuracy, Hu et al. has used the multiple resistance states in the LRS regime for carrying out vector matrix multiplication (VMM) on 1T1R arrays for MNIST data classification and signal processing application [7]. In addition, neuromorphic applications also require accurate programming of NVM arrays to hold the trained weights for inference tasks [8], where each ReRAM serves as the synaptic device. Merced-Grafals et al. reported that programming of resistance states around 12 percent of the target resistance
states on tantalum oxide 1T1R cells may be sufficient for neural network training applications [9]. If passive arrays are used (1R arrays), where device nonlinearity may serve to isolate the devices during programming instead of selector devices, the nonlinearity reduces the programing accuracy of analog resistance levels. Hence, a transistor used as selector device may be a good option [9].

MLC refers to the ability to program a memory device (cell) into more than two resistance levels. Multi-level cells (MLC) have been implemented with ReRAM using programming voltage amplitudes and pulse widths, as well as current control [10-12]. The latter approach refers to the use of a current compliance during the device switching. Hence, the accurate programming of ReRAM devices is also vital for non-von Neumann computing, and serves as the initial step before moving forward to applications of memory solutions.

3.2. Fabrication of ReRAM Devices

At SUNY Polytechnic Institute, many materials have been investigated in our lab for ReRAM switching, including hafnium oxide and tantalum oxide, along with titanium nitride, tungsten and other electrodes, and different oxygen exchange layers [3,4, 13-34]. The ReRAM discussed in this dissertation is based on hafnium-oxide switching material, which was custom-developed in [13,26-27] and fabricated in the 300mm fabrication facility at SUNY Polytechnic Institute.

At SUNY Polytechnic Institute, we have fabricated 100 x 100 nm² bipolar hafnium oxide ReRAM, integrated with CMOS at the metal 1/ via 1 interface using a 65nm CMOS process technology on the 300mm wafer platform, in a 1-transistor-1-ReRAM (1T1R) configuration [13-15,17-22,27-34]. The ReRAM consists of a TiN/Ti/HfO2/TiN material stack (top to bottom), where 6.3 nm hafnium oxide is the switching layer (SL), 6 nm Ti is the oxygen exchange layer (OEL), 40 nm and 70 nm of TiN serves as the top electrode (TE) and bottom electrode (BE),
respectively. The transmission electron micrograph (TEM) image of the fabricated ReRAM at the metal 1/ via 1 interface is shown

![Transmission Electron Micrograph](image)

**Figure 3-1**: A transmission electron micrograph (TEM) of the fabricated 1-transistor-1-ReRAM (1T1R) device, and the cross-section of the ReRAM device showing the device stack composition, the TiN top electrode, the Ti oxygen exchange layer (OEL), the HfO₂ switching layer and the TiN bottom electrode. © 2021 IEEE, Reprinted, with permission.

in Figure 3-1 [18]. The ReRAM devices are integrated onto the drain of the transistor, to form the 1-transistor-1-ReRAM (1T1R) configuration.

The fabrication of the custom-built ReRAM integrated with CMOS at the 300mm fabrication facility at SUNY Polytechnic Institute is as follows. The bottom electrode, TiN, is deposited by physical vapor deposition (PVD) above patterned M1 using a subtractive integration process and is defined as 100 nm x 100 nm by a reactive ion etch (RIE) process [27]. After the BE is defined, a via insulating layer of Si₃N₄, is deposited and patterned on the BE layer. The hafnium oxide layer is then deposited by atomic layer deposition (ALD), to obtain a uniform layer at 300 degree Celsius. This is followed by the deposition of 6 nm Ti and the TiN TE through the physical vapor deposition process (PVD). The stack is then lithographically patterned by a custom-
developed reactive ion etch (RIE) process, such that there is an extension of 50nm of the subsequent layers (SL/OEL/TE) around the BE, to prevent edge effects. To protect the ReRAM stack from the subsequent V1/M2 patterning steps, a passivation Si₃N₄ mask layer, was deposited on top of TE. A thick V1 was deposited and patterned, followed by a dual-damascene Cu V1/M2 module to complete the ReRAM integration between M1/V1 interface.

Furthermore, various test structures of ReRAM, including individual ReRAM (1R) structures, 1-transistor-1-ReRAM (1T1R) structures, as well as array structures of both 1R and 1T1R, have been developed on the 300mm wafer platform. The devices are operated in the 1-transistor-1-ReRAM (1T1R) configuration. The integrated transistor provides the current compliance of the device during switching between the binary states. Current compliance is the maximum current that is allowed to flow through the device during the onset of FORM and SET operation, to prevent hard breakdown of the device [35]. This chapter will discuss the electrical characterization of 8x8 array structures of the 1T1R hafnium oxide ReRAM devices, as the crossbar arrays are required for the subsequent non-von Neumann computing applications discussed in later chapter. The schematic of the 8x8 1T1R array is shown in Figure 3-2 [18]. The arrays have demonstrated excellent yield of greater than 90 percent [18].

3.3. Electrical Characterization and Operation of ReRAM Devices

The device switches using the valence change mechanism (VCM), where a filament of oxygen vacancies forms in a soft breakdown when a high voltage pulse of 3V - 3.5V is applied to the top electrode of the ReRAM. The ReRAM devices undergo forming at around 2.5V. In this work, the ReRAM devices are integrated with on-chip NFET transistors based on the 65nm CMOS technology to operate the device in the 1-transistor-1-ReRAM (1T1R) configuration. The transistor is connected in series with the ReRAM device to limit the maximum current that flows
during FORM and SET operations, to prevent hard breakdown of the dielectric [5, 35]. The integration of the transistor on-chip is preferred compared to having an external transistor to eliminate parasitic effects at the transistor-ReRAM junction [2, 36]. It therefore helps mitigate the overshoot current due to charging effects from parasitic capacitance [36], and also results in lowering resistance variability significantly during SET operations as opposed to having no selectors [9] and arrays with non-linear memristors [37].

Electrical characterization of the ReRAM devices were performed by a semi-automated SUSS MicroTech probe station, an Agilent B1500 device parametric analyzer equipped with 4 Source Measurement Units (SMUs), and a B1530 module installed onto the B1500 for high frequency measurements. The B1530 module, also known as the Waveform Generator Fast Measurement Unit (WGFMU), was used in conjunction with the B1500, to provide the tens of microseconds switching pulses for low-power ReRAM operation, and improved performance [38]. The minimum time resolution of the WGFMU is 10 ns. This provided switching pulses in the range of tens of milliseconds pulses for FORMING and tens of microseconds pulses for the SET/RESET/READ operations of ReRAM devices. The test measurement programs were developed in-house on the Python environment and a Python interactive GUI was used to control the experiments.

A typical pulse train applied for ReRAM operation is shown in Figure 3-2. A pulse-based FORM, RESET and SET programming was adopted as it results in better switching characteristics than DC sweep [38]. For a pristine ReRAM device, the device needs to undergo a soft breakdown, known as the FORMING operation, to form the conductive filaments that can conduct current in the switching oxide [5, 35]. This is achieved by applying the FORM triangular pulse, a positive voltage that is higher than the subsequent switching pulses. After the device is formed, the
switching pulses, positive and negative triangular voltage pulses are applied for SET and RESET operations, respectively. During SET and RESET of ReRAM, the conductive filaments are recovered and partially dissolved to switch the device to low resistance state (LRS) and high resistance state (HRS), respectively [6]. After each programming pulse, a read pulse of -0.2V with 10µs pulse duration is applied to read the resistance of the device, such that the voltage is low enough not to disturb the programmed resistance. The switching cycle, consisting of RESET, READ, SET, READ pulses, is highlighted in green box in Figure 3-2. Thus, switching cycles are applied repeatedly to program ReRAM devices into the LRS and HRS binary states. All the voltage pulses are applied to the top electrode of ReRAM, as shown in Figure 3-3. The maximum current determined by the saturation current of integrated transistor is known as the current compliance [36]. The ReRAM is integrated on drain side of the transistor. The source of the transistor is grounded during the measurements. A gate voltage is applied to control the saturation current of

**Figure 3-2:** A typical pulse train of FORMING, RESET, SET, READ voltage pulses applied for bipolar hafnium-oxide ReRAM operation. The FORMING pulse is applied once to pristine device, and the switching cycle consisting of RESET, READ, SET, READ are applied repeatedly for switching the ReRAM between the low resistance state (LRS) and the high resistance state (HRS). The READ pulse is applied to read the programmed resistance after FORM, SET and RESET operations.
the transistor, that provides the current compliance during the pulsed measurements. The typical drain current vs drain voltage, \( I_d-V_d \) characteristics of the integrated NFET is shown for various gate voltages in Figure 3-3 (b), the saturation current increases from 40\( \mu \)A to 380\( \mu \)A with the increase in applied gate voltage.

**Figure 3-3:** a) Voltage pulses for FORM, RESET, SET and READ operations are applied to the top electrode of ReRAM. The ReRAM is connected in series with the drain of the transistor, the source of the transistor is grounded and the gate voltage is applied to control the current compliance b) drain current vs. drain voltage, \( I_d-V_d \) characteristics of integrated NFET transistor for drain voltage sweeps at various gate voltages, yielding a range of saturation currents that can provide the current compliance needed during the 1T1R switching.
3.4. Operation of ReRAM Devices

The bipolar hafnium-oxide ReRAM switches based on the valence change mechanism (VCM), as mentioned previously. The switching material is based on HfO₂ as the dielectric. Initially, an electric field is applied using a high enough positive voltage to break Hf-O bonds due to strong bond polarizability of high-k HfO₂. The generation rate of positively charged oxygen vacancies is facilitated by elevated temperature along grain boundaries due to electron transport through trap-assisted tunneling (TAT) [5]. Thus, oxygen vacancy generation rate increases exponentially due to positive feedback between electric field and enhanced local temperature, giving rise to runaway forming process [5]. This causes an abrupt breakdown of dielectric and creates stable hafnia-rich metallic conductive filaments to conduct current in the dielectric. The schematic of the proposed cross-section of the ReRAM stack during FORMING, RESET and SET, are shown in Figure 3-4. In a study, it is proposed that the interplay between electric field and elevated local temperature out-diffuse oxygen ions towards the top electrode (TE) [5]. An oxygen exchange layer (OEL), sandwiched between the TE and dielectric layer (not shown in the figure), getters oxygen ions from dielectric during forming process to help out-diffusion of oxygen ions and create oxygen vacancies [39]. In addition, the deposition of this OEL layer forms an oxide by itself, which helps to make the hafnium oxide layer slightly defective before the FORMING process. This therefore ensures that the FORMING voltage of the MIM stack is low enough to create a filament that can be partially dissolved during switching. If the FORMING voltage is too high, the created filament may be too large that it cannot be reset. In our case, 6nm Ti was used as the OEL layer in our device stack.

Once the device is formed, it can now be switched. A RESET operation is performed by applying a negative voltage to the TE of the bipolar ReRAM device. The oxygen ions driven by
the electric field, undergo recombination with oxygen vacancies at the CF tip near the BE, as proposed in [5]. This re-oxidizes the bottom CF tip and creates a dielectric barrier, reducing the current flow in the ReRAM and transitioning to the high resistance state (HRS) [40-41]. The schematic of the proposed cross-section of ReRAM after RESET is shown in Figure 3-4. During SET operation, the dielectric barrier undergoes breakdown through application of smaller positive voltage than forming voltage to the

![Figure 3-4](image)

**Figure 3-4**: The valence change mechanism (VCM) of bipolar metal-oxide ReRAM. During FORMING, a conductive filament of oxygen vacancies is formed in the metal-oxide layer. During RESET operation, negative voltage is applied to top electrode to push oxygen ions and undergo recombination to partially dissolve the conductive filaments. During SET operation, positive voltage re-connects the conductive filaments to conduct current.

TE, known as SET voltage. The SET voltage is smaller as the voltage mostly drops along the dielectric barrier and re-generates the oxygen vacancies at this portion to reconnect the conductive filament. This results in abrupt increase in current and transitions to the low resistance state (LRS). The schematic of the proposed cross-section of ReRAM after SET is shown in Figure 3-4. The
ReRAM device can be repeatedly SET and RESET through application of SET and RESET voltages to the ReRAM device, respectively.

The pulsed current-voltage (I-V) characteristics of ReRAM during FORMING, RESET and SET are shown in Figure 3-5. A FORMING voltage pulse of 3V-3.5V is applied to pristine hafnium-oxide ReRAM to undergo forming process, where the ReRAM formed at a forming voltage of 2.7 V. After device is formed, the ReRAM device is RESET and SET for ten cycles in Figure 3-5, with applied RESET and SET pulses of -1.5V and 2.5V, respectively. The pulse width of forming pulse and the switching pulses were kept in the range of tens of milliseconds and tens of microseconds, respectively. The current compliance during SET and FORM is controlled by the integrated transistor, as mentioned previously. The first RESET current is higher than the subsequent RESET currents. This is due to greater energy needed to dissolve the greater CF filament formed with higher positive voltage during FORMING step. The subsequent positive switching voltages for SET are lower as it only re-connects the CF at dielectric gap. Thus, the subsequent RESET currents are due to re-creating the dielectric gap that was dissolved by smaller positive SET voltage. The average LRS and HRS of the ten cycles of pulse I-V characteristics shown is 7.8 kΩ and 100 kΩ, respectively. However, higher HRS values are possible based on applied pulse conditions, with higher RESET voltages as in [14-15].
Figure 3-5: The pulsed I-V measurements of ReRAM FORMING, RESET and SET. The ReRAM forms at a voltage of 2.3V, and undergoes RESET and SET operations at 0.75V and -0.85V, respectively. The applied FORMING, RESET and SET voltages were 3V, -1.5V and 2.5V respectively.

Due to uniformity of conductive filaments after SET operations, leading to Ohmic current conduction at Hf-rich clusters in the CF and reduced resistance variability, the multi-level resistance states in the LRS regime were studied. The multiple resistance states in the LRS regime are required later for non-von Neumann computing applications, which are discussed in Chapter 5.

3.5. Multi-Level Resistance States with Current Control

In this study, multiple resistance states were obtained in the LRS regime by varying the current compliance through gate voltage modulation of integrated transistors on array structures of 1T1R. A schematic of the 8x8 1T1R array is shown in Figure 3-6. Each 1T1R array element was accessed through its drain, source and gate lines [18]. The array elements were switched with fixed SET and RESET voltages at six current compliances, ranging from 50 µA to 390 µA, and the resulting LRS resistance states were assessed for intra-array, inter-array and die-to-die resistance distribution. The typical I-V characteristics of an array element for each of the switching conditions
with respect to current compliances is shown in Figure 3-7. The pulsed I-V measurements in Figure 3-7 shows ten switching cycles of SET and RESET of the 1T1R at each current compliance, and demonstrates uniform switching. The SET and RESET voltages are 2.5V and -1.5V respectively. With increasing current compliance (I\text{\text{\textnormal{comp}}}) enabled by integrated NFET transistor during SET operations, the maximum SET current increases along with an observed increase in maximum RESET current. In [6], it is proposed that with greater I\text{\text{\textnormal{comp}}}, there is an increase in the oxygen vacancy concentration and the width of the conductive filament widens laterally in the HfO\text{\textsubscript{2}} switching layer [6], including the more resistive portion of CF that re-oxidizes to create the dielectric barrier during RESET. Therefore, the resulting LRS states decreases with increasing current compliance. This observation is consistent with our measurement results.

**Figure 3- 6:** The schematic of the 8x8 1-transistor-1-ReRAM (1T1R) array structures. Each 1T1R array element can be accessed through its drain, source and gate lines. The 1T1R FORM, SET, RESET, READ operations are the same as mentioned previously. © 2021 IEEE, Reprinted, with permission.
Figure 3- 7: The pulsed I-V measurements of ten ReRAM RESET and SET cycles at six different current compliances, ranging from 50µA to 400 µA. The current compliance regulates the maximum SET current for each switching cycle. The pulse IV switching cycles are uniform for each set of conditions. The applied RESET and SET voltages were -1.5V and 2.5V respectively. Higher current compliance therefore results in lower LRS resistance states.

