Resistive Switching Devices and Their Applications for Computing Beyond von Neumann Architecture

by

Jong Hoon Shin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctorate of Philosophy
(Electrical Engineering)
in the University of Michigan
2019

Doctoral Committee:

Professor Wei D. Lu, Chair
Professor L. Jay Guo
Professor Cagliyan Kurdak
Assistant Professor Becky Peterson

Jong Hoon Shin

jhoons@umich.edu

ORCID iD: 0000-0002-5120-2716

© Jong Hoon Shin 2019

Dedication

To the almighty God, my wife Eunice, my family, and friends

Acknowledgements

I gratefully acknowledge all the support I have been given during my course of study at the University of Michigan. I would like to express special thanks to my mentor, Prof. Wei D. Lu, who has always guided me with his insightful advices and inspiring comments. All the research I have explored and studied during my Ph. D course would not be possible without his pioneering spirit, passionate attitude, and thoughtful conversation. Besides my advisor, I would like to thank the rest of my committee members, Dr. Cagliyan Kurdak, Dr. L. Jay Guo, and Dr. Becky Peterson, for their great support and invaluable advice.

I also would like to thank my parents, family, and friends for supporting me through my challenging journey. A special thanks goes to my wife Eunice Y. Shin for providing me with unwavering love, support, and encouragement. I would like to express my gratitude to past and present members of the laboratory. Thank you to Dr. Shinhyun Choi, Dr. Ugo Otuonye, Dr. Patrick Sheridan, Dr. Jihang Lee, Dr. Wen Ma, Dr. Jiantao Zhou, Dr. Fuxi Cai, Dr. Chao Du, Dr. Yeonjoo Jeong, Dr. Mohammed Zidan, Dr. Xiaojian Zhu, Seung Hwan Lee, John Moon, Fan-Hsuan Meng, Qiwen Wang, Xinxin Wang for friendship, support, assistance and encouragement. In addition, I would like to extend my thanks to friends and colleagues in the Electrical Engineering and Computer Science who have provided me with assistance, support, and advice as well.

Table of Contents

Dedication ii

Acknowledgements iii

List of Figures vii

Abstract xviii

Chapter 1. Introduction 1

1.1 von Neumann Architecture and Memory Wall Problem 1

1.2 Neuromorphic Computing using Resistive Switching Devices 2

1.3 Analog TaOx-Based Valence-Change Memory (VCM) Device 6

1.4 Digital Cu-based Conductive Bridge Random-Access Memory (CBRAM) 9

1.5 Stochastic Switching Behavior of RRAM devices 11

1.6 Organization of the Thesis 13

Chapter 2. Experimental Demonstration of Feature Extraction and Dimensionality Reduction using TaOx Analog RRAM devices 15

2.1 Introduction 15

2.2 Fabrication of Forming-Free Tantalum-Oxide RRAM Devices 16

2.3 Analog Resistive Switching of TaOx Devices for Neuromorphic Application 19

2.4 Generalized Hebbian Rule for Unsupervised Learning 20

2.5 Operation of Memristor Array for PCA Implementation 21

2.6 Structure and Operation of the Test Board 24

2.7 Experimental Result of Dimensionality Reduction Based on PCA 27

2.8 Conclusion 33

Chapter 3. Self-Limited and Forming-Free CBRAMs With Double Al_2O_3 ALD Layers 35
3.1 Introduction........35
3.2 Device Fabrication........37
3.3 Self-Limited and Forming-Free Resistive Switching........38
3.4 Role of HT-ALD barrier layer........42
3.5 Cycling Test Results........43
3.6 Optimization of Pulse Programming Method........44
3.7 Conclusion........46
Chapter 4. Hardware Acceleration of Simulated Annealing of Spin Glass by RRAM Crossbar Array 47
4.1 Introduction........47
4.2 Spin Glass Problem and Simulated Annealing........48
4.3 Simulated Annealing Accelerated By RRAM Array and Stochastic CBRAM........50
4.4 Experimental Demonstration Of RRAM-Based Simulated Annealing........55
4.4 Parallel Spin-flip Strategy Using Memristive Simulated Annealing........59
4.5 Digital Annealing with RRAM crossbar array........61
4.6 Conclusion........64
Chapter 5. Stochastic Learning of Deep Neural Network Using Stochastic RRAM Crossbar Array 65
5.1 Introduction........65
5.2 Stochastic resistive switching with adjustable probability........71
5.3 Stochastic Binarization Mapping in RRAM Crossbar Array........75
5.3.1 Deterministic Data Migration Strategy........75
5.3.2 Single-Bit Stochastic Binarization Processes........75
5.3.3 Multi-Bit Stochastic Binarization Process........79
5.4 Stochastic Learning of Convolutional Neural Network........87

5.5 Optimization and Nonideality of RRAM-based Stochastic Binarization 91

5.5.1 Effects of Weight Precision on Stochastic Binarization 92

5.5.2 Device-to-Device Variability Effect 93

5.5. Conclusion 94

Chapter 6. Future Work and Summary 96

6.1 Solving general combinatorial optimization problems 96

6.2 Summary..99

Bibliography 101

List of Figures

Figure 1-1 Schematic of the von Neumann Architecture. Reproduced using data from Ref. [4]. The computer is composed of input devices, output devices, a memory unit, and a processing unit. The processing unit with control units and arithmetic units are connected to the memory unit through a system bus. 2

Figure 1-2 Neural network and its elements. Reproduced using data from Ref. [9]. (b) Schematic diagram of a hardware neuromorphic system that corresponds to the neural network in (a). Beside each neuron circuit, synapse elements that store weights between the neuron and its neighbors are needed. The grey communication bus transfers weighted neuron signal data from input neurons to output neurons. 3

Figure 1-3 (a) An illustration of a memristor, or a RRAM device, as a synapse between neurons. The top and bottom electrode of the RRAM device is connected to the pre-synaptic neuron and post-synaptic neuron, respectively. (Reprinted from Ref. [14] with permission) (b) A crossbar structure made of input neurons, output neurons, and RRAM devices that act as synapses with weights w_{ij} represented by their conductance values. 5

Figure 1-4 (Reprinted from Ref. [18] with permission) (a) Schematic of a TaOx-based bi-layer RRAM device. The TaOx layer is the oxygen-deficient layer that supplies oxygen-vacancies to the Ta_2O_5 switching layer. (b) Pulse test of TaOx devices showing LTP/LTD. The concept of the synaptic junction and STDP implementations are shown in (c) and (d), respectively. 8

Figure 1-5 (Reprinted from Ref. [22] with permission) TEM images of (a) a complete conductive filament in a Ag/a-Si/Pt CBRAM device and (b) partially formed Ag filament. (c) schematic of charge transfer and ion migration processes during resistive switching of a CBRAM device. 10

Figure 1-6 Schematics of (a) in-memory computation aided with periphery circuit (Reprinted from Ref.[23] with permission) and (b) crossbar structure for binary coded neural network. (Reprinted from Ref. [24] with permission) 11

Figure 1-7 (Reprinted from Ref. [26] with permission) Bias-dependent stochastic switching behavior. (a-c) Histograms of the wait time for the first switching event at bias voltages of 2.6, 3.2, and 3.6 V, respectively, while the deterministic SET voltage is 6.0V. (d) schematic of the ion hopping processes. (e) Log plot of the wait time τ vs bias voltage V. 12

Figure 2-1 (a) SEM image of TaOx array consisting of 18 rows and 2 columns. Inset is a schematic device structure of Ta/TaOx/Pd device (b) I-V sweep curve of resistive switching of a single TaOx device (c) Initial positive sweep for filament formation with different annealing time. Inset is relation diagram between the annealing time and forming voltage (Reprinted from Ref.[40] with permission) (d) Table of $|V_{SET}|$ and $|V_{forming}|$ measured from different TaOx-based RRAM device structures (reproduced using data from Ref[40], [46], [50]–[56])..18

Figure 2-2 (a) Distribution of the current response to 50 SET/ 50 RST pulse train of18 device and their average (b) Device model result and experimental average of pulse train. (Reprinted from Ref.[40] with permission)...20

Figure 2-3 (a) Schematic of the memristor network operation. The input voltage signals are applied to the rows and flow into the network, while the outputs are connected to the columns and

are collected as current. (b) Optical image of the test board. The memristor array was wire-bonded and inserted in the board. (Reprinted from Ref.[40] with permission) 21

Figure 2-4 List of parameters used in this study. $\alpha, \beta, \gamma, \delta, k, \mu 1, \mu 2$, and η are off-state leakage current, non-linear I-V scaling coefficient at off-state, on-state current coefficient, on-state non-linear I-V scaling coefficient, weight update coefficient, nonlinear voltage coefficient for set process, nonlinear voltage coefficient for reset process, and learning rate, respectively. (Reprinted from Ref.[40] with permission) 23

Figure 2-5 Circuit schematic of the test board (Reprinted from Ref.[40] with permission).........26

Figure 2-6 Flow chart of the PCA network operation (Reprinted from Ref.[40] with permission) 26

Figure 2-7 (a) Principal component analysis results. (a) Results obtained with untrained memristor network. The data are plotted based on the first (y1) and second (y2) output values. (b) Principal component analysis results after solving the traditional covariance matrix of the input data. The malignant and benign cells are largely separated into two clusters. (c) PCA results obtained from numerical simulation of the network, using the dynamic device model and Sanger's learning rule. (d) Experimental PCA results obtained from the memristor network using Sanger's learning rule after 35 cycles of training. The blue and magenta color labels in the plots represent the ground truth. Note the color labels are used only to highlight the effects of clustering in the plots but are not used in the network training or PCA analysis. (Reprinted from Ref.[40] with permission)...29

Figure 2-8 (a) Evolution of the principal component vectors. (a,b) The primary principal component (a) and the secondary principal component (b) before and after training. Black bars: memristor weights constituting the principal component vectors before the learning

process. Red bars: simulated memristor weights after the learning process. Blue bars: Experimentally obtained weights after the learning process. (c,d) Evolution of the Euclidean norm of the primary principal component and the secondary principal component vectors, respectively, during training. (e) Evolution of the inner product of the primary principal component vector and the secondary principal component vector during training. (Reprinted from Ref.[40] with permission) 31

Figure 2-9 Classification based on the trained memristor network. (a) Decision boundary (purple dotted line) obtained using a supervised training process. (b) Overlay of the decision boundary obtained from (a) and the test data from PCA analysis. A cell is classified to be malignant or benign based on whether it is located to the left or right of the decision boundary. The classification accuracy is obtained by comparing the classification with the ground truth (shown as the color label of the test data). (Reprinted from Ref.[40] with permission)...32

Figure 2-10 Classification results based on the exact solution of PCA. (a) Decision boundary (dotted line) obtained by fitting the training data set using logistic regression. (b) The decision boundary calculated from (a) overlaid with test data after PCA analysis. Prediction was made based on a data point's location with respect to the decision boundary. The blue and magenta color labels represent the ground truth. (Reprinted from Ref.[40] with permission)...33

Figure 3-1 (a) SEM image of Cu/CuO_x/LT-ALD/HT-ALD/Pd crossbar devices. Inset: the schematic structure of D-ALD devices. (b) Successive I-V sweeps including SET switching of as-fabricated device (red) (c) I-V curves of LT-ALD devices without CuO_x ($V_{forming}$~3V), LT-ALD devices with CuO_x ($V_{forming}$~2V), and HT-ALD devices

($V_{forming}$~5.5V). Results from three devices are shown for each structure. (d) Fitting of the D-ALD device I-V. The fitting was limited to below 2.0V where resistive switching occurred. (Reprinted from Ref.[68] with permission) 41