The schematic of the cross-section of the bipolar HfO$_2$ ReRAM in Figure 3-8 represents the possible proposed evolution of the conductive filament during SET operations with increasing $I_{\text{comp}}$. There is a RESET operation in-between each of the SET operations of the ReRAM in Figure 3-8, which is not shown here. The conductive filament is proposed to expand laterally within the switching oxide layer, thereby increasing current flow during SET operations. The effect of $I_{\text{comp}}$ on the increase in CF radius can be explained by the following set of equations, discussed in [5,40,42]. As $I_{\text{comp}}$ increases, the current flow increases temperature. The effect of Joule heating is
shown in equation (1), where the temperature of filament, \( T \), increases in proportion to the square of increased current flow \( I \), the current through the CF. The rest of the parameters are the room temperature \( T_1 \), and the effective thermal resistance of the conductive filament \( R_{th} \). The rate equation of defect migration in the CF is shown in equation (2), where, \( V \) is applied voltage, \( \alpha \) is a barrier lowering constant, \( q \) is elementary charge, \( k_B \) is Boltzmann constant, \( T \) is temperature and \( E_{a0} \) is the activation energy of defect migration at no applied electric field. The applied SET voltage \( V \) lowers the activation energy by \( \alpha q V [43] \). In the experiment, the current compliance was increased while the applied SET voltage was kept constant. Therefore, the rate of defect migration increased with temperature \( T \), and increased the concentration of defects in the CF, thereby expanding the radius of the CF. The expansion of the radius of CF \( r_{CF} \) therefore lowers the resistance of the CF according to equation (3), where \( \rho_{CF} \) is the resistivity of the CF and \( t_{ox} \) is

![Figure 3-8](image-url)

**Figure 3-8:** The evolution of the conductive filament consisting of oxygen vacancies, with respect to increasing current compliance during SET operations, shown from left to right. The conductive filament widens laterally with respect to an increase in current compliance, and results in further reduction of the low resistance state \( R_{L,RS} \). The \( R_{L,RS} \) can be controlled via modulation of current compliance.
the thickness of the oxide switching layer, with increase in $I_{\text{comp}}$. This gives the resulting programmed LRS resistance states with respect to each current compliance. With increasing current compliance, the resistance of the LRS drops due to rising electrical conductivity with respect to concentration of oxygen vacancies [12, 44].

\[
T = T_1 + R_{TH} I^2 R
\]

(1)

\[
\frac{d(2r_{CF})}{dt} = Ae^{\frac{E_{\text{go}} - \eta V}{k_BT}}
\]

(2)

\[
R_{CF} = \frac{\rho_{CF\text{ox}}}{\pi(r_{CF})^2}
\]

(3)

Next, to obtain statistical analysis of the resistance distribution of the multi-level cell using the LRS regime, multiple 8x8 1T1R arrays were characterized with an endurance of 10,000 switching cycles with respect to the switching conditions discussed above, to yield 8x8 1T1R intra-array, inter-array and die-to-die resistance distributions. Multiple arrays from multiple die have been evaluated to determine the variability within arrays and die-to-die. Intra-array refers to 1T1R devices on the same array and inter-array refers to 1T1R devices on different arrays. The cumulative distribution of intra-array resistance distribution from 10,000 switching cycles of each device, of the multiple low resistance states programmed at the six current compliances, from 60 $\mu$A to 400 $\mu$A is shown in Figure 3-9. The programmed LRS resistance decreases with increasing current compliance as explained previously, and the distribution is wider with increased programmed resistance. Based on previous studies in the literature, the increase in the resistance variation at lower current compliance are proposed to be attributed to random defect migration in the CF growth, as a result of the random fluctuations in defect migration barrier $E_A$ [42]. In [42], it is proposed that the lower defect concentration at lower current compliances leads to wider variation of the mean and standard deviation of the defect migration barrier $E_A$, which increases
the cycle-to-cycle resistance variation of low resistive states programmed at lower $I_{\text{comp}}$ [42]. The study also suggested that at higher current compliances, the defect concentration is high enough such that the mean and standard deviation of the defect migration barrier $E_A$ has much narrower spread and results in less cycle-to-cycle resistance variation of programmed LRS states [42]. Despite this, there is still a distinction between the six low resistance states.

Then, two 8x8 1T1R arrays (array 1 and array 2) on the same die were characterized with the same test conditions and their respective cumulative distribution of the multi-level LRS resistance states are shown in Figure 3-10. The solid line and dashed line denote device resistance states for array 1 and array 2 respectively, at each of the switching conditions. The same trend in decreasing resistance and resistance variation for intra-array resistance states also holds for the inter-array resistance states. As can be seen, the multiple LRS states on both arrays can still be distinguished from one another.
Lastly, a die-to-die resistance variation of the multiple LRS states of 8x8 1T1R array devices with respect to current compliance across a 300mm wafer, is shown in box-and-whisker plot in Figure 3-11. The programmed LRS states decreases with increasing current compliance, along with a decrease in resistance variation. Figures 3-9, 10, 11, show that there has been a five-fold change in resistance, from an average resistance of 15 kΩ to 2.8 kΩ respectively, with increasing current compliance from 60 µA to 400 µA. Similarly, the resistance variation has also decreased from 2 kΩ to less than 400 Ω with increasing current compliance due to possible reasons discussed above. The resistance plateau observed for LRS states programmed with I_{comp} greater than 250 µA may be due to the finite size of the conductive filament [36]. The mean resistance for each box plot for 60 µA current compliance may be higher than the median due to the measurement resolution of test equipment. Secondly, according to [42], there is greater variation in each cycle for low I_{comp} due to greater difference between the spread of defect migration barrier E_A in CF growth for cycle-to-cycle [42], so this may have contributed to the mean resistance states programmed at 60 µA I_{comp} to be higher than its corresponding median resistance for the tested dice.

The resistance variations in the LRS originate mainly from the ReRAM and have minimal contribution from the integrated NFET due to following reasons. Firstly, the resistance of the NFET decreases with increase in gate voltage, due to higher current flow through the transistor. For instance, at 60 µA current compliance, the average 1T1R resistance was 15 kΩ while that of transistor alone on average is 2.4 kΩ. Additionally, the resistance standard deviation of the transistor providing the 60 µA current compliance at the applied read voltage is 275 Ω, as opposed to the observed standard deviation of 2 kΩ of the programmed 1T1R LRS state at this current compliance. Therefore, the transistor resistance variation has minimal contribution to the variation
Figure 3-10: Cumulative frequency distributions of inter-array LRS resistance values for 8x8 1T1R arrays across a die with respect to current compliance. The current compliance was controlled by varying the gate voltage of the integrated transistor.

Figure 3-11: Box-and-whisker plot for the die-to-die resistance variation in 8x8 1T1R array devices across a 300mm wafer, with respect to current compliance (by varying gate voltage of the control transistor). The standard deviation of the LRS resistance decreased with increasing current compliance.
in the multiple LRS states, so the resistance variation obtained are mainly from the ReRAM device [18].

The multi-level resistance states in the LRS regime have been used to program binary patterns into the 8x8 array structures using current compliance through gate voltage modulation. As such, examples of patterns programmed with a multitude of LRS states enabled by current compliance modulation is shown in Figures 3-12 (a) and 3-12 (b). The pixel intensity in each of these figures represent the programmed resistance state. In Figure 3-12 (a), binary LRS states programmed with current compliances of 60 µA and 400 µA yielded programming “NSF” logo onto the 8x8 array. In Figure 3-12 (b), a range of current compliances between 175 µA to 400 µA was used to program a greyscale image into the 8x8 array. Therefore, this approach was used to program 8x8 array structures for further use in non-von Neumann computing.

![Figure 3-12](image.png)

**Figure 3-12**: a) NSF logo programmed with binary LRS resistance values, using current compliance of 60 µA and 400 µA. d) A greyscale image was programmed into 8x8 1T1R array using current compliance from 175 µA to 400 µA. Each LRS resistance value corresponds to intensity of pixel in image.
3.6. Conclusion

At SUNY Polytechnic Institute CNSE, structures of hafnium-oxide based ReRAM has been integrated with 65nm CMOS technology on 300 mm wafer platform. The hafnium-oxide ReRAM consists of TiN/Ti/HfO$_2$/TiN stack, where the TiN forms the top and bottom electrodes, Ti is the oxygen exchange layer (OEL) and HfO$_2$ is the switching oxide layer. Array structures of 1-transistor-1-ReRAM (1T1R) devices have been evaluated for multi-level cell (MLC) for further use in non-von Neumann computing applications. As such, 1T1R device operation has been discussed, and multi-level resistance states in the LRS regime was obtained by varying current compliance of integrated transistor through gate voltage modulation. These multi-level resistance states were assessed for repeatability across the 300mm wafer, for 1T1R devices within an array (intra-array), between arrays on a die (inter-array) and die-to-die resistance variation over an endurance of 10,000 switching cycles. The results show that the multiple states are uniform, with decreasing average resistance and standard deviation with an increase in current compliance from 60 µA to 400 µA. With increasing current compliance from 60 µA to 400 µA, there is a five-fold change in resistance from 15 kΩ to 2.8 kΩ and the standard deviation decreases from 2 kΩ to less than 400 Ω. Lastly, this approach has been used to program binary and greyscale images with multiple LRS states into 8x8 arrays, which paves the way for further implementation of non-von Neumann computing applications.

3.7. Acknowledgment

The authors acknowledge the CNSE fabrication facility, process integration engineer, Dr. Karsten Beckmann, for the fabrication of hafnium-oxide ReRAM.
3.8. References


Chapter 4: Temporal Correlation Detection in ReRAM Arrays

Some of the data discussed in this chapter were previously published in 2020 IEEE Student Conference on Research and Development (SCoReD), written by the author Sarah Rafiq:


4.1. Introduction

Correlation shows the degree of association between variables [1]. Binary stochastic processes, may be represented by discrete random variables based on binomial distribution, occurring over time intervals denoted by k. To obtain the similarity between independent binary processes that occur simultaneously over time, where there is a correlation coefficient between each pair of binary processes (described in more detail in Section 4.2.1), is referred to as temporal correlation detection in this work. Correlation detection may be applicable for IoT, social networks, large scientific data, where data may be processed into binary data streams. The correlation between these binary data streams may be carried out in real-time to identify clusters of correlated data in incoming data streams [2].

The data transfer of operands between memory and processing unit in the von-Neumann architecture incurs more energy than the computation itself [3]. With the decline of Moore’s law and beginning the era of data intensive applications, non-von Neumann computing becomes critical [3-4]. The physical attributes and the state dynamics of non-volatile memory arrays may be used for non-von Neumann computing. This is accomplished by co-locating memory and processing units and is expected to decrease the latency and energy consumption in the von-
Neumann bottleneck [2-4]. Researchers have reported that the task of correlation detection have been accelerated by 200 fold using phase change memory (PCM) devices as opposed to performing the same task across four graphical processing units (GPUs), and decreased the power consumption by two orders of magnitude [2].

Phase change memory, the most mature among the emerging memory technologies, has been explored for detecting the correlation between statistical event-based data streams [2]. It exploits the crystallization dynamics of the PCM devices, where the crystallization growth of the phase change material is monotonic with the subsequent application of SET pulses. This monotonic change of device conductance captures the temporal correlation between discrete data streams in parallel, where conductance of each array device shows the correlation of each data stream that arrives at a particular time instant [2].

In another study, a neuromorphic architecture detected the temporal correlation between input streams using a single-neuron computational primitive and a level-tuning concept [5]. The single-neuron computational primitive consists of a phase-change neuron to implement integrate-and-fire mechanism, and an array of PCM devices to work with simplified spike timing dependent plasticity (STDP) learning rule. The array of PCM devices represents the STDP synapses. The weight of a STDP synapse represents the strength of the connection between the pre-neuron and post-neuron [6]. If the weight of STDP synapse is potentiated (the conductance of PCM device is increased), it strengthens the connection between the pre- and post-neurons. Otherwise, if the weight of STDP synapse is depressed, it weakens this connection [6]. The incoming input spikes from pre-neurons, the STDP synapses, the integrate-and-fire-neuron (orange box) and the postsynaptic spikes are shown in Figure 4-1. In an integrate-and-fire-neuron, the incoming spikes of pre-neurons weighted by the STDP synapses, are accumulated to give the total post-synaptic potential (tPSP), which
Figure 4-1: Schematic illustration of an all-memristive computational primitive, adapted from [19]. PCM devices are used for both the neuronal membrane potential and the synapses. The weight of synapse represents the strength of the connection between two neurons. The incoming input spikes to the synapses (orange dots), the array of PCM used as synapses (orange box), the integrate-and-fire neuron (green box), and the postsynaptic spikes (blue dots) are shown. Insets show (a) the neuronal membrane potential of primary neuron and (b) the spike-timing dependent plasticity (STDP) learning rule, where $\Delta w$ (y-axis) represents the change in the weight of STDP synapse based on the timing of the presynaptic spike time and the postsynaptic spike time (x-axis). The $\Delta t$ in the x-axis represents the $(t_{\text{PRE}} - t_{\text{POST}})$ of the presynaptic and postsynaptic spike events. The LTP (green) and LTD (purple) are the long term potentiation and depression curves respectively, where potentiation refers to increase in weight of synapse and depression refers to decrease in weight of synapse. Adapted from [5].

change the neuronal membrane potential of the primary neuron (or post neuron). In [5], the neuronal membrane potential was represented by the conductance of a PCM device (shown in Figure 4-1(a)). When the neuronal membrane potential exceeds a threshold, the primary or post neuron fires a spike, known as the postsynaptic spike. If the pre-neuron fires spikes before post neuron, the weight of the STDP synapse is increased (potentiated), otherwise, the weight is decreased (depressed). This is known as the STDP rule and is shown in Figure 4-1(b). In Figure
the difference between the timing of the pre-neuron spike and the postsynaptic spike represents the x-axis, which dictates whether the weight of the STDP synapse is to be increased (potentiated) or decreased (depressed). This is represented by the $\Delta w$ term, the change in weight, on the y-axis. The level-tuning concept refers to using the total postsynaptic potential (tPSP) of primary neuron to communicate with the dependent output neurons to enable their activity. The correlation between incoming input spikes is detected using this method [5]. The architecture contains one layer feedforward network between the primary neuron and the two level-tuned neurons (output neurons), where the primary neuron strengthens the synaptic connections related to the correlated input spikes (using STDP described above) [5]. Each of the two level-tuned neurons (the output neurons) detects a correlated group. This scheme used 400 synapses for each of the three neurons, where each synapse was connected to an input source.

This class of in-memory computation have been investigated with phase change memory [2][5], and therefore the application of ReRAM for this type of computing remains to be explored. Furthermore, another reason to investigate the correlation detection using ReRAM is that the switching characteristics of phase change memory differ from resistive memory (as described in Chapter 1), even though both exhibit stochasticity in their switching and multiple conductance states. The low power consumption, high switching speed, good CMOS compatibility, high density and fabrication with fab-friendly materials, makes ReRAM an attractive candidate. This work explores the feasibility of temporal correlation detection between binary events using the non-volatile accumulative behavior of ReRAM, as an application of non-von Neumann computing.

4.2. Temporal Correlation Detection using ReRAM arrays

In this work, the possibility of temporal correlation detection was investigated with an empirical ReRAM model [7], which was developed based on the performance metrics of fabricated
HfO2 ReRAM in our lab. The temporal correlation detection was simulated on a 5x5 array of ReRAM in the Python environment. Although the temporal correlation between event-based data streams and input streams has been detected using the crystallization dynamics of phase change memory, it has yet to be investigated with metal-oxide ReRAM. The following sections describe the generation of the binary events and their correlation in Section 4.2.1., temporal correlation detection algorithms and the corresponding simulation results in Python environment with the empirical ReRAM model in Section 4.2.2. and ReRAM experimental data in Section 4.2.3., and the conclusion in Section 4.2.4. The results of Section 4.2.2. has also been published in 2020 IEEE Student Conference on Research and Development (SCoReD) [8]. The results of Section 4.2.3. has been submitted for publication in conference and is under review.

4.2.1. Generation of Correlated and Uncorrelated Processes

The discrete-time binary events can be represented by 0 and 1, where the events occurring gives a 1 in the discrete time instance, k. The Bernoulli distribution was used to generate the binary processes, with a probability of the event occurring, \( P(X=1) = 0.5 \). The Bernoulli distribution gives a discrete-time stochastic process, where there are only two possible outcomes, 1 (success) or 0 (failure), and each Bernoulli trial is statistically independent. It is a special case of the Binomial distribution, where the number of trials, \( n=1 \). However, the probability at each trial is fixed. Since the number of repeated trials of Bernoulli processes was fixed, the number of outcome 1 in the certain number of trials has the binomial distribution, of a fixed probability [9]. The reference binary process, \( X_r(k) \), is based on this distribution, and was later used to generate correlated binary processes. Since there are multiple processes used in this work, we will use \( i \) and \( j \) to distinguish between the individual processes.

\[
P(X(k) = 1) = p
\] (1)
The probability of an event occurring (outcome 1) or not (outcome 0) in a discrete-time process \(X\) at a given time instant \(k\), is shown in equations (1) and (2), as \(p\) and \(q\), respectively. The mean or the expected value, and the variance of the discrete-time process \(X\) is given in equations (3) and (4).

The correlated processes are generated as follows. The reference binary process, \(X_r(k)\), with a probability of \(p\), is first generated. Then, the following equations (5) and (6), are used to generate the correlated processes, based on the conditional probabilities of the binary process \(X_r(k)\), as well as the correlation coefficient \(c\) [2,10].

\[
P(X_i(k) = 1 | X_r(k) = 1) = p + \sqrt{c}(1 - p) \tag{5}
\]
\[
P(X_i(k) = 1 | X_r(k) = 0) = p(1 - \sqrt{c}) \tag{6}
\]

Covariance defines the dependence between random variables, where the normalized covariance gives the correlation coefficient. The covariance between two binary processes, \(X_i(k)\) and \(X_j(k)\), is \(\text{Cov}(X_i(k), X_j(k))\), as given in equation (7). The first term is the expectation of their products and the second term is a product of their means. The relation between the covariance and the correlation coefficient \(c\) of binary processes, \(X_i(k)\) and \(X_j(k)\), is given in equation (8) [2,8,10-11]. The correlation coefficient of identical processes is equal to 1. If the binary processes, \(X_i(k)\) and \(X_j(k)\), are correlated, then their correlation coefficient is positive, i.e. \(c>0\). Otherwise, the
binary processes are uncorrelated. The uncorrelated binary processes are produced using the binomial distribution with probability p, such that they are independent of the reference process X_r(k). All binary processes were generated for a total time instant, K, where K represents the total number of trials. These steps were used to generate correlated and uncorrelated discrete-time binary data streams for simulation with ReRAM model and the experimental ReRAM data. For the latter case, a lower probability of the event occurring, P(X=1)=0.1, was used for generating the correlated and uncorrelated processes.

4.2.2. Temporal Correlation Detection on ReRAM Arrays using the ReRAM Model

4.2.2.1. The ReRAM Model

The resistive random access memory empirical model [7] used in this work, is based on the switching performance data of HfO2 ReRAM in our lab. More information on the empirical model can be found in [7]. The ReRAM model has the following parameters, the switching time, \( t_{sw} \), the SET and RESET threshold voltages, \( V_{tp} \) and \( V_{tn} \), respectively, the low resistance state \( LRS \) and high resistance state \( HRS \), the window parameters \( f_{LRS}(R(t)) \) and \( f_{HRS}(R(t)) \), powers \( P_{LRS} \) and \( P_{HRS} \). These parameters are used to compute \( dM/dt \), the rate of change of memresistance (M) of the ReRAM model with respect to the application of electrical pulses of amplitude \( V(t) \), as shown in equation (9) [7-8]. The threshold voltages of SET and RESET denote the voltages for the onset of SET and RESET operations if applied \( V(t) \) exceeds \( V_{tp} \) or falls below \( V_{tn} \), respectively.
Therefore, if applied $V(t)$ is a positive voltage with amplitude greater than $V_{tp}$, $dM/dt$ reduces the resistance of ReRAM, as SET operation results in creating filament and switching the device to the LRS. Similarly, if applied $V(t)$ is a negative voltage lower than $V_{tm}$, $dM/dt$ increases the resistance of ReRAM, as observed in RESET operations. Otherwise, if the applied voltage $V(t)$ does not satisfy any of the threshold voltages for SET and RESET, the $dM/dt$ is zero, meaning that the applied $V(t)$ results in no resistance change. This is the case for read operations of the ReRAM model.

\[
\frac{dM}{dt} = \begin{cases} 
- \frac{(HRS-LRS)}{t_{sw}} \left(\frac{V(t)-V_{tp}}{V_{tp}}\right)^{P_{LRS}} f_{LRS}(R(t)), & V(t) > V_{tp} \\
\frac{(HRS-LRS)}{t_{sw}} \left(\frac{V(t)-V_{tn}}{V_{tn}}\right)^{P_{HRS}} f_{HRS}(R(t)), & V(t) < V_{tn} \\
0, & \text{Otherwise}
\end{cases} 
\]

(9)

\[
R(t + 1) = R(t) + \frac{dM}{dt} \Delta t
\]

(10)

The window functions $f_{LRS}(R(t))$ and $f_{HRS}(R(t))$, define the non-linearity of the resistance during resistance saturation. It helps ensure that the resistance saturation is limited to the specified LRS or HRS values, and does not exceed this range of resistance. The window parameters for the $f_{LRS}(R(t))$ and $f_{HRS}(R(t))$ functions have been set to 0, to ensure that the final resistance stops at the specified LRS or HRS resistance. This helps maintain the resistance at LRS or HRS once resistance saturates, where subsequent application of SET or RESET pulses have no effect on resistance change. The powers $P_{LRS}$ and $P_{HRS}$ were set to 3 so that the resistance change would be non-linear. The new resistance of the ReRAM, $R(t+1)$, is then updated using $dM/dt$, the initial resistance $R(t)$ of the ReRAM and the pulse width $\Delta t$ of the applied voltage $V(t)$, as given by
equation (10) [8]. The resistance of ReRAM can be gradually modulated between LRS and HRS through the repetitive application of small pulses, voltage pulses $V(t)$ of small pulse width $\Delta t$, in either SET or RESET operations, until the resistance saturates in the low resistance regime or the high resistance regime, respectively. The successive application of SET pulses (of small pulse width) results in gradual resistance decrease from the HRS state towards the LRS state, until the resistance saturates. Similarly, the successive application of RESET pulses results in gradual resistance increase from the LRS state towards the HRS state, until the resistance saturates. This type of monotonic change of resistance can be referred as the non-volatile accumulative behavior of ReRAM. This is an important property of ReRAM that is leveraged for the correlation detection algorithm and is used in the simulation by setting the pulse width $\Delta t$ of applied voltage pulses to 5ns.
4.2.2.2. The Algorithm with ReRAM Model

The temporal correlation detection algorithm, adapted from [2] and modified according to the simulation work, is shown in Figure 4-2. It was implemented in a Python environment. Initially, each of the correlated and uncorrelated processes $X_{ij}$, are assigned to each ReRAM device in the array, where $i$ and $j$ denote the row and column index of the array elements. Therefore, some ReRAM devices are assigned correlated processes while others were assigned uncorrelated processes. $X_{ij}(k)$ is the value of a process at row $i$ and column $j$ at time instance $k$, where the value is either 0 or 1. The gradual resistance modulation in the SET region, which gives a monotonic increase in device conductance (decrease in resistance) for the accumulative behavior, was used for this algorithm. In Fig. 4-2, the entire ReRAM array is RESET first to give HRS as the initial
resistance of each device, so each device begins at low conductance. This is the resistance at the initial time instant $K=0$ [8].

Parameter $M$ represents a critical parameter, called momentum, that depends on the values of all the processes at each given time instant. It is a sum of all the nonzero instantaneous processes, $X_{ij}(k)$, at each time instance $k$. Momentum $M(k)$ is therefore computed in the first loop of Fig. 4-2 for each time instant $k$. Then, $V_{\text{pulse}}$, the amplitude of the SET voltage pulse, is modulated according to momentum $M(k)$ and a constant $Q$. $Q$ is included to ensure that the modulated SET voltage pulse remains within the switching thresholds of the ReRAM. $V_{\text{pulse}}$ is the same as the $V(t)$ term in the ReRAM model. Then, the second loop in Fig. 4-2 decides which devices to send the SET pulse to in the array. If the assigned process at an array device, $X_{ij}(k)$, has a nonzero value at time instant $k$, then this ReRAM device will be applied the SET pulse. The resistance at this ReRAM device is updated using the ReRAM model. Therefore, for a given time instant $k$, all the ReRAM devices whose process $X_{ij}(k)$ is nonzero are applied the SET pulse through this loop. Once all the processes $X_{ij}(k)$ has been checked, it exits the loop. The time instance $k$, is incremented to the next time instant $k+1$, and momentum $M(k)$ is reset to 0. The steps are re-iterated until the final time instance, $k_{\text{final}}$, is reached [8].

Therefore, the resistance or conductance of each ReRAM device in the array evolves over time according to the algorithm and the stochastic processes. The final resistance or conductance of the ReRAM devices holds the result of the computation, such that they distinguish between the correlated and the uncorrelated processes. The resistance of the ReRAM devices are read with a read voltage of -0.2V [8]. If the processes are correlated, their devices will reach the saturated resistance or conductance at the final time instance $k_{\text{final}}$, similar to the behavior observed with phase change memory due to the accumulative characteristic in resistance change.
The reason why the correlated processes should have a greater change of absolute conductance or resistance than the uncorrelated processes may be described as follows. In a covariance matrix of dimension NxN, where N is the total number of processes, each matrix element compares two processes, \(X_i\) and \(X_j\). An estimate of this matrix element over the total time instants is given in equation (11). As each row of the matrix corresponds to a process, the process \(X_i\) for each row is compared with other (N-1) processes in the covariance matrix.

\[
\sigma(X_i, X_j) = \frac{\sum_{k=0}^{k_{final}} X_i(k)X_j(k)}{k_{final}} \quad (11)
\]

\[
\hat{W}_i = \sum_{j=1}^{N} \sigma(X_i, X_j) \quad (12)
\]

Then, a sum across all elements of the row gives the estimated weight \(\hat{W}_i\), of each process \(X_i\), as shown in equation (12). Equations (11) and (12) were adapted from [2]. If the processes are correlated, then the absolute change in their weights are supposed to be higher than that of uncorrelated processes during the final time instant \(k_{final}\). An estimate is obtained for equations (11-12) by iterating the algorithm till final time instant \(k_{final}\), with momentum \(M(k)\) being computed at each time instant from N processes as data is passed through the array. Weight can be represented as the conductance of the non-volatile memory. In the algorithm, these computations for comparing all the processes and the sum of each array element of covariance matrix is performed in parallel for all processes at each time instant such that the conductance (weight) of each ReRAM device shows which of the assigned temporal processes were correlated after a sufficient time. The momentum \(M(k)\) helps compare the processes by adjusting the weights of each process (conductance of device). The accumulative behavior of the non-volatile memory then gives the absolute change in weight over time. Hence, the devices with correlated processes are expected to exhibit higher absolute change of conductance over time. This change of conductance
can be positive or negative depending on whether the SET region or the RESET region, respectively, is used for the non-volatile memory.

4.2.2.3. Results and Discussion

![Conductance (S)](image)

**Figure 4-3:** The conductance of the 5x5 ReRAM array at time instant k=5. © 2020 IEEE, Reprinted, with permission.

![Conductance (S)](image)

**Figure 4-4:** The conductance of the 5x5 ReRAM array at time instant k=20. © 2020 IEEE, Reprinted, with permission.
The possibility of detecting temporal correlation between incoming binary processes was tested with ReRAM, using the ReRAM model described earlier on a 5x5 ReRAM array in Python. The algorithm shown previously in Fig. 4-2 was implemented and the conductance of the array devices at different time instants, k=5, k=20, k=50 and k=70, are shown in Figures 4-3, 4-4, 4-5 and 4-6, respectively [8]. All device conductance were read with a voltage of -0.2V. The correlated processes were assigned to the first 4 devices in first row of the array. The array was reset to a conductance of 20 µS (or a high resistance of 70 kOhms). Then, the algorithm was iterated till final time instant, k=70. The results for k=20, k=50 and k=70, showed that the conductance of all
array devices increased over time, till the devices with correlated processes reached the saturated conductance of 140 µS at the final time instant. This is illustrated by the highest conductance reached for the first four devices in first row. During the same time, the conductance of the rest of the array devices with uncorrelated processes was lower than the saturated conductance. Hence, the conductance of the array devices at the end of the algorithm showed which of the processes were correlated, with each device represented by the ReRAM model. Therefore, it is possible to detect correlation between binary processes with ReRAM, as shown by the simulation [8].

During the simulation of the algorithm, the parameters final time instant $k_{\text{final}}$ and constant Q are critical. For instance, if the $k_{\text{final}}$ is too high it may cause uncorrelated processes to be misclassified as correlated processes. On the other hand, if the $k_{\text{final}}$ is too low, not all the correlated processes present may be found. Additionally, since constant Q helps keep the SET pulse within switching thresholds, too high of constant Q may increase the rate at which the devices reach saturated conductance [8].

4.2.2.4 The Simulation Block Diagram

The simulation block diagram that implemented the temporal correlation detection algorithm in Python environment is shown in Figure 4-7. It shows the organization of five modules, where each module is a class of specific attributes and functions. The attributes are the characteristics present at each module, and the functions control what the module can do. The separate modules are as follows. One module generates the correlated and uncorrelated processes using equations described earlier, based on the correlation coefficient c, the probability $P(X=1)=p$, total number of time instants K, the total number of correlated and uncorrelated processes. The result is stored in a database. This module is called upon by the another module for process generation. The latter module decides where the processes are assigned in the array, computes momentum $M(k)$ and
modulates applied voltage amplitude $V(t)$ accordingly, decides where the pulse $V(t)$ is applied to, reads back the resistance and plots array resistance when the specified time instant is reached. An interface is always needed for communication between the actual ReRAM array and the control unit, so this interface is a module on its own.

Next, a module represents the ReRAM array, where it is defined by the number of rows and columns of the array, and stores the resistance of each array element (ReRAM). Since ReRAM array can only be read from or written to, the function of this module also includes a read or write function. Lastly, when the ReRAM array module receives the voltage amplitude $V(t)$ of pulse width $\Delta t$, it calls upon the module for empirical ReRAM model and updates its ReRAM resistance if it is a write (SET/RESET) pulse. Otherwise, a read operation is performed to extract the resistance of the array element (ReRAM). The module ReRAM model computes resistance change according to the empirical ReRAM model described in earlier section. Hence, each ReRAM in the array follows the characteristics of the ReRAM model. On the other hand, for the later sections with regards to simulation with ReRAM experimental analog data, the empirical ReRAM model class in this module is replaced with another class that stores experimental data. Hence, the resistance of each device is updated with the experimental data when this module is called upon in the later sections.
Figure 4-7: Block diagram of the simulation of the algorithm on a 5x5 ReRAM array in Python. Each module is a class of specific attributes and functions, to generate the processes, to control the algorithm (assign the processes, compute momentum, compute V(t) according to momentum, etc.), the 5x5 ReRAM array, the ReRAM model used and an interface to provide communication between the ReRAM array module and the module implementing the algorithm.

4.2.3. Temporal Correlation Detection on ReRAM Arrays using ReRAM Experimental Data

In this section, the ReRAM model in Figure 4-7 was replaced with real ReRAM experimental data obtained through application of ultra-short pulses of 300ps duration and fixed SET and RESET voltage amplitudes. The temporal correlation detection was implemented through simulation because of equipment limitations in applying such short pulses to an array of devices. The following sections are organized as follows. Details on the analog switching data used in simulation is reported in Section 4.2.3.1, how the algorithm is applied with the experimental data in Section 4.2.3.2, the ideal simulation results and the results with experimental data are discussed.
separately for RESET and SET in Sections 4.2.3.3 and 4.2.3.4, respectively. The energy consumption of temporal correlation detection using both RESET analog data and SET analog data for 25 processes are discussed in Section 4.2.3.5.

4.2.3.1. Electrical Characterization Setup and ReRAM Analog Switching Data

For ultra-short sub-nanosecond pulses, a high frequency electrical measurement setup consisting of Berkeley Nucleonics pulse generator was used, in conjunction with power splitter, RF (Radio Frequency) probes and a 50 Ohm termination resistance between the probes and the power splitter to minimize signal reflection. The generated pulses from pulse generator are fed to the power splitter, where 50% of the input is applied to the top electrode of 1T1R and the other 50% is input to an oscilloscope. The oscilloscope was connected to the other end of power splitter to display the applied pulse train of ultra-short pulses. This measurement setup was used to apply the short pulses to 1T1R devices, where the 1T1R devices required extra pads to accommodate the measurement setup used. Thus, this imposed a limitation to apply ultra-short pulses to an array of connected 1T1R devices. Therefore, the measurement results were used in the simulation of temporal correlation detection.

Ultra-short pulses are preferred for reduced energy consumption of the ReRAM, and has been demonstrated in the literature for tantalum and hafnium oxide ReRAM [12-18]. The overall endurance of the ReRAM is not affected by the reduced pulse widths [13].

For VCM-based ReRAM, the resistance modulation is gradual for RESET operations and the SET operations are abrupt [12,19-20]. This is attributed to thermal runaway that increases the local temperature in the switching layer and is aided by Joule heating to give the abrupt SET behavior [12]. For RESET, though Joule heating is still present, the drift of oxygen vacancies from bottom
electrode (BE) and onset of diffusion current towards the BE results in a more gradual resistance modulation [12,19-20]. In [19], the abrupt SET behavior is described by a voltage divider model as follows. In HRS, there is a depleted gap near the BE, which can be called $\Delta$. The resistance in switching layer in this model is composed of resistance of conductive filament and resistance of $\Delta$. When SET voltage is applied, due to greater thickness and high resistance of $\Delta$ during HRS, the electric field decreases and results in the increase of SET voltage across the $\Delta$. Ionized defects start migrating at SET voltage and decreases thickness of $\Delta$, which increases the electric field and accelerates the migration rate of defects. This results in the abrupt SET behavior of ReRAM.

For the analog switching ReRAM data used in this work, ultra-short sub-nanosecond pulses of fixed pulse width and voltage amplitude were repeatedly applied to obtain the gradual transition from the HRS to the LRS (RESET) and the LRS back to HRS (SET), for a total of 200 pulses, along with a read verify approach. A read verify approach was used to measure the intermediate resistances between each SET/RESET pulse, with -0.3V read voltage pulses of 10 us duration. The SET and RESET voltages were set to 1V and -0.75V respectively, with 300ps duration. A RESET voltage amplitude of -0.75V was chosen as higher RESET voltages with greater pulse amplitudes would result in increased variability in HRS resistance and not give the incremental resistance change [14]. The data used for this simulation was collected in our lab by another graduate student, Minhaz Abedin. The analog switching data used in this work is shown in Figure 4-8. Four RESET and SET cycles, denoted by R1-R4 and S1-S4, respectively, for cycles 1-4, were evaluated with the temporal correlation detection algorithm. The various colors are used to distinguish the data for R1-R4 and S1-S4. Due to the increasing resistance with successive RESET pulses for 200 RESET pulses and decreasing resistance with successive SET pulses for 200 SET pulses, the sub-nanosecond analog data can be used as the accumulative curve required for the temporal
Figure 4-8: Analog switching data using ultra-short 300ps pulses of 200 RESET pulses followed by 200 SET pulses, for four cycles. The four cycles for RESET and SET were denoted by R1-R4 and S1-S4, respectively, using different colors. The SET and RESET voltages used are 1V and -0.75V, respectively, of 300ps duration. After each SET/RESET pulse, a read-verify approach was used to measure the intermediate resistance states.

correlation detection algorithm. These notations R1-R4 and S1-S4 will be used in later sections to refer to the ReRAM data used in the simulation of the algorithm, and their respective impact on the performance of the algorithm will also be discussed.
4.2.3.2. The Algorithm with Analog Switching Data

The algorithm of the temporal correlation detection was modified to accommodate simulation with analog switching data of fixed voltage amplitude and pulse width. The momentum, $M(k)$, is computed using the same method as in Section 4.2.2.2., as the instantaneous sum of processes at time instant $k$. However, in the modified algorithm, the $M(k)$ modulates the number of programming pulses to be applied to ReRAM instead of modulating the pulse amplitude of the applied voltage pulse, as previously done in [2,8]. This is introduced by the parameter $f(M)$, shown in equation (13), which dictates the number of programming pulses to be applied according to the momentum $M(k)$. This

$$f(M) = \begin{cases} 
1 & 1 \leq M(k) < 3 \\
2 & 3 \leq M(k) < 10 \\
3 & 10 \leq M(k) < 15 \\
4 & 15 \leq M(k) < 20 \\
5 & 20 \leq M(k) < 25 \\
0 & \text{Otherwise}
\end{cases} \quad (13)$$

eliminated the need for constant $Q$ that was required in the previous algorithm with ReRAM model in Figure 4-2.

The algorithm of the temporal correlation detection using the RESET analog data and SET analog data from ReRAM are shown in Figures 4-9 and 4-10, respectively. The initial steps are the same, whereby the correlated and uncorrelated processes are assigned to each ReRAM device in the array, with their rows and columns denoted by $i$ and $j$, respectively. The parameters time
Figure 4-9: Algorithm of the simulated temporal correlation detection with reset cycles R1-R4 of the ReRAM analog switching data using 200 RESET pulses of 300ps pulse width.

instance k, momentum M and the number of pulses applied to ReRAM device, n, are initialized. The parameter n is introduced to account for the 200 pulses in the dataset, such that the ReRAM resistance saturates at the 200th applied pulse and remains at this resistance for subsequent pulses (when more than 200 pulses are applied). At initial time instance k=0, the entire ReRAM array is SET using the last data point of its SET cycle from SET analog data in Figure 4-9. Similarly, for Figure 4-10, the entire ReRAM array is RESET using the last data point of its RESET cycle from
Figure 4-10: Algorithm of the simulated temporal correlation detection with set cycles S1-S4 of the ReRAM analog switching data using 200 SET pulses of 300ps pulse width.