Figure 3-2 (a) An I-V sweep with larger voltage range showing over-programming (red) and the subsequent restore to the original resistive switching curve (blue) (b) Schematic illustration of the switching mechanism of D-ALD device. (Reprinted from Ref.[68] with permission) 43

Figure 3-3 (a) Pulse cycling test (3.0V/5ms for SET pulse, -2.5V/5ms for RST pulse, and 1.0V for read pulse) (b) Cumulative probability of resistance in LRS and HRS. Inset: the box plot that summarize C-to-C variation. (c) Box plots of on/off current from 10 devices (d) retention test of LRS and HRS at T = 100°C. (Reprinted from Ref.[68] with permission) 44

Figure 3-4 (a) Schematic pulse sequence consisting of SET pulse train and RST pulse train with verification scheme, for the cumulative fixed pulse mode (C-FP) and the non-cumulative fixed pulse mode (NC-FP). (b) Switching probability of SET (upper) and RESET (lower) for attempts using C-FP (purple) and NC-FP (red). The conditions of SET, RST, pre-SET pulses are fixed at 3.0V/5ms, -2.5V/5ms, and 2.0V/5ms, respectively. (Reprinted from Ref.[68] with permission) 46

Figure 4-1 A 2D spin glass and the spin interactions represented by (a) connections to neighboring spins and (b) circular graph showing the complex couplings. (Reprinted from Ref.[90] with permission) 49

Figure 4-2 Flow chart of the SA algorithm. (b) Schematic showing finite spin flip probability even for positive ΔH can help the system escape from local optima. (Reprinted from Ref.[90] with permission) 49

Figure 4-3 (a) ΔHy due to the change of σy surrounded by its neighbor spins. (b) CS matrix where the 5th column represents interaction between 5th spin and all the other spins (c) Schematic of inner product between the 5th CS column vector and spin vector σ conducted by RRAM array. (Reprinted from Ref.[90] with permission) 50

Figure 4-4 81×81CS matrix of a 9×9 2D spin array. The large but sparse CS matrix can be sliced to fit into a smaller RRAM array. (Reprinted from Ref.[90] with permission) 51

Figure 4-5 9 sub-patterns with three columns each from the 81×81 CS matrix, depending on the position of the spin in the 2D spin glass. (a) Top-Edge Row case, (b) Mid Row case, and (c) Bottom-Edge Row case. (d) All the non-zero and unique patterns in (a-c) can be stored in a single 11×3 RRAM array. (Reprinted from Ref.[90] with permission) 52

Figure 4-6 (a) Schematic of the Ta_2O_5-based RRAM cell and array structure. SEM image of the RRAM crossbar array. (b) Test board comprised of FPGA, peripheral circuit, and the RRAM array chip for experimental implementation of simulated annealing. (Reprinted from Ref.[90] with permission)....................52

Figure 4-7 (a) I-V curves showing the forming (red) and subsequent switching (blue) processes. (b) Distribution of $V_{Forming}$, V_{SET}, V_{Reset} of the 33 cells in the RRAM array. (c-d) Variation of device current without (c) and with (d) write-verify pulse method. (Reprinted from Ref.[90] with permission) 54

Figure 4-8 (a) Structure and SEM image of Cu-based CBRAM devices. (b) Experimentally measured probability of HRS→LRS switching (blue). The Boltzmann factor (red) can be

obtained by the probability of the device staying at HRS after applying a single SET pulse with pulse width Δt. (Reprinted from Ref.[90] with permission)....................................55

Figure 4-9 Flowchart of implementing the SA algorithm using RRAM array for the 2D spin glass problem. (Reprinted from Ref.[90] with permission) 56

Figure 4-10 (a) Randomly initialized 15×15 spin array (with 225 spins). (b) The sparse 225×225 CS matrix. (c) Coupling strength patterns stored and measured from the RRAM array used in the experimental setup. (Reprinted from Ref.[90] with permission)............................56

Figure 4-11 Evolution of the spin configuration at different time steps for the fixed spin-edge case. Data obtained experimentally from the RRAM array-based hardware system. (Reprinted from Ref.[90] with permission)...58

Figure 4-12 Time-dependent evolution of the spin glass system solved by the RRAM hardware, for random initial states with no fixed spins. Two ground states with global energy minima, 'all-up' state and 'all-down' states, can be generated from the same initial state in different runs. (Reprinted from Ref.[90] with permission)...58

Figure 4-13 (a) Average energy and (b) magnetization as a function of cooling schedule. Conventional software version of SA (red) and experimental SA results obtained from the RRAM array (blue) are compared. (Reprinted from Ref.[90] with permission) 59

Figure 4-14 Schematic illustration of multi-spin flip method that exploits parallel vector-matrix multiplications in RRAM crossbar array. (Reprinted from Ref.[90] with permission) 60

Figure 4-15 Comparison of (a) energy, (b) magnetization, and (c) spin configuration snap shots, for results obtained using the single-spin method with 100 iterations per time step (blue), single-spin method with 200 iterations per time step (black), and double-spin method with

100 iterations per time step (red). All results are obtained from the RRAM hardware setup. (Reprinted from Ref.[90] with permission) 61

Figure 4-16 Comparison between multiple spin flip strategy and parallel trial strategy (Reprinted from Ref.[90] with permission)..62

Figure 4-17 Acceleration of convergence with parallel spin trial method (Reprinted from Ref.[90] with permission) 63

Figure 5-1 Properties of deep neural networks for the ImageNet dataset, reproduced using data from Ref.[97]–[99] As the number of the convolution layers increases the top-error rates reduce while the total number of parameters and the number of MAC operations for a single input increase. 66

Figure 5-2 Rough energy cost and chip area for various operation and bit-width precisions in 45nm CMOS technology, Reproduced using data from Ref.[106]. As precision increases from 8bits to 32bits and the integers change to floating point numbers, the energy and area consumption increases accordingly. The last 2 rows are cost for data fetching from memory. 67

Figure 5-3 Schematic of BinaryConnect (Reproduced using data from Ref.[105]). At the first stage of network training, a binarized network (left panel) is created from the high precision network (right panel). The binary network is used in the propagation step to obtain gradients with respect to weights, which are then fetched to the high precision network for weight update. 69

Figure 5-4 Data flow of stochastic binarization process. Weights of the high precision network (stored in DRAM #1) are fetched to the CPU, the binarization probability is calculated, and compared to random numbers to determine the binary values. The generated binary

numbers are then transferred to a different part of DRAM (DRAM#2) to store the weights of the binary network..70

Figure 5-5 (a-b)Cumulative probability of the SET switching with (a) continuous pulse and (b) discrete pulses, obtained from Ref. [28]. (c-d) Stochastic switching observed in D-ALD CBRAM devices. Top panel (c) is a histogram of switching events for SET pulses (3.0V/10ms), and bottom panel (d) is a histogram of switching events for RST pulses (-2.5V/10ms). The histograms can be modeled with Poisson distributions [68]................74

Figure 5-6 Schematic of in-situ data migration as suggested in Ref.[24]. Resistive states of an RRAM array is tracked during in-situ data migration process. 1 and 0 are represented by the LRS and the HRS of the RRAM devices. The left column of the array contains data devices, and the right column contains target devices. With this data migration setup, when the data device is at LRS essentially all V_{SET} is delivered to the top electrode of the target device, while when the data device is at HRS only half of V_{SET} is delivered to the top electrode of the target device, allowing 1 and 0 to transferred to the target device..........77

Figure 5-7 Schematic of single-bit stochastic binarization for stochastic SET pulse with 50% SET probability. A stochastic SET pulse with 50% switching probability is applied to the target column. The stochastic pulse is delivered to target devices only if the associated devices is in LRS. As a result, the data '1' stored in the data column is copied to the target column by 50% probability. 78

Figure 5-8 Schematic of single-bit stochastic binarization for stochastic RST pulse with 50% RST probability. The initialization condition of the target column is all LRS in this case. The 50% RST stochastic pulse is applied to target column devices and turn them off by 50% only if the associated data column devices are in LRS. 78

Figure 5-9 Schematic of multi-bit stochastic binarization process using an RRAM array. The system includes a weight array, an inversion array, and a binary column. Initialization of the weight array and the inversion array are explained in STEP1 and STEP2. The grounded binary column is connected to the least-significant-bit (LSB) of the weight array with 50% SET pulse first in STEP3, then the LSB of the inversion array with 50% RST pulse in STEP4. The processes, STEP3 and STEP4 are repeated after moving to the next bit until binarization is complete. 83

Figure 5-10 The probability history of the binary column devices to be '1' or LRS from initial state to end of the stochastic binarization step. The log starts from 'initial' column in the table. For each nth LSB, STEP3 and STEP4 is conducted by connection of the grounded binary column to n^{th} LSB in the weight array and the inversion array, respectively. The stochastic binarization proceed from the 'initial' column to the 'Final Binary Probability' column through the table. Weights from -1.0 to +0.75 are successfully transformed to binary states following expected probability distribution. 84

Figure 5-11 The generalization of stochastic binarization of k-bit probability to (k+1)-bit probability. (a) For both '1' MSB and '0' MSB cases, the operations required for p_{k+1} obtained from p_k is described in the upper red box and lower blue box, respectively. In the listed operation, (b) The k-bit probability (red solid line) should be converted by MSB = '1' to red dashed line by changing the lowest probability from 0.0 to 0.5. (c) The k-bit probability (blue solid line) should be lowered by MSB = '0' to blue dashed line by changing the highest probability from 1.0 to 0.5. 84

Figure 5-12 Schematic of the proposed in-memory computing approach for multi-bit stochastic binarization of a neural network. The binarization process is much more simplified compared with the von Neumann architecture implementation depicted in Figure 5-4. ..85

Figure 5-13 Monte Carlo simulation results of stochastic binarization for 2 bit, 4 bit, and 6bit weights. Left panel of each case shows the weight array and the binary column. Red and blue colors in the weight array represent LRS and HRS, respectively. Color in the binary column representing the probability of having '1' (LRS) from the 1000 tests, based on data in the weight array. The right panel plots the measured LRS probability in the binary column with respect to the weight array value..86

Figure 5-14 Comparison of (a) error rate and (b) cost function of the softmax output layer among the baseline model (high precision network training), the deterministic binarization model, and the stochastic binarization model...90

Figure 5-15 Reduced MNIST training results, using the baseline model (left panel) and the stochastic binarization model (right panel)...91

Figure 5-16 Error rate for MNIST classification, for different weight precisions using stochastic binarization...93

Figure 5-17 Effect of device-to-device variability on error rate of stochastic learning of CNN...94

Figure 6-1 Schematic of implementing the travelling salesman problem (TSP). The nonzero elements of the $W(u,j)(v,k)$ matrix are formed by repeating patterns of the Wuv sub-array. 98

Abstract

As the demand for processing artificial intelligence (AI), big data, and cognitive tasks increases, new devices and computing architectures that can reduce the cost of the memory bottleneck have gained significant interest. One of emerging device that can enable non-von Neumann architectures such as neuromorphic computing and in-memory computing, resistive random-access memory (RRAM), has been extensively studied due to its properties such as nano-scale feature size, low power, and inherent functionalities that allow it to emulate biological synapses and stochastic events.