RESET analog data at k=0. This gives the devices a high conductance for Figure 4-9 and a low conductance for Figure 4-10, at the beginning of the algorithm. Then, momentum M(k) is computed in the first loop. This determines y, according to \( f(M) \) in equation (13). The inner loops after \( f(M) \) does the following. If the process \( X_{ij}(k) \), at the ReRAM \( R_{ij} \), has a nonzero value at given time instance k, then \( R_{ij} \) will be applied the number of y RESET pulses for Figure 4-9. Similarly, \( R_{ij} \) will be applied the number of y SET pulses for Figure 4-10. The parameter pulse_no
ensures that \( y \) number of pulses are applied to the ReRAM device \( R_{ij} \). As each RESET (SET) pulse is applied, the resistance of \( R_{ij} \) is updated from the RESET (SET) cycle analog data, \( R_{1-4} \) \( (S_{1-4}) \). The loops also ensure that the resistance at each cell is updated with the analog RESET (SET) data till the 200\(^{th} \) applied pulse, after which the resistance saturates. This is checked for all the 25 processes. Then, the time instance is incremented and the algorithm is iterated till final time instance \( k_{final} \). The algorithm is repeated separately for \( R_{1-4} \) and \( S_{1-4} \) with the same processes, which were generated with a probability \( P(X=1)=0.1 \) and correlated with a correlation coefficient of 0.8 using the method described in Section 4.2.1.

4.2.3.3. Results and Discussion of Temporal Correlation Detection with RESET Analog Data

The monotonic change in resistance in the RESET analog switching data, namely the four RESET cycles \( R_{1-4} \), served as the accumulative curve required for the algorithm. Each RESET cycle, \( R_{1-4} \), were implemented with the algorithm shown in Figure 4-9, to determine 10 correlated processes from the 25 processes in a 5x5 ReRAM array. The 10 correlated processes

![Figure 4-11](image-url)

**Figure 4-11:** The ideal results with temporal correlation detection algorithm on a 5x5 ReRAM array using RESET analog switching data. Correlated processes are assigned to the first two rows of the array. The rest of the array are assigned uncorrelated processes. The ReRAM devices at these correlated processes, represented by the blue squares, are expected to go from high conductance to low conductance, while the rest of the array, represented by yellow squares, are expected to have higher conductance than the first two rows of the array at the end of the algorithm. In other words, the blue and yellow boxes represent devices at low and high conductance, respectively, at final time instance \( k_{final} \).
were assigned to the top two rows, while the rest of the array were assigned uncorrelated processes.
The ideal results of this algorithm using the accumulative behavior in the RESET regime, at the final time instance k_{final}, is shown in Figure 4-11. The conductance of the devices with correlated processes are expected to decrease with the iterations of the algorithm till final time instance k_{final}, where the overall absolute change of conductance for these devices are expected to exceed that of devices with uncorrelated processes. Figure 4-11 shows the conductance of 25 devices at k_{final}, where their conductance distinguishes the correlated processes (low conductance represented by blue boxes) from the uncorrelated processes (high conductance represented by yellow boxes). The reason behind why the greater absolute change of conductance is expected for devices with correlated processes has been explained in Section 4.2.2.2.

The temporal correlation detection results with the four RESET cycles R1-R4 of the analog data are shown for time instances k of 500, 600, 900, 1200 and 1300 in Figures 4-12, 4-13, 4-14, 4-15 and 4-16 respectively. The following sections discuss the results with respect to each RESET cycle. The same 25 processes were used with the four RESET cycles R1-R4.

4.2.3.3.1. Temporal Correlation Detection Results with R1 RESET Cycle Analog Switching Data

For reset cycle R1 of analog data, the resistance is incremented gradually from 11 kΩ to 19 kΩ with the first 130 pulses, hits a peak, then it undergoes a decrease in resistance till the 200th pulse, as shown in black in Figure 4-8. As the algorithm progresses, it resulted in greater absolute resistance change for devices with correlated processes using the ReRAM data for the first 130 pulses of the incremental RESET dataset, and successfully detected all the correlated processes from the uncorrelated processes at k=500 and k=600 as shown in Figures 4-12(a) and 4-13(a), respectively. As the algorithm iterates to higher time instances k of 900, 1200 and 1300, shown in Figures 4-14(a), 4-15(a) and 4-16(a), the algorithm fails to detect the correlated processes due to
resistance decrease after the 130 pulses were applied in R1 analog data. The results show that the gradual resistance modulation in increasing resistance in the RESET is important for the algorithm to work, where the change in conductance is obvious for the correlated processes at this RESET regime. At the time instances k of 500 and 600 where the algorithm detected the correlated processes, the absolute difference in median values of the conductance (resistance) of the correlated and uncorrelated processes is 20 µS (4070 Ω). The median differences in conductance and resistance are calculated directly from the data and does not correspond to direct reciprocal of difference.

![Figure 4-12](image)

**Figure 4-12:** Conductance of 5x5 ReRAM array at k=500 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data, the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes; d) Conductance of 5x5 ReRAM array at k=500 using analog R4 reset cycle data.
4.2.3.3.2. Temporal Correlation Detection Results with R2 RESET Cycle Analog Switching Data

For reset cycle R2 of analog data, the resistance is incremented gradually from 10.5 kΩ to around 17 kΩ with 200 successive RESET pulses. Despite the variability present, it has the monotonic increase in resistance that is needed for the algorithm as shown in blue in Figure 4-8, and hence worked for the entire data range of 200 pulses with the algorithm. The correlated processes were detected at the later time instances k of 900 and 1200, as shown in Figures 4-14 (b) and 4-15(b), respectively. After that, at time instance k of 1300, shown in Figure 4-16 (b), some

![Figure 4-13](image)

**Figure 4-13:** Conductance of 5x5 ReRAM array at k=600 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data d) R4 reset cycle data. For (a) and (d), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R1 and R4 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.
of the conductance of devices with uncorrelated processes may have become lower than the saturated conductance of correlated devices at the 200th pulse, where this final data point in R2 data (resistance of devices with correlated processes) is slightly lower than the highest resistance in R2 data. Hence, the algorithm could not detect the correlated processes at k=1300. On the other hand, the reason why the correlated processes were not detected at earlier time instances of k of 500 and 600 in Figures 4-12 and 4-13, respectively, may be due to the variability present in the R2 RESET cycle analog data. Similar data trends like R2 RESET cycle data also worked well (78%) with the algorithm. At the time instances k of 900 and 1200 where the algorithm detected the correlated processes, the absolute median difference between the conductance (resistance) of the

![Figure 4-14: Conductance of 5x5 ReRAM array at k=900 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data and d) R4 reset cycle data. For (b), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R2 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.](image-url)
correlated and uncorrelated processes is 15 µS (3040 Ω). The median differences in conductance and resistance are calculated separately from the data and does not correspond to direct reciprocal of difference.

4.2.3.3.3. Temporal Correlation Detection Results with R3 RESET Cycle Analog Switching Data

For reset cycle R3 of analog data, the resistance is incremented gradually from 10 kΩ to around 16 kΩ with 100 successive RESET pulses, then maintains resistance around 14 kΩ with standard deviation of 830 Ω for the next 100 successive RESET pulses. The temporal correlation detection

![Figure 4-15](image)

**Figure 4-15**: Conductance of 5x5 ReRAM array at k=1200 using analog a) R1 reset cycle data b) R2 reset cycle data c) R3 reset cycle data and d) R4 reset cycle data. For (b), the conductance of the correlated processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus, the temporal correlation detection algorithm worked with R2 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.
algorithm worked well with the gradual resistance modulation using the first 100 pulses of R3
RESET data, where it detected all the correlated processes at time instance k of 500 as shown in
Figure 4-12 (c). Surprisingly, the algorithm also detected the correlated processes at later time
instance k of 1300 as shown in Figure 4-16 (c), probably due to the last datum of R3 RESET data
being among the highest data points among the 200 pulses, so that the saturated conductance of
the devices with correlated processes still gave greater change in absolute conductance. Hence, the
algorithm worked at k=1300. For the time instances in between k=500 and k=1300, shown for k

![Figure 4-16](image)

Figure 4-16: Conductance of 5x5 ReRAM array at k=1300 using analog a) R1 reset cycle data b) R2 reset
cycle data c) R3 reset cycle data and d) R4 reset cycle data. For (c), the conductance of the correlated
processes are lower than the conductance of the uncorrelated processes, since RESET data was used. Thus,
the temporal correlation detection algorithm worked with R3 analog data as it resulted in greater absolute
change in conductance for the correlated processes compared to the uncorrelated processes.

of 600, 900, 1200 in Figures 4-13 (c), 4-14 (c), 4-15 (c), respectively, the algorithm failed to detect
the correlated processes due to no resistance modulation in the last 200 pulses of the R3 RESET
analog data. These results show that the gradual resistance modulation in increasing resistance in
the RESET is important for the algorithm to work. At the time instances \( k \) of 500 and 1300 where
the algorithm detected the correlated processes, the absolute median difference between the
conductance (resistance) of the correlated and uncorrelated processes is 9.9 \( \mu S \) (1800 \( \Omega \)). The
median differences in conductance and resistance are calculated separately from the data, and does
not correspond to direct reciprocal of difference.

4.2.3.3.4. Temporal Correlation Detection Results with R4 RESET Cycle Analog Switching Data

For reset cycle R4 of analog data, the resistance is incremented gradually from 11.4 k\( \Omega \) to 15
k\( \Omega \) with the first 130 successive RESET pulses, then the resistance decreases till the 200\(^{th}\) pulse,
as shown in light blue color in Figure 4-8. The temporal correlation detection algorithm worked
with the gradual resistance modulation using the first 130 pulses of R4 RESET data, where it
detected all the correlated processes at time instance \( k \) of 600 as shown in Figure 4-13(d). As
expected, the decrease in resistance after application of 130 RESET pulses using R4 RESET data
caused the algorithm to fail to detect the correlated processes at the later time instances \( k \) of 900,
1200, 1300 in Figures 4-14(d), 4-15(d), 4-16(d), respectively. These results again show that the
gradual resistance modulation in increasing resistance in the RESET is important for the algorithm
to work. At the time instance \( k \) of 600 where the algorithm detected the correlated processes, the
absolute median difference between the conductance (resistance) of the correlated and
uncorrelated processes is 9 \( \mu S \) (1700 \( \Omega \)). The median differences in conductance and resistance
are calculated directly from the data and does not correspond to direct reciprocal of difference.
4.2.3.4. Results and Discussion of Temporal Correlation Detection with SET Analog Data

The monotonic change in resistance in the SET analog switching data, namely the four SET cycles S1-S4, served as the accumulative curve required for the algorithm. Each SET cycle, S1-S4, were implemented with the algorithm shown in Figure 4-10, to determine 10 correlated processes from the 25 processes in a 5x5 ReRAM array. The 10 correlated processes were assigned to the top two rows, while the rest of the array were assigned uncorrelated processes. The ideal results of this algorithm using the accumulative behavior in the SET regime, at the final time instance kfinal, is shown in Figure 4-17. The conductance of the devices with correlated processes are expected to increase with the iterations of the algorithm till final time instance kfinal, where the overall absolute change of conductance for these devices are expected to exceed that of devices with uncorrelated processes. Figure 4-17 shows the conductance of 25 devices at kfinal, where their conductance distinguishes the correlated processes (high conductance represented by yellow boxes) from the uncorrelated processes (low conductance represented by blue boxes).

The temporal correlation detection results with the four SET cycles S1-S4 of the analog data are shown for time instances k of 130, 317 and 1200 in Figures 4-18, 4-19 and 4-20 respectively. The same 25 processes were used with the four SET cycles S1-S4. The following sections discuss the results with respect to each SET cycle.
Figure 4-17: The ideal results with temporal correlation detection algorithm on a 5x5 ReRAM array using SET analog switching data. Ten correlated processes are assigned to the first two rows of the array, and the rest of the array are assigned uncorrelated processes. The ReRAM devices at these correlated processes, represented by the yellow boxes, are expected to go from low conductance to high conductance, while the rest of the array, represented by blue boxes, are expected to have lower conductance than the first two rows of the array at the end of the algorithm. In other words, the blue and yellow boxes represent devices at low and high conductance, respectively, at final time instance $k_{final}$.

4.2.3.4.1. Temporal Correlation Detection Results with S1 SET Cycle Analog Switching Data

For set cycle S1 of analog data, the resistance decreases gradually from 14 k$\Omega$ to 11.5 k$\Omega$ with the first 35 successive SET pulses and maintains around this resistance for next 55 successive SET pulses, then had two dips in resistance between 11.2 k$\Omega$ and 10.5 k$\Omega$ with an average resistance of 11 k$\Omega$ with standard deviation of 410 $\Omega$ till the 200th pulse, as shown in red in Figure 4-8. Only around 10 data points in the first 20 pulses were higher than the rest of the resistances, and no resistance modulation for subsequent pulses, followed by two dips till the 200th pulse, gave poor results with the algorithm. At best, this SET analog data detected 90% of the correlated processes during the first 35 pulses at time instance $k$ of 130, as shown in Figure 4-18 (a). The algorithm failed to detect the correlated processes at the later time instances $k$ of 317 and 1200 in Figures 4-19 (a) and 4-20 (a), respectively, due to no monotonic change in resistance after the first 35 SET pulses in the S1 analog data. However, similar data trends for incremental SET pulses that
Figure 4-18: Conductance of 5x5 ReRAM array at k=130 using analog a) S1 SET cycle data b) S2 SET cycle data c) S3 SET cycle data and d) S4 SET cycle data. For (a), the conductance of 9 correlated processes are lower than the conductance of the uncorrelated processes, since SET data was used. Thus, the temporal correlation detection algorithm worked 90% with S1, 50% with S2 analog data as it resulted in greater absolute change in conductance for the correlated processes compared to the uncorrelated processes.

decreases resistance monotonically with measurable resistance differences between initial and later cycles resulted in better performance with the algorithm. At k=130, where the algorithm detected 90% of correlated processes, the difference in median of the conductance (resistance) of the correlated and uncorrelated processes using S1 set cycle analog data is 7 µS (915 Ω). The median differences in conductance and resistance are calculated directly from the data and does not correspond to direct reciprocal of difference.

4.2.3.4.2. Temporal Correlation Detection Results with S2 SET Cycle Analog Switching Data

For set cycle S2 of analog data, the resistance decreases gradually from 14 kΩ to 10.5 kΩ with the first 70 successive SET pulses, then maintains this resistance due to resistance plateau at 10.5
\[ \pm 0.5 \, \text{k}\Omega, \text{as shown in green in Figure 4-8.} \]
Therefore, the temporal correlation detection algorithm detected 50% and 100% of the correlated processes during the first 70 SET pulses at time instances \( k \) of 130 and 317, shown in Figures 4-18 (b) and 4-19 (b), respectively. Due to no resistance modulation at subsequent SET pulses after the 70 pulses in the S2 analog data, the algorithm failed to detect the correlated processes at the later time instance \( k \) of 1200 shown in Figure 4-20(b). The last datum is also at 10.3 \( \text{k}\Omega \), which is near the average resistance of the plateau so it did not give much resistance difference after resistance saturation at 200\( ^{\text{th}} \) pulse. At \( k=317 \), where the algorithm detected all the correlated processes, the difference in median of the conductance (resistance) of the correlated and uncorrelated processes using S2 set cycle analog data is 12 \( \mu \text{S} \) (1600 \( \Omega \)). The median differences in conductance and resistance are calculated directly from the data and does not correspond to direct reciprocal of difference.

### 4.2.3.4.3. Temporal Correlation Detection Results with S3 SET Cycle Analog Switching Data

For set cycle S3 of analog data, the resistance decreases gradually from 14 \( \text{k}\Omega \) to 11.2 \( \text{k}\Omega \) with the first 20 successive SET pulses, then maintains this resistance due to resistance plateau at 11.2 \( \pm 0.43 \, \text{k}\Omega \). The algorithm detected all the correlated processes at time instance \( k \) of 1200, as shown in Figure 4-20 (c). This is because the final datum of S3 analog data is lower than the rest of data in the 200 SET pulses, at 10.4 \( \text{k}\Omega \). Therefore, when resistance saturated at the 200\( ^{\text{th}} \) SET pulse, it still gave larger absolute resistance difference for the algorithm to work. Data trends similar to this trend worked 77% with the correlation detection algorithm. However, the algorithm could not detect all the correlated processes at lower time instances \( k \) of 130 and 317, shown in Figures 4-
Figure 4-19: Conductance of 5x5 ReRAM array at k=317 using analog a) S1 SET cycle data b) S2 SET cycle data c) S3 SET cycle data and d) S4 SET cycle data. For (b), the conductance of all correlated processes are lower than the conductance of the uncorrelated processes, since SET data was used. Thus, the temporal correlation detection algorithm worked with S2 analog data as it resulted in greater absolute change in conductance for all the correlated processes compared to the uncorrelated processes.

18 (c) and 4-19 (c) respectively, due to the variability present at the plateau in most of the data below 200 SET pulses of the S3 analog data. At k=1200, where the algorithm detected all the correlated processes, the difference in median of the conductance (resistance) of the correlated and uncorrelated processes using S3 set cycle analog data is 8.3 $\mu$S (990 $\Omega$). The median differences in conductance and resistance are calculated directly from the data and does not correspond to direct reciprocal of difference.
Figure 4-20: Conductance of 5x5 ReRAM array at k=1200 using analog a) S1 SET cycle data b) S2 SET cycle data c) S3 SET cycle data and d) S4 SET cycle data. For (c), the conductance of all correlated processes are lower than the conductance of the uncorrelated processes, since SET data was used. Thus, the temporal correlation detection algorithm worked with S3 analog data as it resulted in greater absolute change in conductance for all the correlated processes compared to the uncorrelated processes.