In this thesis, I will discuss optimization and development of RRAM devices as well as the application of RRAM devices for machine learning tasks and combinatorial optimization problems. Experimental demonstration of feature extraction by using tantalum oxide-based analog RRAM devices will be first introduced. To achieve robust operation of RRAM crossbar array, tantalum oxide devices are optimized to reduce the forming voltage. The optimized RRAM array is successfully used to perform principal component analysis (PCA), an unsupervised learning algorithm for feature extraction and dimensionality reduction, of a breast cancer dataset. In the second project, an RRAM structure that offers very low power and large on/off ratio is developed using copper active electrode and atomic layer deposited Al_2O_3 layers for low-power in-memory computation and digital version of neuromorphic computing applications. Desirable device performance such as self-current limiting, forming-free resistive switching, ultra-low current, and improved uniformity have been obtained.

Beyond device optimizations, I will present two projects that aim at demonstrating the applications of RRAM devices, implementing RRAM-based hardware acceleration of simulated annealing of the two-dimensional spin glass problem, and stochastic learning of deep neural networks. At the end of this thesis, a practical application of RRAM array for combinatorial optimization like travelling salesman problem is proposed as a future work.

Chapter 1. Introduction

1.1 von Neumann Architecture and Memory Wall Problem

Demand on cognitive computational tasks such as object and speech recognition using deep neural networks (DNNs) has been rapidly increasing in recent years.[1] These algorithms however require ever powerful hardware to implement. In particular, the data-intensive nature of DNNs is well known. For example, in the case of Alpha-Go, 48 CPUs and 8 GPUs were used to train the Go network.[2] In addition to machine learning, providers that support and analyze big data from web-based images, videos, and cloud system are suffering from the rising computational cost.[3] However, conventional computing hardware systems face fundamental limits originating from the nature of the von Neumann architecture, as shown in Figure 1-1.[4] Although the performance of processing units and the storage capacity of memory units have been continually improved by successive scaling according to the Moore's Law, data communication between processor and memory becomes the limiting factor of the performance for data-intensive tasks.[5] The von Neumann Bottleneck, or the memory wall problem, only gets worse with scaling since the innovations of bus technology that connects the processing units and the memory units are not as fast as the exponential improvements in complementary metal-oxide-semiconductor (CMOS) or dynamic random-access memory (DRAM) technology. Although computer architectures have evolved and various technologies such as cache, instruction-level parallelism, multithreading, and distributed computing such as graphics processing units (GPUs) have been developed to address the memory wall problem,[6] the fundamental problem associated with separated memory and

logic remain unchanged. As a result, current computer architectures are still significantly less efficient compared to a brain, a biological computer that performs at orders of magnitude higher power and space efficiencies than digital computers.[7]

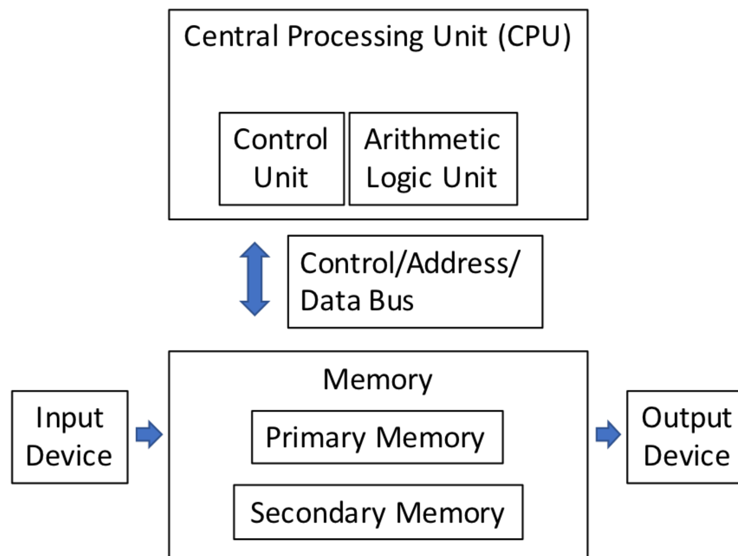


Figure 1-1 Schematic of the von Neumann Architecture. Reproduced using data from Ref. [4]. The computer is composed of input devices, output devices, a memory unit, and a processing unit. The processing unit with control units and arithmetic units are connected to the memory unit through a system bus.

1.2 Neuromorphic Computing using Resistive Switching Devices

To solve the von-Neumann bottleneck problem, neuromorphic computing systems that can mimic the structure of biological nervous systems such as the human brain have gained broad interest.[7], [8] The basic idea is to build networks of electronic elements that can emulate the functions of biological synapse and neurons. For example, a neural network shown in Figure 1-2 (a) can be mapped to the neuromorphic architecture shown in Figure 1-2 (b). In Figure 1-2 (a), each neuron in the input side (left) is connected to output neurons (right) through synaptic

connections with weights that govern the strength of the signal received by the output neuron. In this network, the input signal, or activation a_i from neuron i, and the synaptic weights w_{ij} associated with output neuron j are considered as vectors, and the output activation at neuron j is the function of the inner product $f(\Sigma w_{ij} a_i)$. Therefore, each output neuron can be interpreted as an individual processing unit that calculates the vector inner product $\sum w_{ij} a_i$ and activation function f. For a hardware version of neural network in Figure 1-2 (b), circuits that emulate the leaky-integrate-fire neuron, static random-access memory (SRAM)-based weight storage, and buses for inter-neuron communication are assembled to emulate training and inference in the biological network.

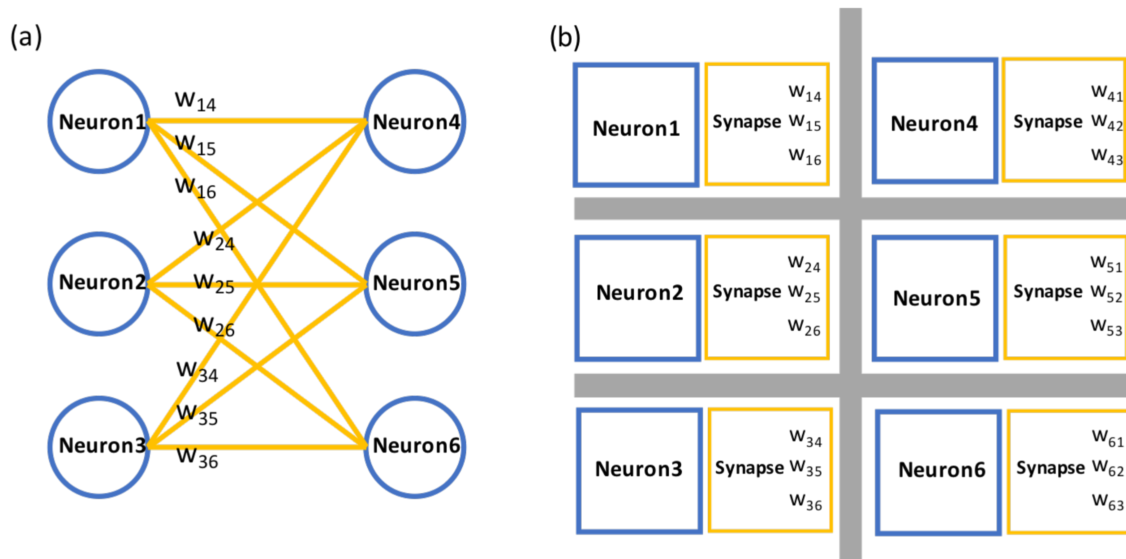


Figure 1-2 Neural network and its elements. Reproduced using data from Ref. [9]. (b) Schematic diagram of a hardware neuromorphic system that corresponds to the neural network in (a). Beside each neuron circuit, synapse elements that store weights between the neuron and its neighbors are needed. The grey communication bus transfers weighted neuron signal data from input neurons to output neurons.

Even though significant progress has been made on CMOS-based neuromorphic computing systems, as demonstrated by IBM's TrueNorth chip,[10] these approaches face significant challenges in terms of scaling and power consumption of CMOS devices in neuron circuits and SRAMs. Especially, synaptic circuits become a bottleneck since the number of synapses scales quadratically with the number of neurons (N^2) and the size of SRAM is more than 100 times of the minimum feature size ($>100F^2$). For these reasons, the second wave of innovation in neuromorphic systems has been initiated by developing novel non-volatile memories such as resistive random-access memories (RRAMs), phase-change random-access memories (PCRAM), and spin-torque transfer magnetic random-access memories (STT-MRAM) that can effectively store the synaptic weights, or better yet, effectively implement the synaptic functions directly.[11]

In particular, RRAMs, also known as memristors or memristive devices,[12], [13] have been successfully used to emulate biological synaptic behaviors such as long-term potentiation/depression (LTP/LTD) and spike-timing dependent plasticity (STDP), allowing energy-efficient, and cost-effective implementation of neuromorphic computing systems.[14] Key advantages of RRAMs for neuromorphic computing are their abilities to store analog weights and to perform vector-matrix operations directly through physics, i.e. Ohm's law and Kirchhoff's current law. Figure 1-3 (a) is an illustration of an RRAM device used as a synapse between a pre-synaptic neuron and a post-synaptic neuron. The two-terminal structure of RRAM devices with a top electrode (TE) and bottom electrode (BE) allow them to directly map the network topology in a crossbar form, as shown in Figure 1-3 (b). For example, an input activation a_i from the input neuron circuit i is converted to voltage level V_i and applied to the top electrode of a device with conductance G_{ij} that represents synaptic weights w_{ij} via the row direction. The output current I_j collected at the j-th output neuron is determined by Kirchhoff's current law and can be used to

produce activation function a_j, $a_j = a_j(I_j) = a_j(\Sigma G_{ij}V_i) = f_j(\Sigma w_{ij}a_i)$. By collecting current in all columns in the crossbar, the vector-matrix multiplication can thus be obtained in a single step.

The ability of RRAM crossbar to obtain the desired output using physics and in parallel makes it far more efficient than conventional CMOS implementations for neuromorphic and other machine-learning applications. The idea of using RRAM array for neuromorphic computing has been realized with small size application for three digit recognition (~10 cells)[15] to large size application (>1000 cells) for MNIST dataset.[16]

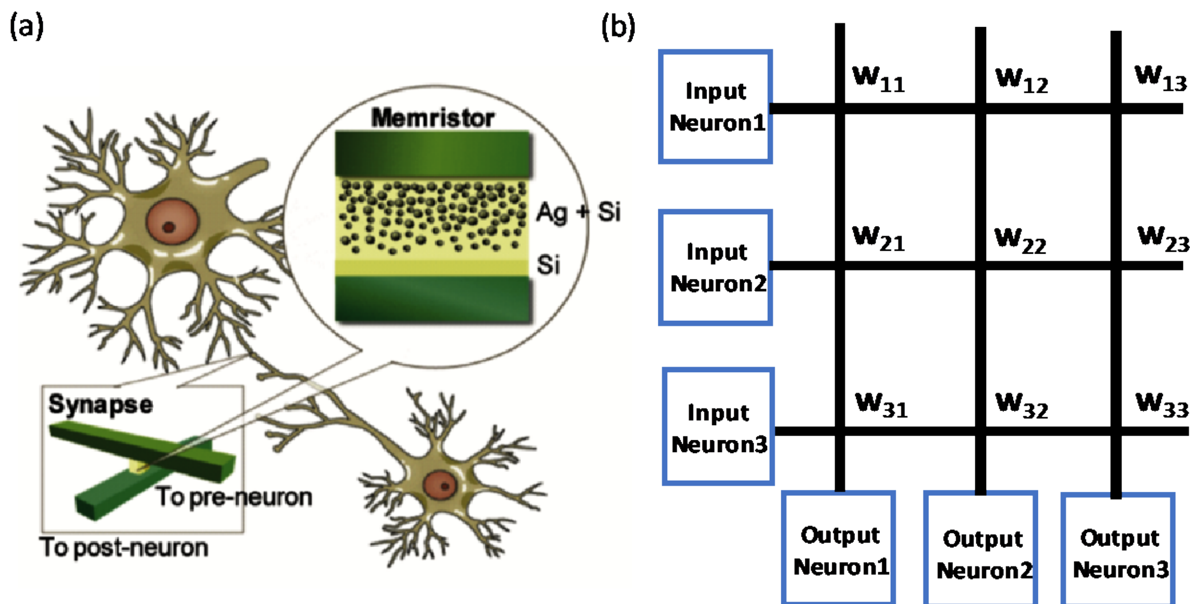


Figure 1-3 (a) An illustration of a memristor, or a RRAM device, as a synapse between neurons. The top and bottom electrode of the RRAM device is connected to the pre-synaptic neuron and post-synaptic neuron, respectively. (Reprinted from Ref. [14] with permission) (b) A crossbar structure made of input neurons, output neurons, and RRAM devices that act as synapses with weights w_{ij} represented by their conductance values.