4.2.3.4.4. Temporal Correlation Detection Results with S4 SET Cycle Analog Switching Data

For set cycle S4 of analog data, the resistance decreases gradually from 12.3 kΩ to 10.4 kΩ with the first 100 successive SET pulses, then resistance starts increasing back to 11.9 kΩ for subsequent SET pulses, as shown in brown in Figure 4-8. The algorithm did not work with S4 analog data for time instances k of 130, 317 and 1200 shown in Figures 4-18(d), 4-19(d), 4-20(d), respectively, as the overall data trend started increasing in the next 100 cycles instead of decreasing till it reached some of the highest data points in initial cycles, hence not giving much absolute change in resistance. At best, the algorithm only detected 50% of the correlated processes during the first 50 pulses at time instance k of 255. At k=255, the difference in median of the conductance
(resistance) of the correlated and uncorrelated processes using S4 set cycle analog data is 3 µS (450 Ω). The median differences in conductance and resistance are calculated directly from the data and does not correspond to direct reciprocal of difference.

Note that the median difference between the correlated and uncorrelated processes using the S4 set analog data (3 µS), which detected 50% of correlated processes, is lower than those of S1-S3 set analog data (7 µS, 12 µS, 8.3 µS) and R1-R4 reset analog data (20 µS, 15 µS, 9.9 µS and 9 µS). Also note that the median difference between correlated and uncorrelated processes using S1 set analog data, where 90% of the correlated processes were detected, is 7 µS, which is also lower than those of S2-S3 set analog data and R1-R4 reset analog data where 100% of correlated processes were detected.

4.2.3.5. Energy Consumption and Speed of Temporal Correlation Detection using ReRAM

For the temporal correlation detected for 25 processes using the four R1-R4 RESET cycle analog data and the four S1-S4 SET cycle analog data, the energy consumption for each cycle was computed. This is done by integrating the I-V, the pulse duration of applied SET and RESET pulses, total number of pulses applied for all 25 ReRAM devices. The average energy consumption of temporal correlation detection for 25 processes using RESET cycles and SET cycles were 27.7 pJ and 41 pJ, respectively. The energy consumption using SET regime of ReRAM is higher than RESET regime due to higher SET voltage used (1V), as opposed to lower RESET voltage amplitude (-0.75V) used. If all the correlated processes are detected within the same time instant k, the energy consumption of SET pulses was still higher than that of RESET pulses due to higher applied voltage amplitude, and lower programmed resistance which increases current and hence the corresponding energy consumption.
For 25 processes, the temporal correlation detection using SET regime of phase change memory (PCM), is estimated to be 1.47 μJ [2]. Therefore, the temporal correlation detection for 25 processes using ReRAM gives 36000 and 53000 times lower energy consumption than that of PCM, using the SET regime and RESET regime of ReRAM, respectively. Furthermore, due to the fast switching speed of ReRAM, the execution time of temporal correlation detection using RESET analog data and SET analog data are 629ps and 475ps, respectively. This results in a speed-up of the execution time by 1600-2100 times than that of 1xPOWER8 CPU [1 thread], for detecting temporal correlation between 25 processes [2]. The execution time of temporal correlation detection using ReRAM is similar to PCM, in that it accelerates the computation by 200 folds as opposed to carrying out this task on 4 GPU cores [2]. The lower energy consumption of ReRAM, along with its high switching speed and non-volatile accumulative behavior, makes ReRAM a very promising candidate for this high-level computation of temporal correlation detection.

4.3 Conclusion

The temporal correlation detection algorithm successfully detected four correlated discrete-time binary processes through Python simulation with an empirical ReRAM model to check the feasibility of the implementation of this algorithm with ReRAM non-volatile memory. In addition, the non-volatile accumulative behavior of our HfO₂ devices, exhibited in the analog HfO₂ ReRAM data, obtained through electrical characterization with 300 picosecond pulses for both RESET and SET regimes, were used to implement with the algorithm in Python environment simulation due to equipment limitations. Four RESET and SET cycles, R1-R4 and S1-S4, of -0.75V and 1V amplitudes respectively, and of 300 ps pulse widths, were evaluated with the temporal correlation detection algorithm to detect ten correlated processes on the 5x5 ReRAM array, using the same correlated and uncorrelated processes. The results showed that the gradual
resistance modulation in either SET or RESET accumulative behavior with 200 successive pulses of 300 ps duration worked with the algorithm, where all the ten correlated processes out of the 25 processes were detected successfully. However, the algorithm failed to detect all the correlated processes present using regions of analog data for RESET cycles R1-R4 and SET cycles S1-S4 when there was no resistance modulation and if the change in resistance was not monotonic due to variability of ReRAM. This proves the importance of the monotonic gradual change in resistance of ReRAM that is needed for the algorithm to work, to result in greater absolute resistance change for ReRAM devices with correlated processes than those with uncorrelated processes. The simulation results with the experimental ReRAM analog data also showed that the RESET cycles R1-R4 worked better with the algorithm, compared to those with SET cycles S1-S4, as evident in the median differences between device conductance of correlated and uncorrelated processes. The median differences using analog data cycles that detected 100% of correlated processes with the algorithm were 8.3 µS - 20 µS, which is notably higher than of those that detected 90% and 50% of correlated processes, where the median differences were 7 µS and 3 µS, respectively. This could be due to the more gradual RESET behavior of ReRAM over the 200 RESET pulses and the abrupt SET behavior of ReRAM, which gave fewer pulses for the gradual resistance modulation in the SET regime, as a result of different conduction mechanisms present in RESET and SET operations respectively. Overall, the algorithm still detected the correlated processes successfully from the uncorrelated processes using the ReRAM analog data, and stored the result of the computation directly in the ReRAM memory.

The energy consumption of temporal correlation detection for 25 binary processes using ReRAM is 36000-53000 times lower than that of phase change memory and also resulted in a speed-up of the execution time by 1600-2100 times than that of 1xPOWER8 CPU [1 thread], for
the same number of processes. The execution time of this computation of ReRAM is similar to that of PCM, where it also accelerates the execution time by 200 folds compared to 4 GPU cores. Therefore, the lower energy consumption of ReRAM, along with its high switching speed and non-volatile accumulative behavior, makes ReRAM a very promising candidate for this high-level computation of temporal correlation detection.

4.4 Acknowledgment

The authors acknowledge the CNSE fabrication facility for CMOS integrated ReRAM device fabrication on 300 mm wafer platform; Karsten Beckmann (CNSE Integration Engineer), Wilkie Olin-Ammentorp (UC-San Diego) and Sherif Amer (UT-Knoxville) for their helpful discussion on this project; Minhaz Abedin (SUNY Poly CNSE), for collecting the data used for this project. This research work is sponsored by the National Science Foundation (NSF) under agreement number 1823015.

4.5 References


Chapter 5: Flow-Based Computing on 65 nm CMOS Integrated HfO$_2$ based ReRAM Arrays

Most of the data discussed in this chapter were previously published in IEEE TCAS I: Regular Papers and presented in AVS 66th International Symposium and Exhibition (all written by the author Sarah Rafiq):


5.1. Introduction

As one of the approaches to non-von Neumann computing, this chapter will focus on applications of computer vision and Boolean logic with respect to flow-based computing. It is a type of approximate computing, where some trade-offs in accuracy of computation is acceptable to gain performance at lower power [1-4]. The ever-increasing amount of data generated by Internet-of-Things (IoT) and mobile devices, led to the growing demand of low power budget, high scalability and high performance needed by devices to compute data-intensive applications on the traditional von-Neumann architecture [5]. The data transport of operands in von-Neumann
architecture incurs more energy than the computation on the operands [6]. Due to the in-memory computation capability of ReRAM crossbars, the flow-based computing takes advantage of ReRAM devices to compute inside the memory and hence is an alternative to the von-Neumann architecture. These aforementioned stringent requirements make ReRAM a very promising non-volatile memory (NVM) candidate that can achieve these through its multiple resistance states (a.k.a. multilevel cell (MLC)), low power consumption and high access speeds, high scalability, due to its easy integration with CMOS, its simple structure, and fabrication with fab-friendly materials. Therefore, the benefits of these ReRAM devices and their possible logic-in-memory applications hold great potential to overcome the Von-Neumann bottleneck [3,6], and result in increase in performance in terms of speed and energy consumption [4].

5.1.1. Background

Flow-based computing controls flow of sneak path currents in two-dimensional crossbar arrays of non-volatile memory to do computation [7-8]. Crossbar arrays have a set of parallel row nanowires and another set of parallel column nanowires, where the two sets are at right angle to each other. A non-volatile memory is located at the intersection of each row and column nanowire. The top electrodes of the two-terminal non-volatile memory devices in the array are shared by the row nanowire while the bottom electrodes of the NVM devices are shared along the column nanowires. The data for computation is stored on the resistance states of the non-volatile memory. It thus takes advantage of sneak path currents that occur naturally in crossbar arrays to compute where the data is stored.

Sneak path currents are generally an issue in crossbar arrays, where they are unintended currents that flow in parallel to the device being programmed and may lead to incorrect readout of cell [9-10]. For instance, if the NVM cell to be read is at a high resistance state, and there are three
other cells at low resistance states in the array, the applied read pulse will flow through the alternate path of lower resistance instead of just flowing through the desired cell. Thus, the extra current from the alternate parallel path will be added to the read current of the desired cell, leading to incorrect readout. This is demonstrated in Figure 5-1, where a sneak path current occurring in a two-dimensional crossbar array, is shown in red dashed line in Figure 5-1, when a read pulse $V_{\text{read}}$ is applied to the cell to be read. The desired read current flowing through cell to be read is denoted by blue line in Figure 5-1.

**Figure 5-1:** The read current flowing through desired ReRAM cell is shown in blue while the sneak path current flowing in parallel to device being read is shown in red on the memory crossbar array. Sneak path currents would contribute to the read current of desired read cell, giving an incorrect readout. Hence, they are an issue in passive crossbar arrays.

Both read current (blue line) and sneak path current (red line) will accumulate at the measured ground and are hence a hindrance in passive crossbar arrays. However, the flow-based computing approach makes use of sneak path currents to do computation. Therefore, it requires passive arrays to enable sneak path currents to flow in the memory array.
Additionally, flow-based computing can be employed on ReRAM crossbar arrays, which is a common configuration to support various types of in-memory computations. ReRAM crossbars have been reported to enable vector-matrix multiplication (VMM) [11], stateful logic [12-13] and complementary resistive switching (CRS) [14-15], etc. Work has been reported on implementing approximate computations for Boolean logic functions on unique memristor crossbar designs using flow-based computing through Binary Decision Diagrams (BDD) [7], Reduced Ordered Binary Decision Diagram (ROBDD) [8], formal methods [16], free binary decision diagrams (FBDDs) [17-19] and massively parallel simulated annealing search algorithm [20-21]. As such, memristor crossbars have been designed to compute for 1-bit full adder [16, 22], 3-bit parity and 4-bit parity [16], compact 8-bit adder [23] Boolean logic, as well as edge detection [20-21, 24-26] through flow-based computing. A 1-bit full adder has been implemented experimentally on ReRAM crossbars [22].

The concept of flow-based computing can be described as follows. The states of the memristors at each crosspoint will hold either a high resistance state, \( R_{\text{off}} \), denoted by logic 0, or a low resistance state, \( R_{\text{on}} \), denoted by logic 1. The crossbar design will dictate the states of the memristors in the crossbar, where it will map the variables of the computation onto the memristors and keep the other memristors in \( R_{\text{off}} \) or \( R_{\text{on}} \) state. If the memristor is in the \( R_{\text{on}} \) state, it promotes flow of current between row and column nanowires and vice versa. Otherwise, if the memristor is in \( R_{\text{off}} \) state, it minimizes/prohibits current flow from row nanowire to column nanowire and vice versa. Therefore, in flow-based computing, the desired current will flow according to the states of the memristors and reach the output nanowire if the output of computation is a logic 1. Otherwise, if the output of computation is a logic 0, it is not expected to reach output nanowire and hence would give lower measured current at the output.
Applications of flow-based computing in computer vision and Boolean logic such as edge detection and XOR logic will be discussed in this work. Sections 5.2 and 5.3 will discuss electrical characterization setup and the methodology of implementing the flow-based computing, respectively, Section 5.4 will discuss the results of flow-based computing for applications in computer vision and Boolean logic, Section 5.5 will conclude this chapter and Section 5.6 will discuss comparison of our work to other approaches.

5.2. Electrical Characterization Setup

The electrical characterization of flow-based computing was performed on 8x8 arrays of hafnium oxide 1-transistor-1-ReRAM (1T1R) using two test setups. The ReRAM has been integrated with 65nm CMOS technology on a 300mm wafer platform at SUNY Polytechnic Institute. The ReRAM stack comprises of TiN/Ti/HfO$_2$/TiN, where the top and bottom electrodes are TiN, Ti is the oxygen exchange layer (OEL), and HfO$_2$ is the switching material. Hafnium oxide ReRAM is very promising for this application due to its low power consumption, low operating voltages and high switching speed.

An 8x8 1T1R array was used for this work due to its ability to program individual array devices accurately to target resistance states enabled by the integrated transistors on-chip, by reducing parasitic capacitance of wires and preventing overshoot currents [27-28]. The transistors act as selectors to provide isolation of the selected device during programming. In addition to this, all the transistors in the 1T1R arrays can be biased simultaneously to accommodate sneak path currents. This is demonstrated in Figure 5-2, where only programming current (shown in blue line) flows through intended device when its transistor is turned on while the rest of transistors are turned off. Meanwhile, for flow-based computing, the gates of the transistors can
Figure 5-2: Illustration of how a 1-transistor-1-ReRAM (1T1R) array can be used to both accurately program 1T1R device and accommodate sneak path currents. The blue line shows the programming current flowing though device when its transistor is biased but the other transistors are turned off, so that transistor provides isolation and enables accurate programming. The red line shows the sneak path current flowing in the 1T1R array when all the gates of its transistors are biased simultaneously to give passive array. This is useful for flow-based computing.

Figure 5-3: a) The schematic and b) the layout of the 8x8 1-transistor-1-ReRAM HfO₂ array. The array has 24 pads (2x12), consisting of eight source lines S1-S8, drain lines D1-D8 and gate lines G1-G8. For flow-based computing, the number of gates corresponding to flow-based nanoscale crossbar design were biased to give passive array and accommodate the flow of sneak paths.
be biased simultaneously to accommodate sneak path currents, shown by red line in Figure 5-2. This therefore gives a passive array during flow-based computing. Hence, the required number of gates to be biased depends on the dimension of the flow-based nanoscale crossbar design, and thus two test setups were used for edge detection and XOR logic.

Our 8x8 1T1R arrays consist of 24 pads; eight source pads S1-S8, eight drain pads D1-D8 and eight gate pads G1-G8. The schematic of the 8x8 1T1R array is shown in Figure 5-3(a) [26]. Each array device is accessed through its corresponding source, drain and gate pads [26]. To program (SET/RESET) or read each array device, the top electrode of the 1T1R is applied the programming and read signals while the bottom electrode, source of access transistor is grounded. The top electrodes of 1T1R devices are shared along the rows, while their bottom electrodes and the gates are shared along the columns [26]. The layout of the 8x8 1T1R array and its 2x12 pad structure are shown in Figure 5-3(b). For flow-based computing, subarrays of the 8x8 1T1R array were chosen if the nanoscale crossbar design is small.

The flow-based computing for edge detection consisted of an 8x8 nanoscale crossbar design, and hence the entire 8x8 1T1R array was employed to implement flow-based edge detection. The flow-based computing was performed using Agilent E5270A parametric analyzer, Keithley 707A switching matrix, Celadon Ultra High Performance Probe Cards and SUSS MicroTec semi-automated probe station, as shown in Figure 5-4. Figure 5-4 (a) shows the electrical characterization setup with these instruments, and their communication is controlled by a computer through GPIB. The probe card is installed in the highlighted area on the probe station. Electrical characterization of a 300mm wafer consisting of HfO2 1T1R arrays using a probe card on the semi-automated probe station is shown in Figure 5-4 (b). The inset of Figure 5-4 (b) shows the schematic (on left) and transmission electron micrograph TEM (on right) of the HfO2 ReRAM
integrated with NFET at metal 1/via 1 interface, while the ReRAM device stack composition, TiN/Ti/HfO2/TiN, is depicted in the schematic. The size of fabricated ReRAM is 100nm x 100nm [26]. A custom Python program was designed to specify what voltages are to be applied from the E5270A, and connect through the switching matrix to apply these voltages to correct pins on the probe card. The switching matrix coordinated the applied voltages from E5270A to the probe card pins. The probe card is a 2x12 (24 pins) to probe all 24 pads of the 8x8 1T1R array shown in Figure 5-3, to program devices, and bias all eight gates simultaneously to facilitate flow-based computing. During flow-based computing, the input voltage was applied and the output current was measured, while all gates were biased and the rest of the pads were left floating. Pulsed-spot measurement was used to program array and test flow-based computing. The custom Python GUI was designed to select and program/read individual devices in the array, and control these flow-based computing experiments.
Figure 5- 4: a) Electrical characterization setup with the SUSS Microtech probe station, the E5270A parametric analyzer, Keithley 707A switching matrix and their communication controlled by the computer. The probe card is installed onto the probe station in the highlighted area. b) The 2x12 Celadon Ultra High Performance probe card and device under test (DUT), probing all the 24 pads of our 8x8 1T1R HfO₂ array on the 300mm wafer. The inset shows the schematic and the transmission electron micrograph (TEM) of the 1T1R array element, ReRAM integrated with NFET at metal 1/via 1 interface, and the TiN/Ti/HfO₂/TiN ReRAM device stack composition.
Meanwhile, the flow-based computing for XOR logic consisted of a 2x2 nanoscale crossbar design, and hence only required a 2x2 subarray of the 8x8 array. This reduced the number of pads to be accessed, where only 4 probes were needed to perform this operation. The rest of the pads were left floating. Therefore, the flow-based computing was carried out using Agilent B1500 parametric analyzer, Waveform Generator /Fast Measurement Unit (WGFMU), Source Measurement Units (SMUs) and four probes. A custom Python GUI was designed to program and control these experiments.