1.3 Analog TaOx-Based Valence-Change Memory (VCM) Device

Among the complex processes and functions in a synaptic junction, a few properties are essential to the implementation of the functions of neural networks, such as the ability to store the synaptic weights and synaptic plasticity effects. The synaptic weight is a continuous metric that represents the strength of the pre-synaptic neuron for the activation of post-synaptic neuron. The larger the synaptic weight of a synapse, the larger proportion of the signal in the weighted sum that determines the post-synaptic neuron activation. The weight in the synaptic device also needs to be updated. Empirical "rules" that describe the weight changes have been observed in biology. For example, spike-timing-dependent-plasticity (STDP) describes that the weight change depends on the relative timing of the pre-synaptic neuron activation and the post-synaptic neuron activation. STDP has been implemented in RRAM devices where temporal difference of bi-directional activation induces larger voltage than the threshold that can initiate resistance switching.[17] In general, synaptic devices should have the ability to modulate their conductance incrementally in both positive direction (termed potentiation) and negative direction (termed depression). Recently, tantalum oxide (TaOx)-based RRAMs have been studied as analog non-volatile memory devices that can satisfy the requirements for analog synaptic devices.[18] TaOx RRAMs are in a category of valence-change memory (VCM), driven by mobile oxygen vacancies (V_os) that form conductive filaments in the oxide film. In Figure 1-4 (a), the device structure of a TaOx RRAM is illustrated. The TaO_x layer ($x < 5/2$) at the bottom supplies V_os to the highly resistive Ta_2O_5 layer. If the V_o concentration is high enough, the Ta_2O_5 layer can become conductive and result in a higher device conductance. In a typical operation, the top electrode is applied with negative voltage pulses (SET pulses) and the bottom electrode is grounded. The positively charged V_os are attracted to the top electrode and become accumulated and form a conductive filament made of high density of V_o.

Continued application of SET pulses can incrementally accumulate V_os to the filament and cause incremental increases in the device conductance. Similarly, positive voltage pulses (RESET pulses, or RST pulses) drive V_Os away from the conductive filament and decreases the device conductance.

Figure 1.4(b) shows a measurement setup with 20 consecutive SET pulses and 20 consecutive RST pulses, representing a test of the long-term potentiation/depression (LTP/LTD) phenomenon. The incremental conductance increases and conductance decreases can be clearly observed experimentally (bottom panel) and explained by simulation result using a physical device model. Moreover, learning rules such as STDP can be implemented as well (Figure 1-4 (d), further supporting the prospect of analog RRAMs for neuromorphic applications.

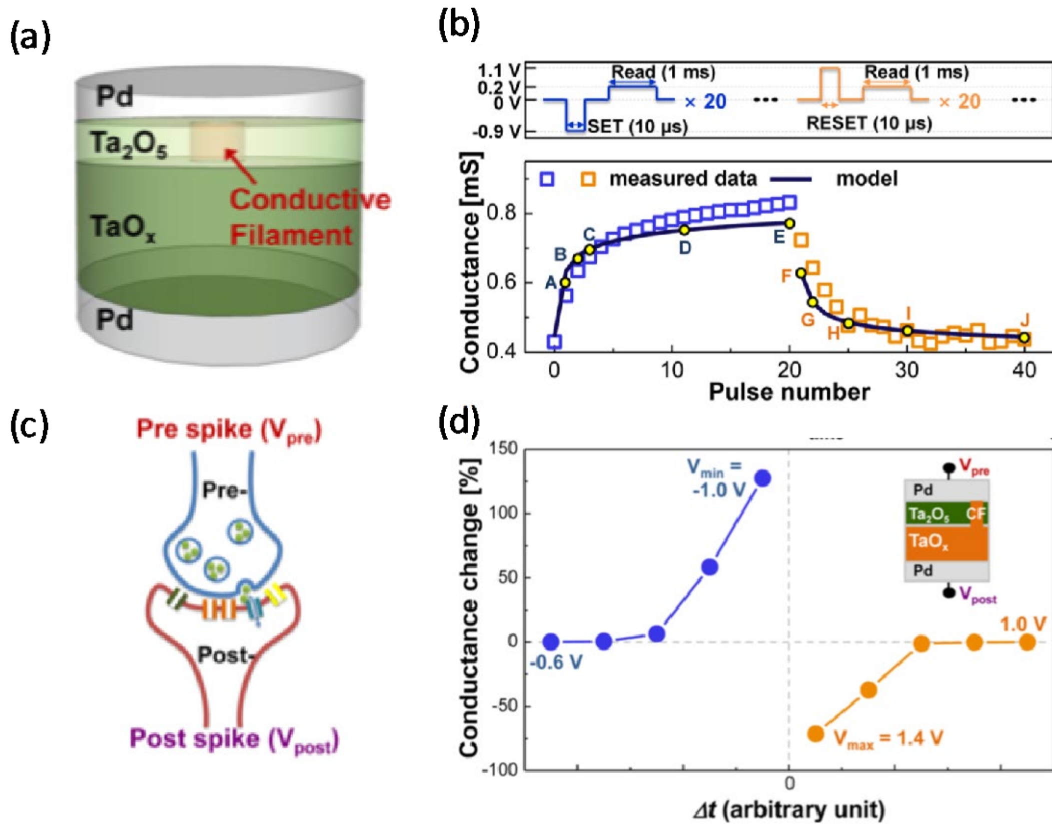


Figure 1-4 (Reprinted from Ref. [18] with permission) (a) Schematic of a TaOx-based bi-layer RRAM device. The TaOx layer is the oxygen-deficient layer that supplies oxygen-vacancies to the Ta_2O_5 switching layer. (b) Pulse test of TaOx devices showing LTP/LTD. The concept of the synaptic junction and STDP implementations are shown in (c) and (d), respectively.

1.4 Digital Cu-based Conductive Bridge Random-Access Memory (CBRAM)

Beside V_O based VCMs, resistive switching can also be obtained in devices based on metal ions such as copper or silver ions for conductive filament formation. These devices are also called conductive bridge random-access memory (CBRAM) devices, and have been considered as promising solutions for storage class memory since digital CBRAMs have shown better performance such as high on/off ratio, fast speed, high endurance, low power, and excellent scalability when compared with other approaches.[19], [20] The fundamental mechanism of a CBRAM device is the oxidation and reduction process of active metal atoms.[21] In the case of copper-based CBRAM, copper atoms at the active electrode can be oxidized and changed to copper cations, Cu^{+} or Cu^{2+}, when the active electrode is applied with a positive voltage larger than the total overpotential, which is the threshold potential that initiates the electrochemical reaction. Afterwards, the ionized cations are driven towards the opposite inert electrode (the cathode) by the applied electric field. At the cathode or inside the dielectric, the copper cations can capture electrons and become reduced to copper atoms. A Cu filament can be formed through nucleation and accumulation of the Cu atoms within the solid electrolyte. When the filament bridges the two electrodes, a much lower conductance can then be obtained in the SET process. The reverse processes lead to the filament rupture and the recovery of the high resistance state. The filament formation/rupture processes have been observed directly in the work of Y. Yang *et. al.* as described in Figure 1-5.[22]

Applications of CBRAM other than storage class memory have also been suggested as the device technology improves. The suggested applications include in-memory computing and binary coded neuromorphic computing. In-memory computing is a method to operate instructions within the memory to address the von Neumann bottleneck problem, as shown in Figure 1-6 (a).[23]

Moreover, neuromorphic computing by grouping a few binary CRAM devices (figure 1.6 (b)) to express analog weights in a neural network is another promising application since digital resistive switching processes are generally easier to control than analog switching processes.[24], [25] Therefore, the development and application of digital CBRAMs is also imperative to address the memory wall problem and the development of novel computing architectures.

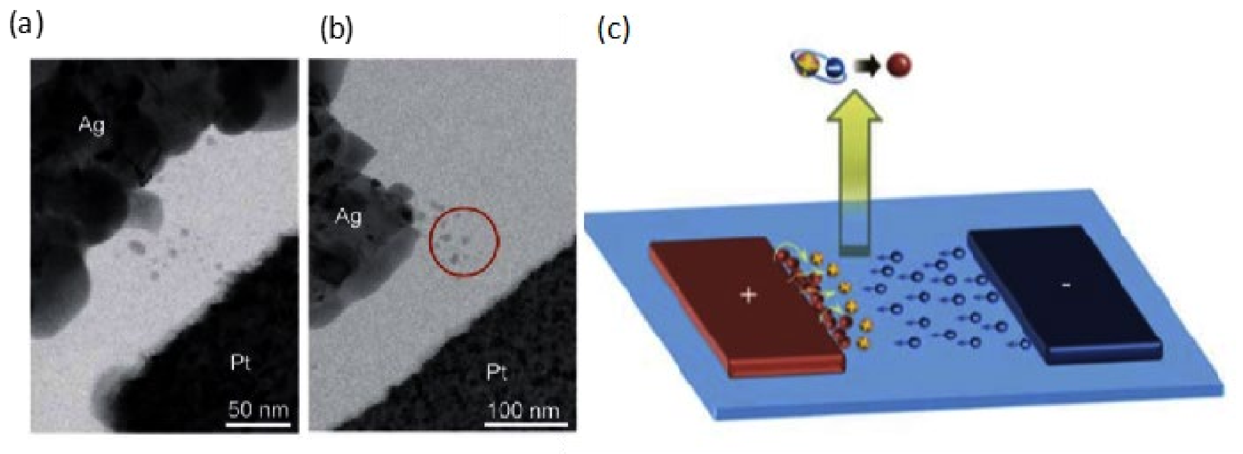


Figure 1-5 (Reprinted from Ref. [22] with permission) TEM images of (a) a complete conductive filament in a Ag/a-Si/Pt CBRAM device and (b) partially formed Ag filament. (c) schematic of charge transfer and ion migration processes during resistive switching of a CBRAM device.

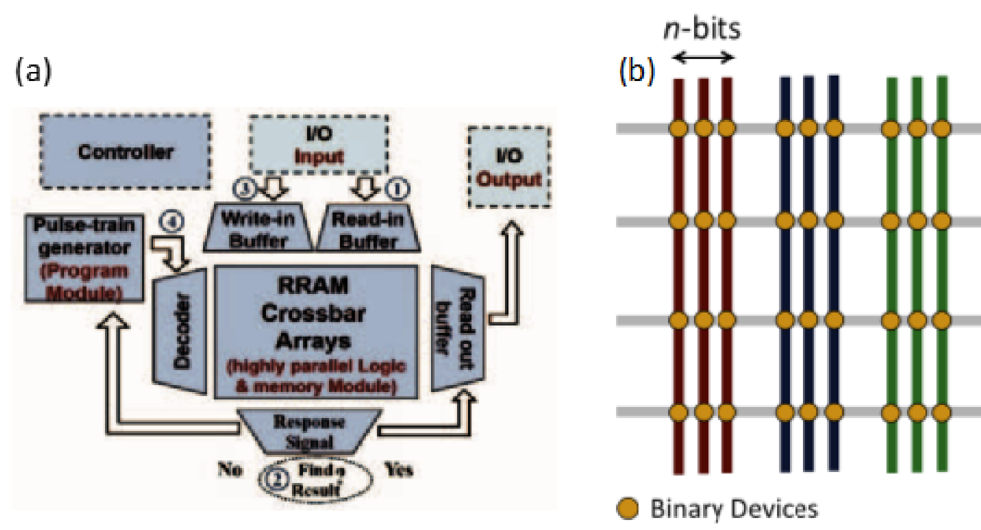


Figure 1-6 Schematics of (a) in-memory computation aided with periphery circuit (Reprinted from Ref.[23] with permission) and (b) crossbar structure for binary coded neural network. (Reprinted from Ref. [24] with permission)

1.5 Stochastic Switching Behavior of RRAM devices

Since resistive switching in RRAM is based on thermally activated processes over high energy barriers (which may be the redox potential or the ion migration barrier, depending on the materials and bias conditions), the processes inherently follow probability distributions determined by statistical physics. As a result, digital RRAM devices show pronounced stochastic switching behavior when the applied voltage is lower than the deterministic SET/RST voltage.[26], [27] According to *Jo et. al.*,[26] formation of a conducting filament in a Ag/a-Si/Si device is a consequence of thermally activated hopping process of Ag particles in a-Si, following Eq. 1-1, where the hopping rate Γ is inversely proportional to the characteristic time τ of the switching process, k_B is Boltzmann's constant, T is the absolute temperature, and ν is the attempt frequency.

$$\Gamma = {}^{1}/_{\tau} = \upsilon e^{-E_a{}'(V)/_{k_B T}} \qquad \text{(Eq. 1-1)}$$

Because the activation energy ($E_a'(V) = E_a - \alpha eV$) is dependent on the voltage applied to the device, the hopping rate Γ and waiting time τ of the first switching event can be controlled and show an exponential dependence on the applied voltage, as shown in Figure 1-7 (a-c).

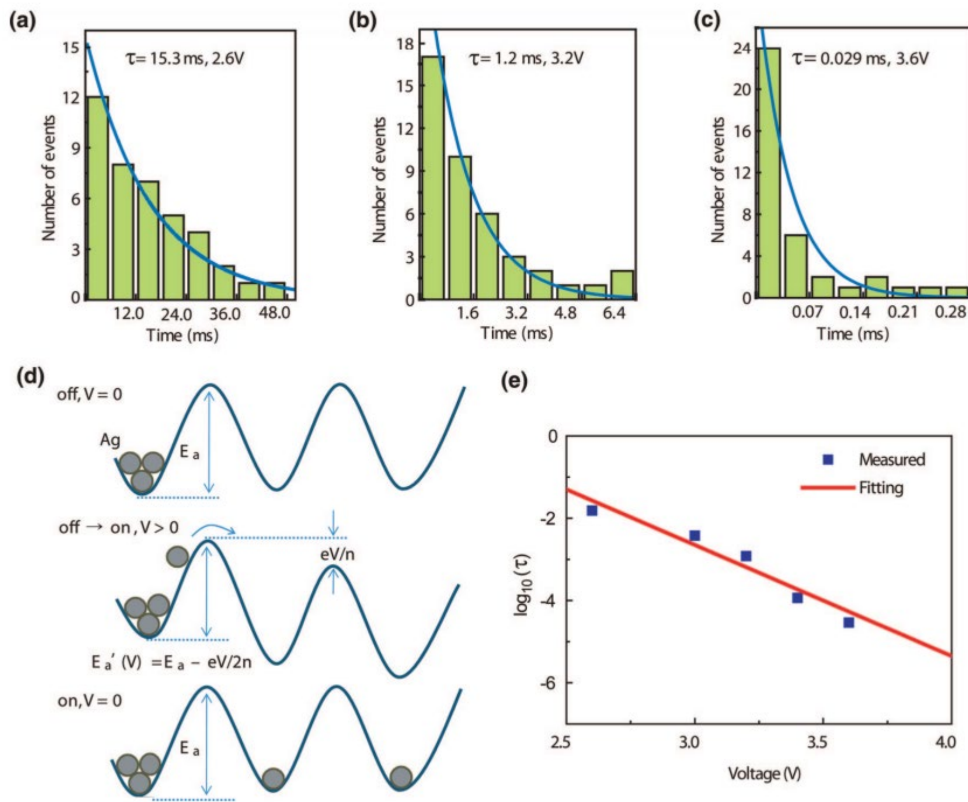


Figure 1-7 (Reprinted from Ref. [26] with permission) Bias-dependent stochastic switching behavior. (a-c) Histograms of the wait time for the first switching event at bias voltages of 2.6, 3.2, and 3.6 V, respectively, while the deterministic SET voltage is 6.0V. (d) schematic of the ion hopping processes. (e) Log plot of the wait time τ vs bias voltage V.

At low applied voltages below the deterministic SET voltage of 6.0V, the device switching shows clear stochastic behavior and can be described by a Poisson distribution. As the applied voltage increases from 2.6V to 3.6V, the characteristic time of the distribution decreases exponentially. The switching process also becomes more deterministic since the switching probability within a specific pulse width increases exponentially due to the barrier lowering mechanism depicted in Figure 1-7 (d).

Even though the stochasticity of RRAM devices can be regarded as a probabilistic failure mechanism of resistive switching that should be eliminated for memory applications, it can be beneficial in other applications such as stochastic computing and stochastic neural networks. For example, S. Gaba *et. al.*[28] exploited the stochastic property of Ag/a-Si/poly-Si RRAM devices for experimental demonstration of stochastic computing for neuromorphic applications. A parallel array of the binary RRAMs succeeded to represent an input analog value with their time-space switching probability distribution. The proof-concept work ignited interests in utilizing the stochastic switching property of RRAMs for machine learning algorithms such as gradient descent optimization, k-means clustering, and winner-take-all networks.[29], [30]

1.6 Organization of the Thesis

In this thesis, research projects on analog VCM devices, digital CBRAM devices, stochastic binary devices, and their applications for non-von Neumann architectures including neuromorphic computing and in-memory computing systems are first introduced in Chapter 1. Chapter 2 discusses optimization of TaOx devices to achieve forming-free characteristics and improved analog switching behaviors for neuromorphic networks with unsupervised learning, and the demonstrations of functions such as feature extraction and dimensionality reduction to process

a breast cancer dataset. In Chapter 3, a novel design of Cu-based CBRAM devices with uniform, self-limited, forming-free, and robust switching is presented. Chapter 4 discusses studies on hardware acceleration of combinatorial optimization problems using both analog RRAM devices and stochastic RRAM devices. In particular, the two-dimensional spin glass problem, one famous NP-hard problem, was solved using the simulated annealing algorithm implemented in the RRAM system. A strategy for further acceleration of RRAM-based in-memory computing inspired by the quantum annealing process is also introduced. In Chapter 5, stochastic learning for convolutional neural networks with parallel stochastic binarization of analog weights is explained and its application of the MNSIT dataset is estimated. Finally, future works on general approaches for acceleration of difficult optimization problems using RRAM arrays is discussed in Chapter 6.

Chapter 2. Experimental Demonstration of Feature Extraction and Dimensionality Reduction using TaOx Analog RRAM devices

2.1 Introduction

The von Neumann architecture, broadly used in digital computing systems, now faces significant challenges for data-intensive tasks due to the inherent limitation of the data transfer rate between the memory and the central processing unit (CPU). Alternative approaches based on neuromorphic computing and machine learning approaches have been extensively studied to solve such "big data" problems.[31], [32] A common technique used to solve data-intensive problems is feature extraction, which has been widely used for making predictive models such as pattern recognition in data analysis.[33] Feature extraction aims to reduce the dimensionality of the data by mapping the original input data into a new space based on identified vectors ("features"). Particularly, principal component analysis (PCA) is widely used for linear dimensionality reduction and has been applied in applications ranging from machine learning to medical fields for tasks such as image processing, face-recognition, interpretation of genetic data, and disease diagnostic predictions and treatments.[34]–[39] However, identifying the features is compute-intensive and traditionally relies on solving the covariance matrix, whose size grows quadratically as the input.[33] In this study, using an unsupervised, online learning rule, we show experimentally that simple memristor-based crossbar networks can learn the principal components from sensory data and effectively separate unlabeled data into clusters.[40] After data clustering, a conventional supervised learning process (logistic regression) can then be used to define a decision boundary

and classify the data with high precision, for example, successfully labeling tumors as malignant or benign with 97.1% success rate, comparable to results obtained from directly solving the covariance matrix.

Memristors, nanoscale resistive switching devices that are often called resistive random-access memory (RRAM) devices when used in memory applications, have attracted significant interest recently as a promising candidate for neuromorphic computing systems.[14], [41] Memristor crossbar arrays are particularly suitable for neural network implementations due to the following reasons. First, the crossbar array can directly implement vector-matrix multiplications (e.g., dot-product) in physics due to the nature of the two-terminal resistive device: the output current vector is a product of the input voltage vector multiplied by the conductance matrix of the memristor array. Second, the ability to incrementally change (and store) the resistance state is compatible with online learning where simple voltage pulses can be used to update the conductance (weight) matrix.[15], [42], [43] A typical memristor device consists of a transition metal oxide layer such as TiOx, HfOx, WOx, TaOx sandwiched by a pair of electrodes,[13], [44], [45] whereas excellent performance such as high density, low power consumption, long cycling endurance, and sub-nanosecond switching speed have already been reported.[46], [47] During weight update, the resistance of the memristor device can be adjusted incrementally by controlling the distribution of oxygen vacancies, which modulate the local conductivity and the overall conductance of the device.[18]

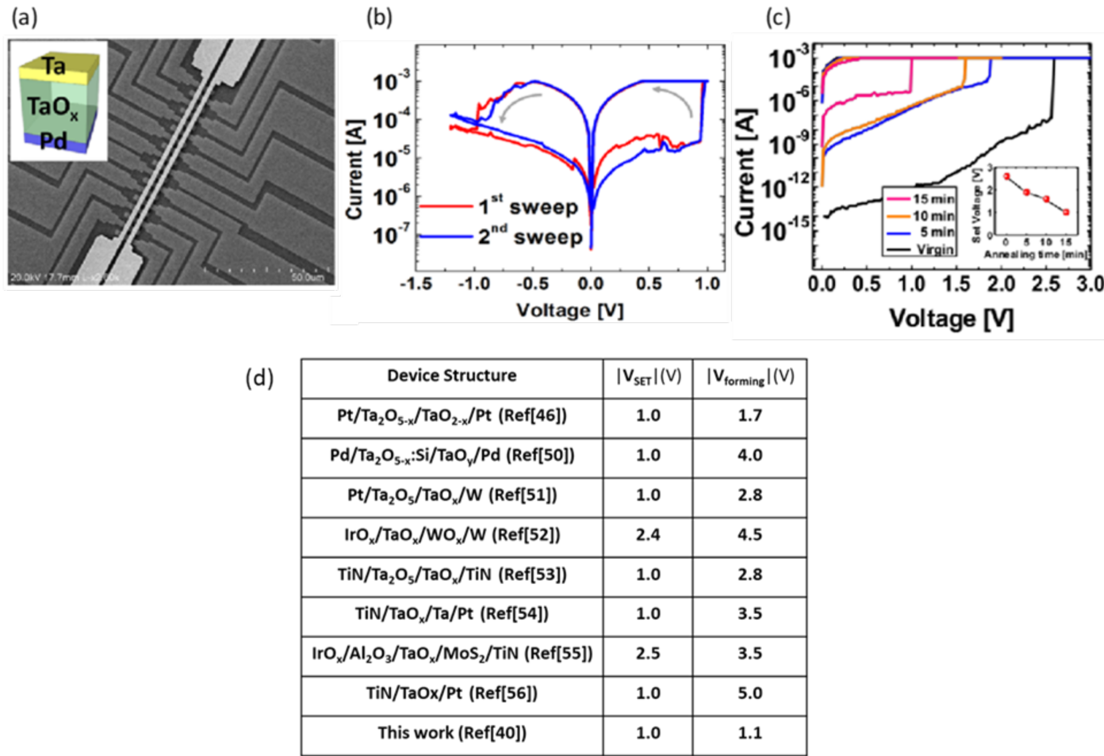
2.2 Fabrication of Forming-Free Tantalum-Oxide RRAM Devices