5.3. Methodology

The flow-based computing was used to implement edge detection for computer vision and XOR Boolean logic, their methodologies are discussed in Sections 5.3.1 and 5.3.2., respectively.

5.3.1. Methodology of Flow-Based Edge Detection

5.3.1.1. Flow-Based Edge Detection Crossbar Design

The nanoscale flow-based edge detection design is shown in Figure 5-5, where the circles represent a non-volatile memory at the intersection of row and column nanowires. The edge detection design shown in Figure 5-5 was developed by our collaborator, a research group at the University of Texas at San Antonio (UTSA) for approximate edge detection between 8-bit pixels [20-21, 26]. An approximation of a ternary function is implemented on crossbar design via flow-based computing using a massively parallel simulated annealing search [20-21, 26]. The ternary function for approximate edge detection is learned from the BSDS500 dataset [29] and has 3 values. The approximate edge detection function maps pixel pairs to true when an edge is present, false when there is no edge and a third value that has an ambivalent result [20-21, 26]. A pixel has a gray-scale value ranging from 0 to 255 and is represented by eight bits. When referring to a pixel and its neighboring pixel in an image, the term “pixel-pair” is used. When the difference between
the pixels in the pixel pair exceed a pre-defined threshold value, an edge is said to exist. The function depends on thresholds, which are adjusted based on the calculated frequency of each unique pixel pair in the training image dataset from BSDS500, and hence the approximate edge detection function is selective in nature. The function evaluates to true based on the thresholds and conditions discussed in [20-21]. As such, pixel pairs that have rare occurrence would be ignored by the approximate edge detection algorithm even if the difference between the pixels exceed the threshold value (i.e. an edge is present). All pixel-pairs have been tested through simulation using the crossbar array designs for edge detection on all images. The 8x8 crossbar designs were used as mathematical functions for approximate edge detection to detect edges on all images. The peak signal-to-noise ratio (PSNR) of the ground truth images versus these images is 13.7 dB [21].

The flow-based edge detection design shown in Figure 5-5 dictates the resistance states of the non-volatile memory at each row/column intersection. Let the pixels in the pixel pair be denoted by pixel A and pixel B. As each pixel has eight binary bits, the input variables are bits A0-A7 for pixel A and bits B0-B7 for pixel B. The most significant bit and the least significant bit of pixels A and B are A0/B0 and A7/B7, respectively [26]. In Figure 5-5, the blue circles represent ReRAM devices that are mapped the binary value of the input variables A0-A7, B0-B7, using their literals or their negation symbolized by “!” preceding variable name [26]. The rest of the cells in the design are fixed at either logic 0 or logic 1, shown by gray and green circles, respectively. This is to reduce the number of devices to be programmed during computation, as their resistance states do not need to change with each new pixel pair [26]. Hence, the design in Figure 5-5 consists of Boolean logic 0 and logic 1 at each node, where the logic 0 and logic 1 are represented by high resistance state $R_{\text{off}}$ and low resistance state $R_{\text{on}}$, respectively. During flow-based computing, a device in the $R_{\text{on}}$ state promotes current flow between the row and column nanowires,
Figure 5-5: The 8x8 ReRAM crossbar design for flow-based edge detection, to detect edges between 8-bit pixels. Each circle represents one ReRAM at the intersection of row and column. The variables of edge detection consist of the bits of the two pixels, bits A0-A7 and B0-B7, of pixels A and B, respectively. The binary values of the bits are mapped to the ReRAM resistances, shown by blue circles, while other ReRAM are fixed at either logic 0 or logic 1, shown by gray and green circles respectively. The “!” represents the negation of the variables of pixels A and B. The logic 1 is mapped to Ron resistance state while the logic 0 is mapped to Roff resistance state. © 2021 IEEE, Reprinted, with permission.

while a device in the Roff state prohibits current flow between the row and column nanowires. This manipulates sneak path currents in the array during computation.

5.3.1.2. Methodology of Flow-Based Edge Detection Implementation

The methodology of flow-based computing is outlined in Figure 5-6. The Boolean value of each array element in the 8x8 array are determined by the edge detection design in Figure 5-5, as the design maps the binary values of the input variables to the corresponding NVM device in the array for flow-based computing. Then, the devices in the array are programmed according to the values of the eight-bit binary pixels. The mapping scheme will be discussed in detail on next
paragraph. The devices that were assigned permanent logic 0 and logic 1 may not need to be programmed again for new input variables. After programming is completed, the patterned resistance is read for verification using a read and verify scheme. Then, the flow-based computing is performed. An input signal is applied to a specific nanowire and the output signal is measured at the output nanowire of the 8x8 array. The applied input signal is a read voltage pulse or a current pulse, depending on whether the output of flow-based computing is to be measured in terms of measured current or measured voltage drop, respectively. In this work, we have used a read voltage of -100 mV for flow-based edge detection. The output of flow-based edge detection is a Boolean high (logic 1) or Boolean low (logic 0), where logic 1 represents existence of an edge and logic 0 represents absence of an edge. This information on whether the output of flow-based edge detection would be logic 1 or logic 0 for a given pixel pair, is based on the approximate ternary edge detection function described in Section 2.3.1.1. and is fixed a priori [26]. In the next and final step, the array is read again using read and verify scheme to ensure that the resistance states of the array devices were not disturbed during flow-based computing and that the output of flow-based computing is correct. This step is recommended for initial testing of flow-based edge detection, to verify that the applied input signal would not cause read disturb of the array devices. However, this final step is optional and may be skipped to reduce power consumption.

Figure 5-6: The flow chart of the steps required to implement the flow-based edge detection. © 2021 IEEE, Reprinted, with permission.
A hypothetical mapping scheme of the input variables, bits A0-A7 of pixel A and bits B0-B7 of pixel B, on the 8x8 array to detect an edge between A = 00001110 and B = 01001010, is illustrated in Figure 5-7 [26]. As mentioned earlier, the Boolean value of each ReRAM on the 8x8 array for this pixel pair is generated from the flow-based edge detection design shown in Figure 5-5. Based on this scheme, the ReRAM assigned logic 1 and logic 0 are programmed to R_{on} resistance states shown by green circles and R_{off} resistance states shown by gray circles, in Figure 5-7. For ReRAM devices assigned logic 1, the devices are SET into LRS (R_{on}) with a SET voltage of 2V and a current compliance of 245 µA. This LRS value was used because further increasing current compliance to 400 µA does not result in much lower LRS resistance and instead increases power consumption [26]. For ReRAM devices at logic 0, the devices are RESET into HRS (R_{off}) with a RESET voltage of -1.5V and current compliance of 100 µA. The HRS states were programmed with lower current compliance than 245 µA to yield higher RESET resistance [30]. Additionally, any higher resistance states than that of logic 1 (R_{on}) can be used for logic 0 (R_{off}) in flow-based computing. As such, other R_{off} states instead of HRS were also used for devices assigned logic 0. These devices were SET with a SET voltage of 2V at current compliances of 60 µA and 110 µA, hence are LRS states that are lower in resistance than HRS. The current compliance is provided by the integrated transistor on-chip for both SET, RESET and READ. Additionally, the Source Measurement Unit (SMU) measuring the current was also set to have current compliance of 1mA. After the array has been programmed, flow-based computing is performed by applying a read pulse to bottom row (input nanowire) and the output current is measured along last column (output nanowire). The red line shows one of the many possible sneak paths on the 8x8 array doing the computation. The flow-based output current can be measured as high or low current, which corresponds to low resistance output (logic 1) or high resistance output (logic 0), respectively.
low resistance output and a high resistance output correspond to presence or absence of an edge between a pixel-pair, respectively, according to approximate edge detection ternary function.

Figure 5-7: The mapping scheme of the variables of pixels A and B, namely bits A0-A7 and B0-B7, onto the ReRAM array. Each bit of pixel is a Boolean logic 1 or 0, and depending on the crossbar design, the respective ReRAM of that variable will either hold the input bit or its negation if the variable is preceded by “!”. The rest of the array remains fixed at logic 1 and logic 0. Based on this scheme, the ReRAM assigned logic 1 and logic 0 are programmed to R on resistance states (shown by green circles) and R off resistance states (shown by gray circles), respectively. Once the array has been programmed, flow-based computing is performed by applying a read pulse to bottom row (input nanowire) and the output current is observed along last column (output nanowire). The red line shows one of the many possible sneak paths on the 8x8 array doing the computation. © 2021 IEEE, Reprinted, with permission.

described in Section 2.3.1.1. Therefore, pixel-pairs that are predicted to have an edge or no edge, corresponding to True and False, respectively, of the ternary function, are tested in this work [26] and the results are discussed in next section.

5.3.2. Methodology of Flow-Based XOR Boolean Logic

The flow-based XOR design is shown below in Figure 5-8 (a) [19]. It is a 2x2 crossbar design that maps the two input variables of XOR, A and B, to the R on and R off resistance states of memristors in the array. The R on and R off resistance states correspond to logic 1 and logic 0, respectively, and are determined by the literals of input variables A and B, or their negations, A
and $\bar{B}$, respectively, in the XOR crossbar design. The crossbar design was synthesized using Free Binary Decision Diagram (FBDD) [19]. Details on the synthesis of this flow-based design can be found in [19].

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_8.png}
\caption{a) The 2x2 crossbar design for flow-based XOR consisting of literals and negations of the input variables of Boolean XOR logic. b) Methodology of implementing Boolean XOR logic on a 2x2 1T1R array. After the array is configured according to input variables of XOR, A and B, the sneak path currents computing the flow-based XOR are shown in red. The input read pulse is applied to top row and output current is measured along last row of the 2x2 array.}
\end{figure}

The schematic of the 2x2 1T1R sub-array used for implementing flow-based XOR is shown in Figure 5-8 (b). Once the devices are programmed, a read pulse is applied to the top row nanowire and the flow-based current is measured along bottom row nanowire. The red line shows the many possible sneak paths on the 2x2 array doing the flow-based XOR computation. The output current would be measured as either high or low current, corresponding to low resistance output when output is logic 1 and to high resistance output when output is logic 0, respectively. The input and output nanowires of the flow-based XOR compute were modified in this work, where the input was applied to top row instead of bottom row line, and the output was measured along bottom row.
instead of top row nanowire. Nevertheless, if the input row is swapped with the output row, the flow-based XOR would still work.

5.1. 5.4. Results and Discussion

5.4.1. XOR Boolean Logic Results and Discussion

The truth table of Boolean XOR logic is shown in Table 5-1. The expected flow-based result for each input configuration is also shown in Table 5-1. When the inputs A and B are not at the same logic, the XOR output is logic 1. Otherwise, the XOR output logic is 0. As mentioned earlier, the flow-based output is Boolean, where output logic 0 and output logic 1 are expected to have high resistance output and low resistance output, respectively.

The flow-based XOR was implemented on a 2x2 sub-array of the 8x8 1T1R HfO2 array. The experimental results of flow-based XOR for the four input configurations, 00, 11, 01 and 10, are reported in Table 5-2. For each input configuration of flow-based XOR, the logic states of each 1T1R are determined by the 2x2 crossbar design, where logic 1 and logic 0 are mapped to LRS and HRS, respectively, of each device. The LRS states were programmed with a SET voltage of 2V at current compliance of 60µA. The HRS states were programmed with a RESET voltage of -1.5V at current compliance of 60µA. The respective resistance states of each cell in the 2x2 array are reported in Table 5-2. The experimental results of flow-based XOR matches with the expected XOR logic output for all input configurations. For instance, when inputs A and B are at same logic, the measured flow-based current is low, giving a high resistance readout (logic 0 output). Similarly, when input configurations are 01 and 10, the measured flow-based current is high, giving a low resistance readout, which is logic 1 output.
Table 5-1: Truth table of Boolean XOR logic, and the expected output resistance for flow-based XOR for each input configuration, A and B. The output logic 1 and logic 0 correspond to low resistance output and high resistance output, respectively.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
<th>Expected Flow-based Resistance (Ω)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>High</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Low</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Low</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 5-2: The experimental results of flow-based XOR is reported. For each input configuration, the binary logic of each cell in the 2x2 1T1R array and their respective resistance states are shown, where logic 1 and logic 0 are mapped to low resistance state (LRS) and high resistance state (HRS) of the 1T1R, respectively. The experimental results for each input configuration of XOR confirms the expected logic outputs, where the flow-based XOR gave high resistance when XOR output is logic 0 and low resistance when the XOR output is logic 1.

<table>
<thead>
<tr>
<th>Input Configuration</th>
<th>A=0, B=0</th>
<th>A=1, B=1</th>
<th>A=0, B=1</th>
<th>A=1, B=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell</td>
<td>Logic R (kΩ)</td>
<td>Logic R (kΩ)</td>
<td>Logic R (kΩ)</td>
<td>Logic R (kΩ)</td>
</tr>
<tr>
<td>( \overline{B} )</td>
<td>1 10</td>
<td>0 46</td>
<td>0 56</td>
<td>1 9.02</td>
</tr>
<tr>
<td>( B )</td>
<td>0 120</td>
<td>1 8.3</td>
<td>1 8.2</td>
<td>0 45</td>
</tr>
<tr>
<td>( A )</td>
<td>0 300</td>
<td>1 11</td>
<td>0 160</td>
<td>1 9.57</td>
</tr>
<tr>
<td>( \overline{A} )</td>
<td>1 9</td>
<td>0 1200</td>
<td>1 8.9</td>
<td>0 1410</td>
</tr>
<tr>
<td>Output Logic</td>
<td>0 98</td>
<td>0 60</td>
<td>1 18</td>
<td>1 20</td>
</tr>
<tr>
<td>Expected Resistance</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
Figure 5-9: The resistance states of the 2x2 1T1R cells mapped according to flow-based crossbar design, and the corresponding flow-based XOR computation for input configurations a) A=0, B=0 b) A=1, B=1 c) A=0, B=1 and d) A=1, B=0. The 1T1R devices at LRS and HRS are shown by the white and green circles respectively, and the flow-based current is shown in red. The read pulse is applied to top row and flow-based current is measured along bottom row. The flow-based output current is expected to be low for (a)-(b) as R_off states in each sneak path lowers current measured at bottom row. In contrast, flow-based output current is expected to be high for (c)-(d) as the R_on states are present in the same sneak path, promoting current from row 1 to column 1, and column 1 to row 2, where output current is measured.

\[
\frac{1}{R_{flow}} = \frac{1}{R_B + R_A} + \frac{1}{R_B + R_A} \tag{1}
\]

Additionally, the flow-based current flowing for each input configuration of the flow-based XOR is shown in Figure 5-9. The 1T1R cells at LRS and HRS are shown by white and green circles, respectively, for input configurations 00, 11, 01 and 10, in Figures 5-9(a)-(d), respectively. The input read pulse is applied to top row and output current is measured along bottom row. The predicted flow-based current is shown in red arrows in Figures 5-9 (a)-(d). If a cell is at R_off, less current flows between row and column nanowires. Otherwise, a cell at R_on state facilitates more current flow between row and column nanowires. For the Figures 5-9 (a)-(b), the flow-based output is logic 0 as there is a HRS state in each of the sneak paths. For the Figures 5-9 (c)-(d), the flow-based output is logic 1 as there is a sneak path with 2 LRS cells, which is predicted to result in high flow-based current along bottom row, hence logic 1 output.
Table 5-3: The experimental and simulation results of flow-based XOR are reported for each input configuration. The experimental results are validated by the simulation of flow-based XOR.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Output</th>
<th>Expected Flow-based Resistance (Ω)</th>
<th>Experimental Results Resistance (kΩ)</th>
<th>Simulation Results Resistance (kΩ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>High</td>
<td>98</td>
<td>91</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Low</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Low</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>High</td>
<td>60</td>
<td>55</td>
</tr>
</tbody>
</table>

Due to the small crossbar design for flow-based XOR, the flow-based results can be validated using the equation (1), which shows the parallel circuit formed by the 2x2 crossbar design. Each variable in equation (1) represents the resistance states of the 1T1R cells from the experiment. The flow-based XOR output resistance $R_{\text{flow}}$ can be approximated using equation (1) because the resistance states are non-volatile and does not change during the flow-based computation. The XOR output resistance $R_{\text{flow}}$ has been calculated from the programmed resistance states in the experiment, using equation (1). The experimental and simulation results of $R_{\text{flow}}$ for each input configuration of the flow-based XOR are shown in Table 5-3, and the simulation results confirm the experimental results. Hence, the flow-based XOR was implemented successfully.

5.4.2. Edge Detection Results and Discussion

The flow-based edge detection was implemented on HfO$_2$ 8x8 1T1R arrays, where edges are detected between eight-bit pixels, pixels A and B, through flow-based computing using edge detection crossbar design in Figure 5-5. As pixel-pairs have been tested through simulation, a few pixel-pairs were selected for demonstration on the 8x8 1T1R arrays. Pixel-pairs that yielded true and false values for the approximate edge detection ternary function were tested.
Table 5-4: The values of pixel a and pixel b in each of the five patterns are shown below. The expected output logic of 1 corresponds to the presence of an edge between the 2 pixels and is expected to contribute to low resistance in the edge detection test. Similarly, the expected output logic of 0 implies the absence of an edge and therefore would result in high resistance in the output of edge detection test. The patterns consist of 3 low resistance patterns and 2 high resistance patterns. © 2021 IEEE, Reprinted, with permission.