Crossbar arrays based on a forming-free, tantalum-oxide memristor structure were used in this study to experimentally implement PCA analysis. In general, RRAM devices require an initial

electroforming process, where a high voltage is used to create the ionic distributions necessary for subsequent resistive- switching processes.[48] In a passive memristor crossbar, the high forming voltage (typically ~5 V, whereas the set voltage is ~1.0 V for Ta_2O_5-based devices)[49] can cause damage to the half- selected devices that are already formed and share the same row as the target device, because a voltage of ~2.5 V will be applied to these half-selected devices in a standard protective voltage scheme. Therefore, devices that are forming-free or require only low-voltage forming are essential for the successful operation of passive crossbar systems, that is, systems without the access transistor in one-transistor one-resistor (1T1R) type implementations. To achieve reliable forming-free behavior, a thin tantalum pentoxide (Ta_2O_5) layer and Ta metal were used as the switching layer and the reactive top electrode, respectively. In detail, the memristor crossbar array was fabricated on a Si substrate with a 100nm thermal SiO_2 layer. The bottom electrodes, consisting of 5 nm thick NiCr and 40 nm thick Pd, were patterned by photolithography and deposited by e-beam evaporation followed by a liftoff process. Next, the 10 nm Ta_2O_5 switching layer was deposited by radio frequency (RF) sputtering for 200s at room temperature. The top electrodes, consisting of 40 nm thick Ta and 100 nm thick Pd, were fabricated by photolithography, e-beam evaporation and liftoff. After fabrication, the devices were annealed using rapid thermal annealing (RTP) at 300 °C in N_2 gas for 15 min to create oxygen vacancies in the Ta_2O_5 switching layer after device fabrication in lieu of the forming process.

Figure 2-1 (a) shows a scanning electron microscopy (SEM) image of an as-fabricated array consisting of 18 rows and 2 columns. A 9 × 2 subarray out of the as-fabricated array was used in the PCA analysis. Figure 2-1 (b) shows the direct current (dc) current−voltage (I−V) curves of a typical device starting from the virgin state, showing typical bipolar resistive switching characteristics. Additionally, the first sweep and the subsequent sweep show nearly identical set

and reset characteristics, confirming the form-free behavior. The optimization of annealing process is developed by observation of decreasing forming voltage as annealing time increases described as Figure 2-1 (c). Figure 2-1 (d) compares $|V_{SET}|$ and $|V_{forming}|$ from different TaOx-based RRAM device structures to highlight the improvement of low forming properties in this study.



Device Structure	$\|V_{SET}\|$ (V)	$\|V_{forming}\|$ (V)
$Pt/Ta_2O_{5-x}/TaO_{2-x}/Pt$ (Ref[46])	1.0	1.7
$Pd/Ta_2O_{5-x}:Si/TaO_y/Pd$ (Ref[50])	1.0	4.0
$Pt/Ta_2O_5/TaO_x/W$ (Ref[51])	1.0	2.8
$IrO_x/TaO_x/WO_x/W$ (Ref[52])	2.4	4.5
$TiN/Ta_2O_5/TaO_x/TiN$ (Ref[53])	1.0	2.8
$TiN/TaO_x/Ta/Pt$ (Ref[54])	1.0	3.5
$IrO_x/Al_2O_3/TaO_x/MoS_2/TiN$ (Ref[55])	2.5	3.5
TiN/TaOx/Pt (Ref[56])	1.0	5.0
This work (Ref[40])	1.0	1.1

Figure 2-1 (a) SEM image of TaOx array consisting of 18 rows and 2 columns. Inset is a schematic device structure of Ta/TaOx/Pd device (b) I-V sweep curve of resistive switching of a single TaOx device (c) Initial positive sweep for filament formation with different annealing time. Inset is relation diagram between the annealing time and forming voltage (Reprinted from Ref.[40] with permission) (d) Table of $|V_{SET}|$ and $|V_{forming}|$ measured from different TaOx-based RRAM device structures (reproduced using data from Ref[40], [46], [50]–[56])

2.3 Analog Resistive Switching of TaOx Devices for Neuromorphic Application

In memristor based neural networks, the weights are represented by the memristor conductance and implementation of online learning requires the weights to be incrementally updated. Figure 2-2 (a) shows the conductance updates of devices in the 9 × 2 subarray. A train of 50 pulses (1.1 V, 3 μs duration) was used to increase the device conductance, followed by another train of 50 pulses (−1.4 V, 30 μs duration) to decrease the device conductance. The device conductance was monitored with a 0.3 V read pulse after each programming or reset pulse. As can be seen in Figure 2-2 (a), a positive pulse increases the memristor conductance incrementally while a negative pulse decreases the memristor conductance incrementally, demonstrating analog switching behavior in the device. The analog switching behavior can be attributed to the drift and diffusion of oxygen vacancies in the TaOx switching layer that incrementally changes the profile of the oxygen vacancy-rich conduction region.[18], [50] Figure 2-2 (b) plots the average conductance values measured from the 18 devices in the 9 × 2 subarray shown in Figure 2-2 (a), along with simulation results based on a dynamic memristor model, showing very good agreements between the experimental observations and the model.[57] Below are the physical model of the TaOx analog devices consisting of the I-V equation Eq 2-1 and the state variable dynamic equation Eq 2-2 where *w* is internal state variable and γ, δ, α, β are parameters related to material properties such as effective tunneling distance, tunneling barrier, the depletion width of the Schottky barrier region and barrier height. *u(x)* is the Heaviside step function, *k*, *μ* are positive parameters determined by material properties such as ion hopping distance and hopping barrier heights.

$$\mathrm{I} = \omega\gamma\sinh(\delta \times V) + (1-\omega)\alpha\left(1 - e^{-\beta \times V}\right) \qquad \text{(Eq. 2-1)}$$

$$\frac{d\omega}{dt} = (\omega - 1)^2 \mathrm{k}(e^{-\mu_1 V} - e^{\mu_2 V})u(-V) + \omega^2 \mathrm{k}(e^{-\mu_1 V} - e^{\mu_2 V})u(V) \qquad \text{(Eq. 2-2)}$$

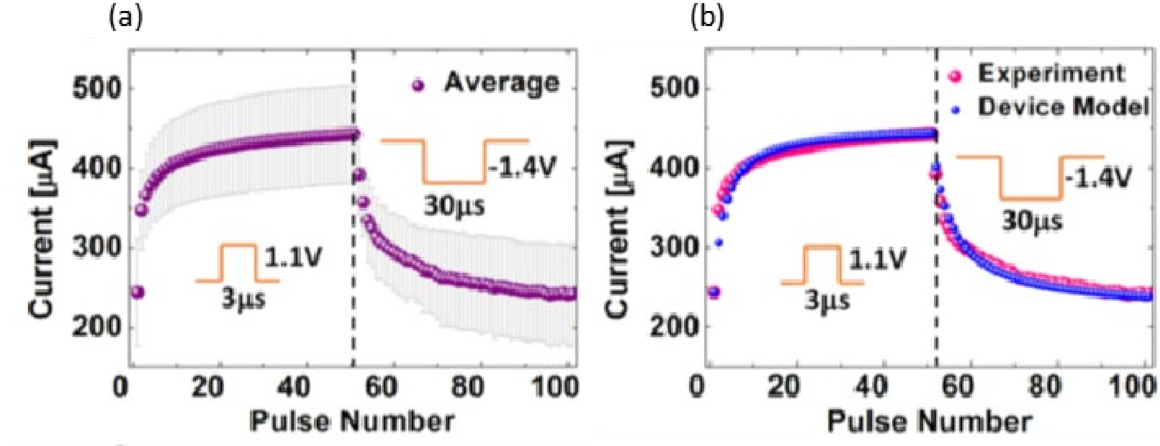


Figure 2-2 (a) Distribution of the current response to 50 SET/ 50 RST pulse train of18 device and their average (b) Device model result and experimental average of pulse train. (Reprinted from Ref.[40] with permission)

2.4 Generalized Hebbian Rule for Unsupervised Learning

To obtain the principal components of the input data set through online learning, we implemented Sanger's rule, also known as the generalized Hebbian algorithm in the memristor crossbar.[58], [59] Specifically, Sanger's rule states that the desired weight change is determined by the current weight (g), the output response (y), and the input (x), following Eq. 2-3.

$$\Delta g_{ij} = \eta y_j (x_i - \sum_k g_{ik} y_k) \qquad \text{(Eq. 2-3)}$$

where η is the learning rate ($0 < \eta \ll 1$), x_i represents the input data at input (row) i ($1 \le i \le 9$ in this study), g_{ij} is the weight at row i and column j in the neural network, and j = 1 or 2 for the primary principal component or the secondary principal component. After training, the weight vectors in columns 1 and 2 determine the primary and secondary principal components of the input

data set, respectively. After learning the principal components, the trained artificial neural network was then used to perform dimensionality reduction and clustering analysis.

2.5 Operation of Memristor Array for PCA Implementation

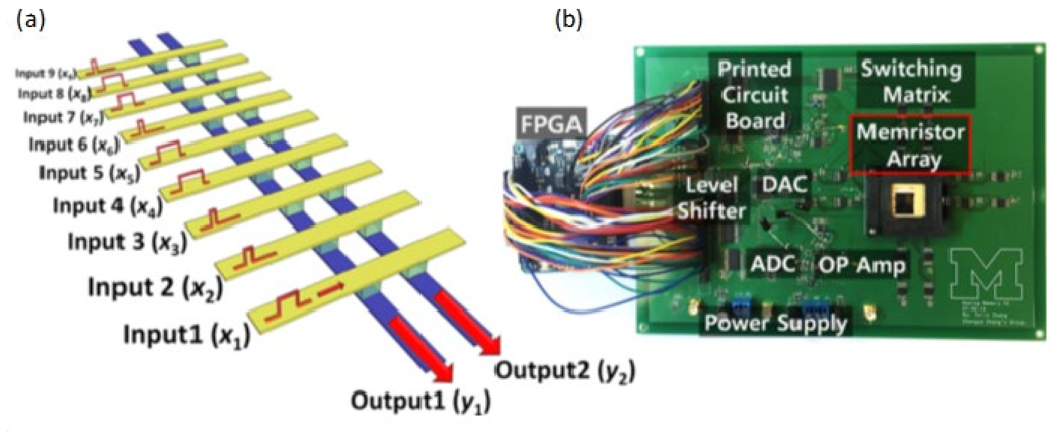


Figure 2-3 (a) Schematic of the memristor network operation. The input voltage signals are applied to the rows and flow into the network, while the outputs are connected to the columns and are collected as current. (b) Optical image of the test board. The memristor array was wire-bonded and inserted in the board. (Reprinted from Ref.[40] with permission)

Figure 2-3 (a) shows a schematic of the memristor-based neural network structure. The input channels are connected to the rows and the output channels are connected to the columns of the memristor crossbar. By using voltage pulses with different pulse widths as the input, the output vectors are determined by the vector–matrix dot-product of the input signal and the memristor weight matrix, while the network learns the principal components by adjusting the weights of the

memristor network during training.[57] To map the physical parameters obtained in the memristor network with the parameters used in PCA analysis and Sanger's rule, a linear transformation from physical charge to output value is needed. Specifically, with the application of an input x_i (represented by the width of the input voltage pulse), the amount of charge collected at the output in the memristor network can be calculated from Kirchhoff's law as

$$Q_j = \sum_i \left[w_{ij}A + \left(1 - w_{ij}\right)B\right] x_i \qquad \text{(Eq. 2-4)}$$

where Q is the charge collected at the output, x_i is input signal applied at input row i, w_{ij} is the state variable of the memristor device at row i and column j as discussed in Eq. 2-1 and Eq. 2-2, while the constants in Eq. 2-1 are lumped into prefactors A and B. The output y_j used to perform the PCA analysis is then obtained from the charge Q_j through Eq. S4:

$$y_j = \frac{2Q_j}{A-B} - \sum_i \left[\frac{A+B}{A-B} x_i\right] = \sum_{i=1}^{n} g_{ij} x_i \qquad \text{(Eq. 2-5)}$$

$$\text{where} \qquad g_{ij} = 2w_{ij} - 1 \qquad \text{(Eq. 2-6)}$$

where y and g obtained in Eq. 2-5 and Eq. 2-6 are used to perform the weight updates and PCA analysis following Sanger's rule (Eq. 2-2). After training, the weights g in columns 1 and 2 form the (first and 2nd, respectively) principal components of the input data set.

From the Sanger's rule shown in Eq. 2-3, the desired weight update Δg_{ij} and corresponding pulse width $|\Delta t|$ can be calculated with Eq. 2-7. Programming voltage pulses are then applied to the inputs to modify the memristor weights. The training pulses are determined by the polarity and magnitude of Δg_{ij}, with potentiation (1.1 V) pulses for positive Δg_{ij} and

depression (-1.4 V) pulses for negative Δg_{ij}, while the pulse widths are determined by the magnitude of $|\Delta g_{ij}|$ and g_{ij}. Here, a simple approach is used to compensate for the non-linear response of the internal state variable *w*. Specifically, the pulse width $|\Delta t|$ is determined as Eq.2-7. The training data set consisted of 100 randomly sequenced data points (50 data points from benign cells, 50 data points from malignant cells). After training, the network was used to analyze another 583 data points that were not included in the training data set.

$$\Delta t_{ij} = \frac{2}{k\left(e^{-\mu_1 V_{potentiation}} - e^{\mu_2 V_{potentiation}}\right)}\left(\frac{-1}{g_{ij,after}-1} + \frac{1}{g_{ij,before}-1}\right) u(\Delta g)$$

$$+ \frac{2}{k\left(e^{-\mu_1 V_{depression}} - e^{\mu_2 V_{depression}}\right)}\left(\frac{-1}{g_{ij,after}+1} + \frac{1}{g_{ij,before}+1}\right) u(-\Delta g) \qquad \text{(Eq. 2-7)}$$

Equations	Parameters	Values
I-V Equation	α	1.58×10^{-3}
	β	0.5
	γ	3.01×10^{-3}
	δ	0.5
State variable dynamic equation	k	1.0×10^{-4}
	μ_1	19.25
	μ_2	13
Sanger's rule	η	0.001

Figure 2-4 List of parameters used in this study. α, β, γ, δ, k, μ_1, μ_2, and η are off-state leakage current, non-linear I-V scaling coefficient at off-state, on-state current coefficient, on-state non-linear I-V scaling coefficient, weight update coefficient, nonlinear voltage coefficient for set process, nonlinear voltage coefficient for reset process, and learning rate, respectively. (Reprinted from Ref.[40] with permission)

In this study, a standard breast cancer data set from the University of Wisconsin Hospital was used as the input.[60] The data set consists of breast cell mass properties measured in 9 categories and each property is scored from 1 to 10. Each input to the memristor network is thus a nine-dimensional vector consisting of scores from the nine measurements. Transformation of the data is achieved in the memristor array through a simple "read" operation, where the input signals are applied to the memristor array as voltage pulses with fixed amplitude (0.3 V) and variable pulse widths (0 to 1000μs duration with 100μs unit pulse width) proportional to the values of the input data. The output charge collected at column j then corresponds to the dot-product of the input vector and the conductance vector stored in column j, allowing the mapping from the original nine-dimensional space to a two- dimensional output space (for the case of considering two principal components). To learn the principal components, programming (+1.1 V) or erasing (−1.4 V) voltage pulses are applied to the memristor array with pulse widths calculated from the amount of the desired weight changes. All the parameters used for the equations about I-V characteristics, weight update, and Sanger's rules are listed in Figure 2-4.

2.6 Structure and Operation of the Test Board

The experiments were carried out using a custom-built test board where the memristor crossbar array was wire bonded to a chip carrier and connected to the periphery circuitry on the board, shown in Figure 2-3 (b). The circuitry on the board provides peripheral functions such as voltage signal generation, addressing signals to the proper row and column in the crossbar, and receiving current at the output of the array. Digital to analog converters (DACs) were used to

provide bias to the top electrodes (TEs) and bottom electrodes (BEs) of the memristor crossbar. An analog to digital converter (ADC) and an Op amp were used to measure the current during the read process. The board is connected to a microcontroller with a field-programmable gate array (FPGA) chip (Xilinx, Spartan 6). The command for execution is programmed by Python.

A schematic of the test board is shown in Figure 2-5. There are 4 DACs on the test board to supply voltage pulses ranging from 0V to 5.0V on the selected bottom electrode (through DAC1), the unselected bottom electrodes (through DAC2), the selected top electrode (through DAC3), and the unselected top electrodes (through DAC4), respectively. Matrix switches (Switch1, Switch2) allocate each memristive device to the corresponding DAC. To measure current through a memristive device in the array, a multiplexer (MUX) is activated to flow the current into ADC. Due to the virtual ground of the op-amp, the voltage biased on a sensing resistor (1kΩ) is measured using ADC and converted to the current value. The arrows in the schematic indicate the current path through a selected memristive device for write, erase, and read processes. The bias voltage of each DAC for each process is specified in the legend of Figure 2-5.

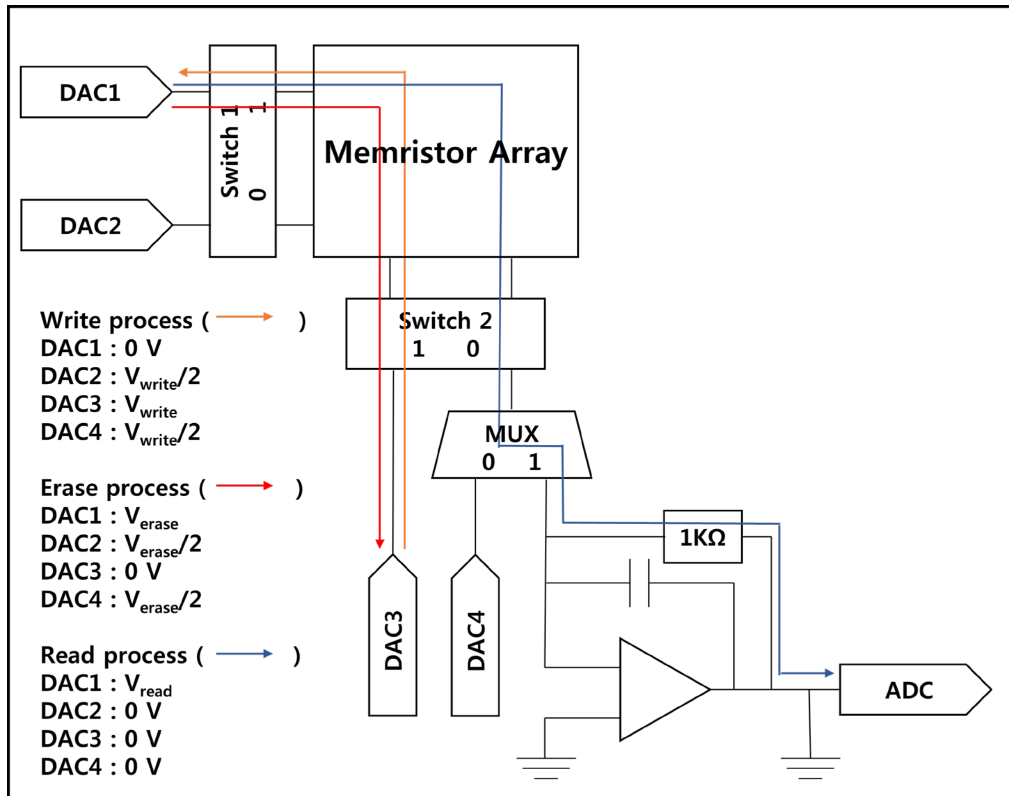


Figure 2-5 Circuit schematic of the test board (Reprinted from Ref.[40] with permission)

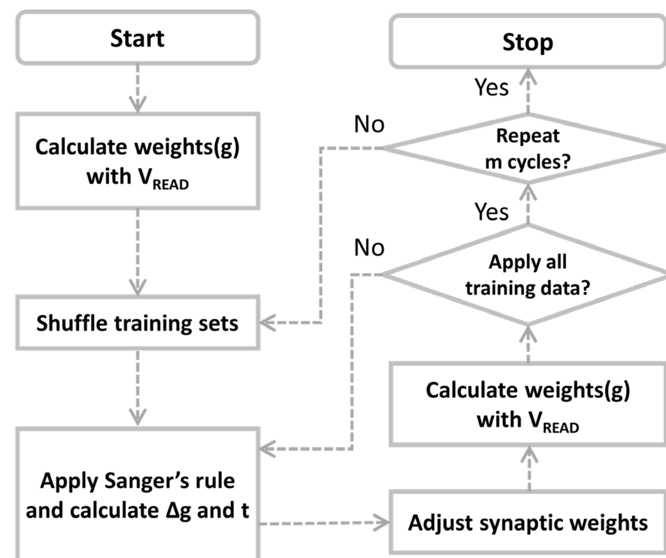


Figure 2-6 Flow chart of the PCA network operation (Reprinted from Ref.[40] with permission)

Figure 2-6 shows the flow chart of the test board to implement PCA analysis. The initial output is obtained with V_{READ}. With randomly sequenced training sets, the desired weight changes and applied pulse widths are calculated following Sanger's rule. After adjusting the memristor weights, the updated outputs are obtained using another read process. This procedure is then repeated for the 100 training data points, over the desired training cycles.