<table>
<thead>
<tr>
<th>Edge Detection Pattern #</th>
<th>Pixel A</th>
<th>Pixel B</th>
<th>Expected Output Logic</th>
<th>Expected Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001110</td>
<td>01001010</td>
<td>1</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>00110101</td>
<td>01001010</td>
<td>1</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>10110111</td>
<td>01110000</td>
<td>1</td>
<td>Low</td>
</tr>
<tr>
<td>4</td>
<td>01101110</td>
<td>01111111</td>
<td>0</td>
<td>High</td>
</tr>
<tr>
<td>5</td>
<td>01010010</td>
<td>11101011</td>
<td>0</td>
<td>High</td>
</tr>
</tbody>
</table>

experimentally. The values of the pixel A and pixel B of the tested pixel-pairs are reported in Table 5-4 [26]. Each pixel-pair is given a pattern number for convenience and will be referred to by these pattern numbers in this section. Furthermore, ternary function outputs that are true pertain to existence of an edge between the pixel-pair, whereas those that are false represents the absence of an edge between the pixel-pair. The expected output logic in Table 5-4 represents the true and false values of the ternary function, as logic 1 and logic 0, respectively. In Table 5-4, the first three pixel-pairs are expected to have an edge based on the ternary function, and hence would give logic 1 (low resistance output) as the flow-based edge detection output. Meanwhile, the last two pixel-pairs in Table 5-4 are expected not to have an edge based on ternary function, and hence are expected to give logic 0, a high resistance output, for flow-based edge detection.

The five pixel-pairs in Table 5-4, are implemented experimentally on the 8x8 1T1R HfO$_2$ arrays using methodology discussed in Section 5.3.1.2. The experimental results of the flow-based edge detection for the five patterns, denoted by E.D. Pattern numbers 1 through 5, are shown in Figure 5-10. The flow-based output currents were measured and converted to their respective flow-
Figure 5-10: Comparison of experimental edge detection outputs of 5 pixel-pairs (each pattern denotes a pixel-pair) with their ideal case LTSpice simulation using HRS and LRS patterned arrays. The result confirms that an edge is detected in the first three pixel pairs, and no edge is present between the last 2 pixel-pairs. The results were validated by LTSpice simulation assuming HRS of 100 kΩ and LRS of 3.5 kΩ, with no resistance variation. © 2021 IEEE, Reprinted, with permission. © 2021 IEEE, Reprinted, with permission.

Based output resistance. The flow-based output resistance is measured to be low for the first three pixel-pairs as an edge is expected between them, giving an output of logic 1. On the contrary, the flow-based output resistance is measured as high when the pixel-pairs are predicted not to have an edge between them, giving an output of logic 0. The ratio between the empirical flow-based logic 0 and logic 1 outputs is 3:1, and hence the outputs are distinguishable. As the 1T1R array is configured to hold the input bits of the pixel-pair tested for flow-based computing, the LRS state and the HRS state were used for the patterned $R_{on}$ and $R_{off}$ resistance states, giving a patterned $R_{off}/R_{on}$ ratio of 28.6:1. In order to validate the flow-based edge detection results, 8x8 arrays were simulated in LTSpice with the patterned mean LRS resistance of 3.5kΩ and the patterned mean HRS resistance of 100kΩ. No resistance variation was assumed for the simulation. The simulation was performed using resistance states of the patterned arrays as the resistance state of devices do not change during the flow-based computation. The LTSpice simulation results validated the
experimental results of flow-based edge detection for E.D. Pattern 1- E.D. Pattern 5, as shown in Figure 5-10. The difference between the experimental and simulated flow-based resistance output of E.D. Pattern 5 may be attributed to the inherent variability of ReRAM devices [26].

Next, the effect of patterned ReRAM resistance variability on flow-based edge detection was investigated using the five patterns, E.D. Pattern 1 to E.D. Pattern 5. In the works reported on flow-based edge detection, the ReRAM elements in the nanoscale crossbar designs are assumed to have ideal resistances, with no variability and are considered as uniform [20, 24-25]. However, due to the stochastic nature of defects in the conductive filament of ReRAM, ReRAM devices have intrinsic variability in their resistance states, such as device-to-device and cycle-to-cycle variation. Hence, it is important to investigate the effect of ReRAM variability on the flow-based edge detection.

To assess the impact of ReRAM variability, the 8x8 1T1R arrays were patterned with different $R_{\text{off}}$ resistance states with different resistance variability, for each of the five pixel-pairs in Table 5-4 and the flow-based computing was performed. The three different $R_{\text{off}}$ states used were HRS with a mean of 100 kΩ and standard deviation of 0.344, and two LRS states, with mean resistances of 9 kΩ and 5.6 kΩ and standard deviations of 0.18 and 0.1, respectively. The average $R_{\text{on}}$ resistance state for logic 1 was 3.5 kΩ, with standard deviation of 0.08. These matrices yielded three patterned $R_{\text{off}}/R_{\text{on}}$ resistance ratios, 28.6:1, 2.5:1 and 1.5:1, respectively, which were investigated on the five pixel-pairs from Table 5-4. Therefore, the 8x8 1T1R arrays were patterned with these matrices and results of flow-based edge detection on the five pixel-pairs are shown in
The effect of three patterned R$_{off}$/R$_{on}$ resistance ratios, 1.5:1, 2.5:1 and 28.6:1, shown left to right, on the five flow-based edge detection outputs for over 50 cycles. The five flow-based edge detection outputs consist of three low outputs (E.D. Patterns 1-3) and two high outputs (E.D Patterns 4-5). With increasing patterned R$_{off}$ state, the ratio between the flow-based binary outputs improves by three-fold from 1.16:1 to 3.08:1, when HRS is used for patterned logic 0 resistance, as opposed to LRS states of 5.6 kΩ and 9 kΩ used for R$_{off}$ state. Also, the variability of flow-based edge detection outputs increases with increasing variability of patterned R$_{off}$ State (logic 0). Hence, there is a trade-off between the binary flow-based output ratio and the variability of the flow-based outputs. One-way ANOVA shows significant difference between the high and low flow-based edge detection outputs for all three patterned R$_{off}$/R$_{on}$ resistance ratios, and hence flow-based edge detection worked for the three conditions. © 2021 IEEE, Reprinted, with permission.

Figure 5-11: An endurance of over 50 cycles was carried out on the arrays for these five pixel-pairs to assess the impact of ReRAM variability of patterned R$_{off}$/R$_{on}$ resistance states 28.6:1, 2.5:1 and 1.5:1, shown left to right in Figure 5-11. There are two important observations.

The results for flow-based edge detection are predicted to have low resistance (logic 1 output) for E.D. Patterns 1-3 and high resistance (logic 0 output) for E.D. Patterns 4-5, due to presence and absence of an edge, respectively, based on ternary function. In Figure 5-11, the difference between the binary flow-based outputs increased with increasing patterned R$_{off}$/R$_{on}$ resistance ratios (shown left to right), where it is greatest when HRS is used for logic 0 (patterned R$_{off}$/R$_{on}$ 28.6:1). The ratio of high to low flow-based edge detection outputs improved from 1.16:1...
to 1.44:1 for patterned $R_{\text{off}}/R_{\text{on}}$ ratios 1.5:1 to 2.5:1 respectively, then to 3.08:1 for patterned $R_{\text{off}}/R_{\text{on}}$ ratio 28.6:1 [26]. Hence, the ratio between the high and low binary outputs improved by three-fold when highest patterned $R_{\text{off}}/R_{\text{on}}$ ratio was used. This shows that higher patterned $R_{\text{off}}$ results in higher flow-based logic 0 outputs, as shown by pixel-pairs for E.D. 4-5. This gives better ratio between the binary flow-based outputs. This may be due to the decreased resistance in each of multiple sneak paths caused by lower $R_{\text{off}}$ states, that resulted in overall resistance drop of flow-based logic 0 outputs [26].

Secondly, in Figure 5-11, the effect of ReRAM variability of the three patterned $R_{\text{off}}/R_{\text{on}}$ ratios, 1.5:1, 2.5:1 and 28.6:1, on the five pixel-pairs are reflected in the box and whisker plots from over 50 cycles [26]. The data shows that increased variability of $R_{\text{off}}$ state resulted in increased variability in the flow-based edge detection outputs, particularly for E.D. 4-5 when 28.6:1 patterned $R_{\text{off}}/R_{\text{on}}$ ratio was used [26]. As discussed previously, HRS states exhibit much greater variability than LRS states due to different conduction mechanisms involved during the switching process [31]. The lower variability in the LRS states gave less variation in the five flow-based outputs, E.D.1-E.D.5, when 5.6 kΩ and 9 kΩ were used for $R_{\text{off}}$ state instead of 100kΩ (HRS). The increased variability of flow-based outputs was attributed to $R_{\text{off}}$ state variability because $R_{\text{on}}$ state was maintained for all the patterned $R_{\text{off}}/R_{\text{on}}$ ratios.

Therefore, the effect of ReRAM variability due to the three patterned $R_{\text{off}}/R_{\text{on}}$ ratios, 28.6:1, 2.5:1 and 1.5:1, reveal that there is a trade-off between the binary flow-based output ratio and the variability of the flow-based outputs. Ideally, it is better to have high binary flow-based output ratio and low variability of the flow-based outputs. From these experiments, the highest patterned $R_{\text{off}}/R_{\text{on}}$ ratio, 28.6:1, gave the highest ratio for the binary flow-based outputs and the greatest variability in the flow-based edge detection outputs. Nevertheless, an ordinary one-way ANOVA
analysis on all five pixel-pairs for each patterned resistance ratio showed that there is significant difference between low (E.D. 1-3) and high (E.D. 4-5) flow-based edge detection outputs. Hence, the binary flow-based outputs are distinguishable for all the three patterned \( R_{off}/R_{on} \) ratios [26], and this shows that flow-based computing can accommodate \( R_{off}/R_{on} \) ratios less than 10.

**Figure 5-12:** The LTSpice Gaussian simulation results for effect of three patterned \( R_{off}/R_{on} \) resistance ratios, 1.5:1, 2.5:1 and 28.6:1, shown left to right, on the five flow-based edge detection outputs for over 200 cycles. The five flow-based edge detection outputs consist of three low outputs (E.D. Patterns 1-3) and two high outputs (E.D Patterns 4-5). With increasing patterned \( R_{off} \) state, the high flow-based outputs increase, giving higher ratio between the binary flow-based outputs. Also, the variability of flow-based outputs increase with variability of patterned resistance. Hence, a trade-off exists between the binary flow-based output ratio and the flow-based outputs, and simulation conform to experimental results. © 2021 IEEE, Reprinted, with permission.

To validate the experimental results of the effect of three patterned \( R_{off}/R_{on} \) resistance ratios on flow-based edge detection, the five pixel-pairs (E.D. Patterns 1-5) were simulated on 8x8 arrays in LTSpice software. A Gaussian distribution was assumed for each of the \( R_{on} \) and \( R_{off} \) states, with their respective mean resistances and standard deviations. The simulation was run for 200 cycles. The simulation includes up to three standard deviations. The simulation results for the three patterned \( R_{off}/R_{on} \) ratios, 28.6:1, 2.5:1 and 1.5:1, on the five pixel-pairs are shown in Figure 5-12 and conform to the empirical results [26]. Therefore, the effect of ReRAM variability in simulation...
for the three patterned $R_{\text{off}}/R_{\text{on}}$ ratios also shows that there is a trade-off between the binary flow-based output ratio and the variability of the flow-based edge detection outputs. In other words, the flow-based output ratio improves with increasing patterned $R_{\text{off}}$, however, the variability of the outputs increases too.

To compare the experimental results with simulation, a Pearson correlation coefficient was extracted for all three patterned resistance ratios for all five edge detection patterns, as shown in Figure 5-13 [26]. The x-axis represents the mean resistance of the experimental data over 50 cycles, while the y-axis shows the mean resistance of simulated data for 200 cycles. The experimental data has a strong positive correlation with the simulation, as shown by extracted correlation coefficient $r$ of 0.9547 in Figure 5-13 [26]. With higher $R_{\text{off}}$ states, the flow-based logic 0 outputs increase significantly along with its variability, resulting in the greatest variability for patterns E.D. 4-5 when HRS is used as $R_{\text{off}}$ both in experiment and simulation. The variability of these patterns are higher than low flow-based outputs (E.D. 1-3) patterned with the same $R_{\text{off}}/R_{\text{on}}$ of 28.6:1, and all five pixel-pairs (E.D. 1-5) patterned with lower $R_{\text{off}}/R_{\text{on}}$ resistances of 1.5:1 and 2.5:1. This resulted in these two means drifting from the simple linear regression line in Figure 5-13, as opposed to rest of data. The possible causes are discussed in next paragraph.

The simulation results show higher flow-based outputs than empirical data. This may be due to ideal conditions assumed during simulation where no read disturb of higher resistance states could occur. HRS regimes normally have read noise [32], which could have contributed to the experimental data. Additionally, it is possible for HRS states to be reduced partially by a few k$\Omega$ of resistance (still in the HRS regime) during flow-based computing due to sneak path currents, while in simulation the HRS states are maintained during the computation [26]. These factors may
have caused the experimental results for E.D. 4-5 to differ from the simulation results when HRS is used as \( R_{\text{off}} \) state in the patterns.

![Pearson correlation coefficient graph](image)

**Figure 5-13**: The Pearson correlation coefficient of the experimental edge detection outputs for 5 patterns and their LTSpice Gaussian simulation for the three patterned \( R_{\text{off}}/R_{\text{on}} \) ratios of 1.5:1, 2.5:1 and 28.6:1. The coefficient \( r \) of 0.9547 shows a strong positive correlation between the experimental data and the simulation. © 2021 IEEE, Reprinted, with permission.

5.5. Conclusion

The data transfer between physically separated memory and processing units in von-Neumann architecture poses severe limits to data bandwidth and increases latency and power consumption for data-intensive applications and IoT. Flow-based computing is one possible solution to von-Neumann bottleneck, where data is computed upon in the memory of emerging non-volatile memory. Bipolar ReRAM is a strong candidate for this application, due to its low power consumption, high switching speed, simple structure, high density, high scalability, good compatibility with CMOS and ease of fabrication. In this work, flow-based computing has been evaluated on fabricated HfO\(_2\) 1T1R arrays for edge detection between eight-bit pixels and XOR Boolean logic. The multi-level resistance states of 1T1R array elements evaluated in previous chapter have been used to implement flow-based computing on the memory arrays. Flow-based computing controls flow of sneak path currents in reconfigurable passive arrays to give Boolean
outputs, logic 0 for high resistance output and logic 1 for low resistance output. The input data is stored in the memory array and is computed upon in flow-based computing. For flow-based XOR, 2x2 sub-arrays were configured according to XOR inputs A and B, where LRS and HRS were used for patterning logic 0 and logic 1 inside the array. Flow-based computing was performed for all XOR input configurations, and the experimental results conform to XOR truth table. The flow-based logic 0 outputs were 98 kΩ and 60 kΩ for inputs 00 and 11 respectively. The flow-based logic 1 outputs were 18 kΩ and 20 kΩ for inputs 01 and 10 respectively. The flow-based XOR experimental results were verified by LTSpice simulation, and results are in close agreement.

Flow-based computing was implemented for approximate edge detection based on ternary function on 8x8 arrays, where an edge is expected between three pixel-pairs, yielding low resistance output (logic 1) and no edge expected between two pixel-pairs, predicted to give high resistance output (logic 0) for flow-based computing. These are the binary outputs of flow-based edge detection, and their output ratio is logic 0/logic 1. As the arrays are configured to hold the eight-bit pixels as inputs, different R_{off} states (HRS, 2 LRS states of 9kΩ and 5.6 kΩ) with same R_{on} state (3.5 kΩ) were programmed into the arrays. Therefore, these five pixel-pairs were evaluated on 8x8 1T1R arrays, using three different patterned R_{off}/R_{on} resistance ratios, 1.5:1, 2.5:1 and 28.6:1, to investigate effect of ReRAM resistance state variability on flow-based computing. The experiments were repeated for over 50 cycles. For lower patterned R_{off}/R_{on} ratios of 1.5:1 and 2.5:1, the variability of flow-based outputs was low but their binary flow-based output ratios were 1.16:1 and 1.44:1 respectively. Meanwhile, for patterned R_{off}/R_{on} ratio of 28.6:1, the binary flow-based output ratios were three-fold, 3.08:1, but the variability of flow-based logic 0 outputs were high. This is attributed to the greater resistance variation present in HRS states being used as R_{off} state, rather than the R_{off} states patterned with LRS states. An ordinary one-way ANOVA showed
there is still significant difference between the high and low flow-based outputs for all patterned $R_{off}/R_{on}$ ratios, 1.5:1, 2.5:1 and 28.6:1, and therefore the binary flow-based outputs are distinguishable for all three conditions. Hence, an edge is detected for first three pixel-pairs and no edge detected for last two pixel-pairs, for all three patterned $R_{off}/R_{on}$ ratios. The experimental results were also verified with LTSpice Gaussian simulation using the means and standard deviations of patterned $R_{on}$ state and $R_{off}$ state. The experimental data shows a strong positive correlation with the simulation data, with a Pearson correlation coefficient $r$ of 0.9547. Hence, the patterned $R_{off}/R_{on}$ resistance ratio and variability in ReRAM resistance states play a role on flow-based computing and give a trade-off between the flow-based edge detection output ratio and variability of flow-based outputs.

5.6. Comparison to Related Works

5.6.1. XOR Boolean Logic

Logic operations have been proposed on NVM devices using either resistance of devices or voltage pulses applied to devices, or a combination of both, by few different methods as a potential in-memory computation. There are various methods reported in the literature for performing logic functions on Resistive Random Access Memory (ReRAM). ReRAM has been regarded as a very promising non-volatile memory, with the advantages of low power consumption, high density, high scalability, good endurance (greater than $10^9$ cycles) and high switching speed. This makes it very attractive for logic functions. For instance, Borghetti et al. also used 3 binary resistive switches and 7 sequential steps to realize NAND operation using the IMPLY and FALSE (output is always logic 0) [12]. The XOR is one of the fundamental logic operation extensively used
Table 5-5: Comparison of proposed flow-based XOR to other approaches using ReRAM array, in terms of number of ReRAM cells used and total number of sequential steps in the XOR operation.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Memristors (excluding transistors)</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref. [14]</td>
<td>4</td>
<td>6 (7 programming steps reduced to 5 due to parallel programming, 1 read step)</td>
</tr>
<tr>
<td>Ref. [34]</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>Ref. [35]</td>
<td>8</td>
<td>10 (8 programming steps and 2 read steps)</td>
</tr>
<tr>
<td>Ref. [36]</td>
<td>6</td>
<td>1 (6 programming steps and one sensed output voltage in 1 step)</td>
</tr>
<tr>
<td><strong>This Work</strong></td>
<td>4</td>
<td>5 (4 programming steps and 1 read step)</td>
</tr>
<tr>
<td>Ref. [15]</td>
<td>2</td>
<td>3 (2 programming steps and 1 read)</td>
</tr>
<tr>
<td>Ref. [40]</td>
<td>1</td>
<td>3 (initialization, writing and read steps)</td>
</tr>
</tbody>
</table>

in error-correcting codes, digital adders and comparators [33], and the different methods of XOR based ReRAM are discussed next.

Material implication (IMPLY or IMP) involves the conditional switching of ReRAM and is based on two variables P and Q, where the logical states of P and Q are stored in terms of the resistance states of two memristors. The two memristors share a common horizontal wire, which is connected to a load resistor R_G. The memristors P and Q are applied voltages V_{conditional} and V_{set}, respectively, during IMP operation. The value of memristor Q depends conditionally on the states of memristors P and Q, giving the “pIMPlpmpq” according to truth table of (NOTp)ORq [12, 34]. FALSE operation refers to resetting the memristor to HRS. The XOR operation for memristor-based IMPLY [34] required 5 memristors (2 input memristors, 1 output memristor and 2 functional memristors) and 13 sequential steps. In addition, IMPLY is destructive to the states of memristors
P and Q, so when inputs are assigned to input memristors, the values also need to be copied to designated memristor [34]. Although 1R crossbars are used in [34], transistors are required at each row and column to select the desired row and column of accessed cell, and a resistor $R_G$ is needed. Although IMPLY and FALSE forms a computationally complete logic set of 16 Boolean functions, the number of sequential steps performed is high based on operation, particularly for XOR Boolean logic.

In [14], a combination of applied voltage pulses and resistive states are used for inputs and computation in complementary resistive switching (CRS) structures in the array. Typically CRS consists of anti-serially connected memristors, where the two memristors share a common electrode. Their method implemented XOR logic on 4 ReRAM cells connected in CRS configuration, sharing a common bottom electrode in the array, in 6 sequential steps. It uses a sequence of NAND operations performed on each pair of CRS cells, where the output is stored as the resistance state of the second cell of the CRS cells, in each pair. Each NAND requires initializing the states of the CRS cells according to one NAND input, and the other NAND input decides what voltage pulses will be applied to the CRS cells. The two CRS cells are programmed simultaneously in one step, reducing the 2 programming steps to 1 step, and the NAND computation step may either result in programming one or both cells again, or no change in stored states [14]. This method therefore imposes design constraints on the threshold voltages of CRS cells. The NAND operation may take up to two steps, which would result in 7 programming steps and 1 read cycle for XOR operation. This would give 8 steps in total, however, as programming of each pair of CRS cells were done in parallel (programming 4 cells in 2 steps), the sequential steps for CRS based XOR reduces to 6 steps. As the number of programmed devices are higher than 4, this approach also consumes more energy than the flow-based XOR.
In [35], eight 2T1R cells performed XOR operation, using a differential read scheme to evaluate the XOR output. It was operated with extra peripheral sensing circuits such as a pre-charge sense amplifier (PCSA) and a MOS layer that connects and disconnects the array from the PCSA [35]. The method requires programming eight ReRAM cells to hold the inputs A and B of XOR. Then, the XOR operation is executed using a two-stage read. Hence, [35] has 10 steps (8 programming steps and 2 stage-read steps) to compute XOR. This method requires higher number of memristors and steps to compute XOR compared to flow-based XOR.

Another approach used a combination of 6 memristors and a pair of CMOS transistor inverters to compute Boolean logic, called memristor ratioed logic (MRL) [36]. The concept of MRL uses 2 memristors in a voltage divider configuration, where the output voltage is sensed at the shared electrode of the 2 memristors, yielding voltage across one of the memristors like a voltage divider circuit. The input voltages are applied to other end of the memristors. Depending on which polarity of the pair of memristors will receive the applied voltages, MRL AND and MRL OR gates are proposed [36]. Then, two MRL AND gates and one MRL OR gate are constructed with two pairs of CMOS inverters to give MRL XOR. In [36], a MRL XOR is proposed using a total of 6 memristors and 4 transistors. The 6 memristors are switched in parallel in one computation step, and the output voltage is also sensed in the same step, with input voltage of 1.1V. However, this method requires design constraints on the threshold of memristors to work, a high $R_{off}/R_{on}$ ratio, and is not crossbar compatible due to the extra CMOS inverters needed. In addition to this, it requires switching 6 memristors in parallel, so this would consume higher energy than our method, where 4 memristors are programmed.

Our work used 4 1T1R cells, and one read step for XOR computation. Since the inputs A and B are programmed into resistance states of the 4 memristor cells, our method requires 5 steps (4
programming steps and 1 read step). The number of memristors and the total number of sequential steps is summarized in Table 5-5, and outlines that our approach used lower number of ReRAM cells and sequential steps than other ReRAM-based XOR in-memory computations discussed in [14, 34-36]. It is possible to tune multiple cells in parallel for faster array programming [37], which would reduce the number of steps required for our method. Furthermore, if only one of the inputs, either A or B, is changed then only 2 ReRAM cells need to be programmed. This would reduce the number of sequential steps to 3 steps. In addition to this, our method can also be implemented in 1R crossbar arrays, as shown in [22] for 1-bit full adder. This would reduce the area consumption of our method. The advantages of our flow-based approach include no destructive readouts of the inputs as they are stored in-memory; no conditional switching of devices as in IMPLY (and FALSE) [34], XOR based on CRS cells [14], MAGIC [38] and MRL XOR [36], relaxing design constraints on threshold voltages and $R_{\text{off}}/R_{\text{on}}$ ratios, in addition to being crossbar-compatible. Although our approach exceeds approaches discussed in [14, 34-36] in terms of number of memristors and sequential steps used, there are a few approaches that showed XOR operation with fewer memristors and number of steps such as [15, 40].

In [15], a pair of memristors connected in CRS configuration is used. One input of XOR is programmed into the binary states of the CRS cells, and the other input dictates to which of two cells will be applied the read voltage. The output voltage is sensed at the shared electrode of the CRS. This method therefore used 2 memristors and 3 steps composed of 2 programming steps and 1 read step. However, the approach is dependent on high $R_{\text{off}}/R_{\text{on}}$ ratio, where a high ratio of 100 was used. Thus, it may be difficult for ReRAM devices with $R_{\text{off}}/R_{\text{on}}$ ratios less than 10. In addition, the destructive readout of CRS cell may need extra initialization cycles [39]. The other approach in [40] implemented XOR operation on a single 1T1R structure, in 3 sequential steps. The inputs
of XOR are assigned to the initialized state of ReRAM and the voltages applied to gate terminal, top and bottom terminals of the device according to the logic performed. The output is stored in the final state of ReRAM. The three steps are initialization step of ReRAM resistance state, the writing step and the final read step to read the resistance of ReRAM. This approach outperformed all the other approaches in terms of number of devices and sequential steps, as shown in Table 5-5. However, this approach may not be possible to implement on 1R crossbars.

5.6.2. Approximate Edge Detection

In the literature, image processing has been carried out using digital logic, mixed-signal processing circuits [41], FPGA implementations [42], ReRAM-based Convolutional Neural Networks (CNNs) [43-44]. Examples of image processing include image dilation by digital ReRAM-Based Convolutional Block consisting of binary 4T2R arrays, XNOR circuits and bit counters [44]; image orientation selectivity via neuromorphic system composed of ReRAM and neuronal circuits [45]; image enhancement on ReRAM crossbar through anisotropic diffusion algorithm [42]. Due to in-memory computation capability of ReRAM crossbars, memory access has been reduced by 64% compared to digital implementations in [42]. Edge detection is an important application of image processing, and has also been demonstrated with ReRAM [46-48]. For instance, Mannion et al. simulated edge detection by mapping pixel intensities to a range of measured frequencies, in a simulation model based on empirical data that detected differences in frequencies through a pair of ReRAM in voltage divider configuration [46]. In addition, Pajouhi et al. simulated edge detection through ant colony optimization algorithms [47]. Li et al. demonstrated different convolutional image filters to detect edges through vector-matrix multiplication (also known as dot product operation) on a large crossbar ReRAM array, which is a 1T1R array [48].
5.7. Acknowledgment

The authors acknowledge the CNSE fabrication facility for CMOS integrated ReRAM array fabrication on 300 mm wafer platform; Professor Sumit K. Jha and Jodh Pannu from University of Texas at San Antonio (UTSA) for the generation of compute kernels for approximate edge detection and XOR Boolean logic; Dr. Karsten Beckmann (CNSE Integration Engineer) for his helpful discussion on this project. This research work is sponsored by the National Science Foundation (NSF) under agreement number 1823015.

5.8. References


Chapter 6: Conclusion and Future Outlook

6.1. Conclusion of Dissertation

Resistive Random Access Memory (ReRAM) is a promising non-volatile memory that has low power consumption, high switching speed, simple structure, high density, good scalability and good compatibility with CMOS. Hafnium-oxide ReRAM integrated with 65nm CMOS technology on a 300 mm wafer platform has been utilized to carry out two novel non-von Neumann computing applications. Arrays of integrated HfO$_2$ 1T1R devices have been characterized for 10,000 binary switching cycles using current compliance control of integrated transistors, via gate voltage modulation, to obtain multi-level resistance states in LRS regime (Chapter 3). Using current compliances from 60µA to 400µA, a five-fold change in resistance from 15 kΩ to 2.8 kΩ, was achieved across arrays on multiple die across the wafer, while suppressing the standard deviation from 2 kΩ to less than 400 Ω. The resistance levels were assessed for intra-array, inter-arrays on the same die, and die-to-die, and were found to be uniform. This method along with read verify approach successfully programmed 8x8 arrays into different resistance levels, as demonstrated by writing images into these 8x8 arrays. This programming approach was used later in Chapter 5 to assess the impact of $R_{\text{off}}/R_{\text{on}}$ resistance ratios and the effect of variability of these resistance states on the non-von Neumann computing application. The distinct multi-level resistance states of emerging non-volatile memory, enabled by the appropriate selection of current compliance, paves the way to implement ReRAM in non-von Neumann applications that require binary or even ternary state logic and memory.

In Chapter 4, a novel in-memory computing application, which was demonstrated earlier with Phase Change Memory (PCM) [1], was evaluated on ReRAM. The application uses memory arrays and a unique property, the non-volatile accumulative behavior of NVM devices, where incremental
change in resistance is obtained through application of ultra-short pulses. In this in-memory application, the degree of association between discrete-time binary processes, called correlation, can be obtained in parallel through the non-volatile memory array through absolute change of device conductance, and the result of this computation is stored directly in memory array. To check the feasibility of temporal correlation detection on ReRAM, an empirical ReRAM model was simulated with the algorithm in a Python software environment. It detected correlated binary processes from the uncorrelated ones successfully using the SET regime of ReRAM. Next, due to equipment limitations, experimental analog ReRAM data was used to successfully detect ten correlated binary processes from 25 binary processes using a modified correlation detection algorithm in Python environment. In the experimental analog data, the resistance was modulated with 200 successive pulses of 300 picoseconds pulse width of SET / RESET voltages of 1V / -0.75V respectively, and four cycles of SET and RESET were evaluated with the algorithm. The algorithm modulated the number of pulses to be applied to ReRAM devices in accordance to number of binary processes that have a ‘1’ at each time instance. The results showed that the gradual resistance modulation in RESET analog data gave better performance than the abrupt SET analog data, where the absolute changes in device conductance for all four cycles of RESET data detected 100% of correlated processes present. Meanwhile, only two of the four SET cycles detected 100% of correlated processes, while the other two cycles detected 90% and 50% of correlated processes. The results also showed that the algorithm failed to work during regions of data with no resistance modulation and when the resistance change was not monotonic, which could be due to ReRAM variability. Due to the low power requirements of ReRAM, the implementation of this non-von Neumann application consumed 36,000-53,000 times lower energy than implementation with PCM, sped-up the execution time by 1,600-2,100 times than that
of 1xPOWER8 CPU [1 thread], for the same number of processes. This work demonstrates the promising low power potential of ReRAM for high-level computation via in-memory computing, where the von-Neumann bottleneck can be circumvented by co-locating memory and processing capabilities of ReRAM non-volatile memory.

Chapter 5 presents another non-von Neumann computing application, using sneak path currents in a memory array to perform logic, called flow-based computing. The flow-based computing was experimentally demonstrated on hafnium-oxide ReRAM arrays for approximate edge detection and XOR Boolean logic. The effect of patterned resistance states $R_{off}/R_{on}$ ratios and the variability of the states on flow-based computing outputs were evaluated for an endurance of 50 cycles. The ratio between the binary flow-based computing outputs, measured as high / low resistance outputs when the binary output is logic 0 / 1, is shown to improve with increasing patterned $R_{off}/R_{on}$ resistance ratios, with the highest $R_{off}/R_{on}$ ratio of 28.6:1 giving a flow-based output ratio of 3. In addition, one-way ANOVA between the binary outputs of approximate flow-based edge detection for all patterned $R_{off}/R_{on}$ ratios (1.5:1, 2.5:1, 28.6:1) tested showed there is significant difference between the low and high binary outputs, even for lower $R_{off}/R_{on}$ ratios. A trade-off was found between the approximate flow-based edge detection output ratio and the variability of the flow-based outputs. Increase in variability of patterned ReRAM resistance state increased variability of the flow-based outputs. Approximate flow-based computing with lower $R_{off}/R_{on}$ ratios have less variability in their outputs, but lower output ratio. This may be suitable for systems with high sensitivity where more consistent binary outputs are needed. For large $R_{off}/R_{on}$ ratios and higher variability of binary outputs, it could benefit more robust systems as these would need better distinction between the binary outputs, and would be able to tolerate loss of accuracy due to variability of these outputs.
Lastly, flow-based XOR Boolean logic was implemented on hafnium-oxide ReRAM arrays, and the results of the computation for all XOR inputs (00,01,10,11) are in good agreement with the XOR truth table. A SPICE simulation was also carried out to validate the results for both approximate edge detection and XOR Boolean logic. A Gaussian simulation was used to simulate effect of ReRAM variability in patterned resistance states on approximate flow-based edge detection, and the simulation results agree with experimental results with a Pearson correlation coefficient $r$ of 0.9547. The flow-based XOR outperformed many of the other XOR in-memory approaches in the literature, in terms of the number of memristors used and the number of sequential steps in the XOR computation. The advantages of our flow-based approach include no destructive readouts of the inputs as they are stored in-memory; no conditional switching of devices during computation as in IMPLY (and FALSE) [2], XOR based on CRS cells [3], MAGIC [4] and MRL XOR [5], relaxing design constraints on threshold voltages and $R_{\text{on}}/R_{\text{off}}$ ratios, in addition to being crossbar-compatible. If separate sub-arrays are pre-programmed to store XOR inputs for each of the four input configurations, the flow-based computing can be computed multiple times without device programming, by either measuring current or voltage drop, without the need to switch devices, relaxing the effect on ReRAM endurance. This would also consume lower power. To avoid data movement, these computations were carried out inside of memory. Therefore, flow-based computing is attractive as a non-von Neumann computing approach as different logic functions can be implemented on dense crossbar arrays of ReRAM.

6.2. Future Outlooks

In-memory computing approaches based on ReRAM emerging non-volatile memory are promising given the fast switching speed, high density, low power, high scalability, good CMOS compatibility with respect to current memory technologies (i.e. higher density than SRAM, faster
access speed than NAND flash). There are still some challenges in the integration of in-memory computing for use on larger scale, which are discussed below.

There are a few future directions with regards to temporal correlation detection with ReRAM. The performance of this algorithm may be improved by attaining a higher memory window ($R_{\text{off}}/R_{\text{on}}$) in the analog ReRAM incremental resistance, which is a challenge. A different metal-oxide ReRAM with higher $R_{\text{off}}$ state, such as tantalum oxide ReRAM may be used, which would give more intermediate resistance states. Optimization of processing conditions on device/array level, as well as intelligent programming algorithms to lower resistance variability of HfO$_2$ ReRAM is needed to ensure a more monotonic resistance modulation in either RESET or SET regimes, this would be helpful to improve performance of this algorithm with ReRAM. Another method that could improve the range of conductance would be to assign binary process to multiple devices instead of one device, as was done previously with PCM, but this method remains to be explored for ReRAM non-volatile memory. Lastly, the modulation of pulse width of ReRAM remains to be explored with the initial algorithm of temporal correlation detection.

There are many future directions of flow-based computing as a non-von Neumann approach. The performance of flow-based computing could immensely benefit from increased $R_{\text{off}}$ states used for logic 0, to give higher binary output ratios for the flow-based outputs. For instance, changing the switching oxide of ReRAM with tantalum oxide may give higher memory window than that of hafnium oxide ReRAM, which would help increase the binary flow-based output ratio. Secondly, improvements of the processing conditions are needed to reduce stochasticity of the ReRAM resistance states, especially for HRS states. This would help to suppress the variability of flow-based outputs. For future work on approximate flow-based edge detection / computing, development of circuit architecture is needed for real-time processing. This could consist of

164
multiple 8x8 array modules, with shared DACs and ADCs, and circuitry to measure the flow-based outputs. For instance, at the output nodes of flow-based computing, a multiplexer may be used to interface output node to either trans-impedance amplifiers (TIAs) or ADCs. ADCs at output node would measure current during device switching. The TIAs convert current to voltage and may be integrated with a threshold comparator circuit to compare the flow-based output against a reference voltage to give binary high/low outputs for flow-based computing. This circuitry could also be shared across multiple 8x8 array modules to reduce area overhead. Lastly, the development of flow-based computing designs such that multiple functions can be accommodated with the same programmed memory array would be beneficial for in-memory computing.

6.3. References