2.7 Experimental Result of Dimensionality Reduction Based on PCA

Figure 2-7 (a) shows results of the 583 test data points before the learning process. Because the weights in the memristor crossbar are initially random, mapping of the input data to the 2D output leads to randomly distributed data points, and the output data from benign cells and data from malignant cells overlap each other. In other words, without training the network does not cluster the data set effectively. Results obtained from classical PCA analysis are shown in Figure 2-7 (b). Here the principal components were obtained by directly calculating the eigenvectors of the covariance matrix, where the primary principal component was obtained in the direction of the largest variance, and subsequently the second orthogonal principal component from the second greatest variance, and so forth. Afterward, the data become clustered by transforming the input along the obtained principal components, as shown in Figure 2-7 (b).

Instead of directly solving the eigenvectors from the covariance matrix, the principal components can also be obtained through neural networks through training, using Sanger's rule (Eq. 2-3). To verify this, we first analyzed the memristor network operation through simulation. Figure 2-7 (c) shows simulation results obtained from a 9×2 memristor neural network using Sanger's rule and the dynamic device model used in Figure 2-2 (b), demonstrating successful PCA

analysis with similar results as directly solving the eigenvectors in software (Figure 2-7(b)). Experimental implementation of PCA analysis was then carried out on the 9 × 2 memristor crossbar using the test board and Sanger's rule. Figure 2-7 (d) shows PCA analysis results obtained after experimentally implementing online learning in the memristor array. Successful data clustering, which is similar to the result shown in Figure 2-7 (c), is obtained, verifying the potential of the memristor-based neural network for feature extraction tasks based on online unsupervised learning.

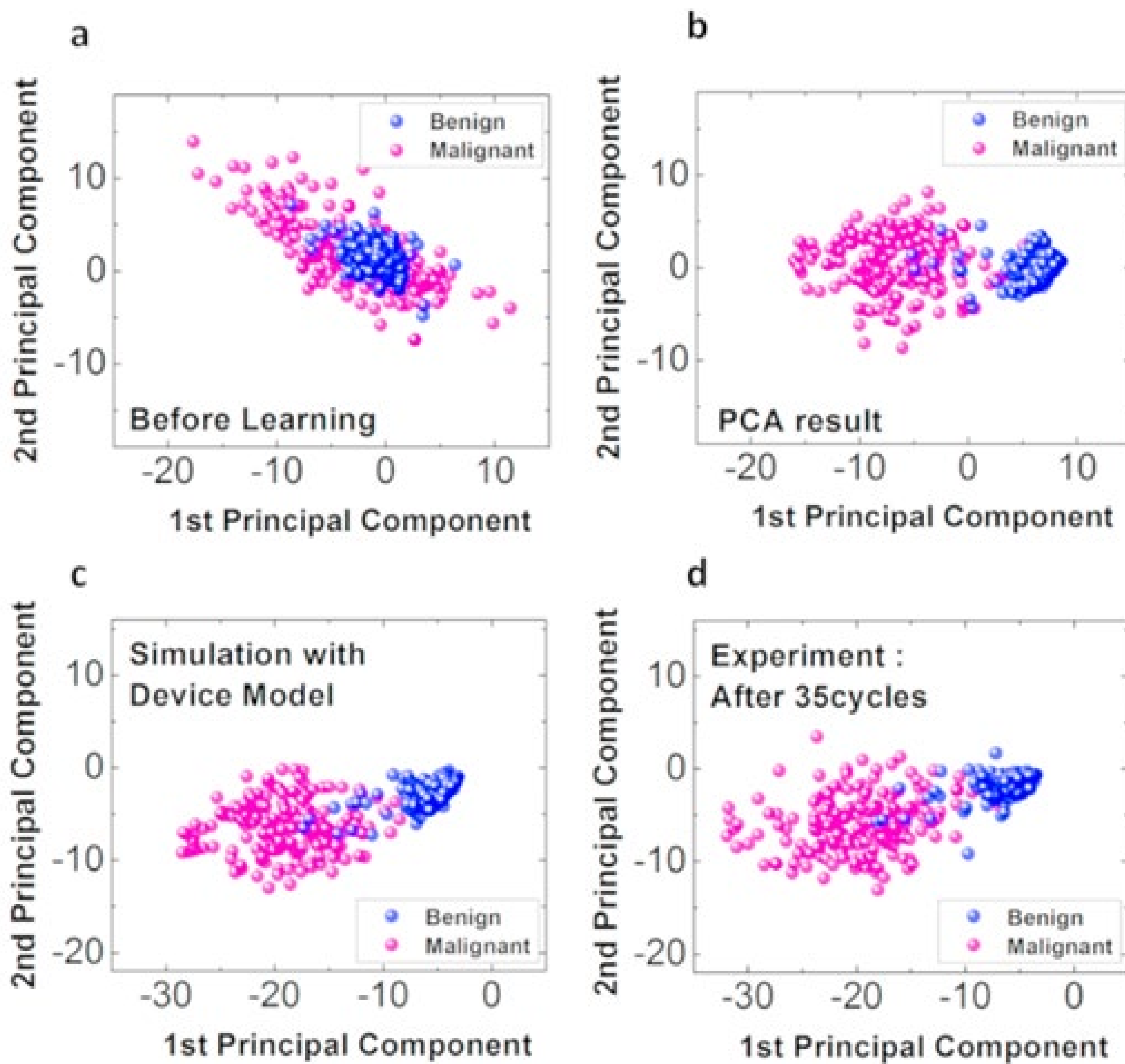


Figure 2-7 (a) Principal component analysis results. (a) Results obtained with untrained memristor network. The data are plotted based on the first (y1) and second (y2) output values. (b) Principal component analysis results after solving the traditional covariance matrix of the input data. The malignant and benign cells are largely separated into two clusters. (c) PCA results obtained from numerical simulation of the network, using the dynamic device model and Sanger's learning rule. (d) Experimental PCA results obtained from the memristor network using Sanger's learning rule after 35 cycles of training. The blue and magenta color labels in the plots represent the ground truth. Note the color labels are used only to highlight the effects of clustering in the plots but are not used in the network training or PCA analysis. (Reprinted from Ref.[40] with permission)

The primary and secondary principal components, represented by the two nine-dimensional weight vectors learned in the memristor network from the training process, are shown in Figure 2-8 (a) and (b), respectively. The black bars show the weight vector before training. The red bars and blue bars represent the weight vector after training, obtained from simulation and the experiment, respectively, by directly measuring the memristor conductance values after training. Comparing the simulation and the experimental results, obtained principal components look similar in both cases, although not identical. This can be understood from the fact that features obtained from neural networks are approximate solutions, both from the simulation and in the experiment. The specific approximate solution depends on the (random) initial condition and any device effects that are not fully captured in the model. The smaller percentage of variance represented by the second principal component (~10%) can also lead to larger differences between the simulation and experiment for the second principal component case. Nevertheless, similar clustering results can be obtained from these different approximate solutions.

Another test to verify if the neural network has performed properly is by checking the Euclidean norm of the learned feature vectors as well as the orthogonality between the vectors.

Specifically, PCA requires the feature vectors to be normalized and orthogonal. This requirement is inherently satisfied by the application of Sanger's rule, which automatically normalizes the feature vectors. Figure 2-8 (c-e) shows the measured Euclidean norm from the memristor array for the primary principal component and the secondary principal component, and their dot-product during the training process, respectively. As expected, as training continues the Euclidean norms of both vectors approach unity while the dot-product of the vectors converges to zero. These measurements verify that the learned vectors indeed become normalized during training and form orthogonal basis of the output space.

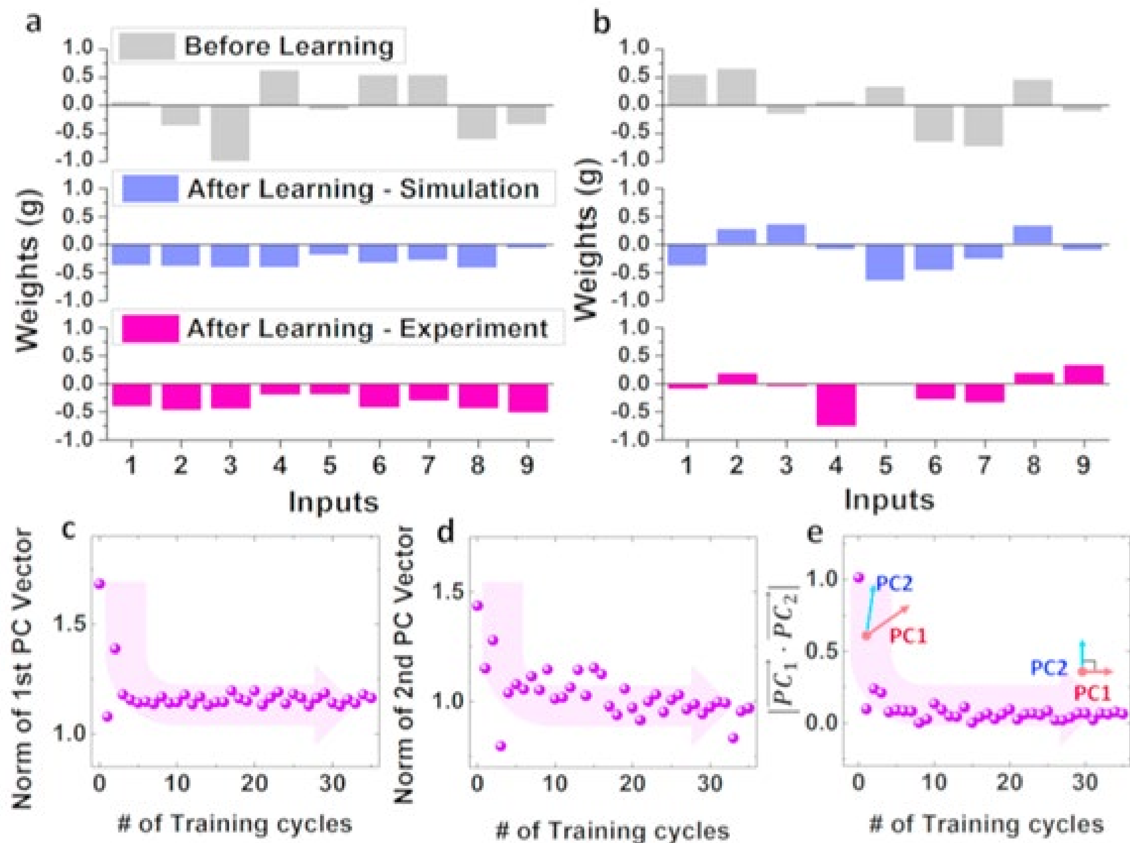


Figure 2-8 (a) Evolution of the principal component vectors. (a,b) The primary principal component (a) and the secondary principal component (b) before and after training. Black bars: memristor weights constituting the principal component vectors before the learning process. Red bars: simulated memristor weights after the learning process. Blue bars: Experimentally obtained weights after the learning process. (c,d) Evolution of the Euclidean norm of the primary principal component and the secondary principal component vectors, respectively, during training. (e) Evolution of the inner product of the primary principal component vector and the secondary principal component vector during training. (Reprinted from Ref.[40] with permission)

The clustered data, obtained from the memristor network, can then be used to implement predictive models. In this case, a decision boundary needs to be developed to separate the two clusters and predict one set as benign and the other set as malignant. The decision boundary was drawn using supervised training (based on logistic regression[33]), using the labeled training data set as shown in Figure 2-9 (a). With the help of the learned decision boundary, replotted in Figure 2-9 (b) along with the transformed data (Figure 2-7 (d)), prediction of the measurements can be accurately made. In summary, only 17 data points among the 583 test data points were misclassified, corresponding to 97.1% accuracy.

On the other hand, in classical PCA analysis, the features of a dataset can be extracted by directly solving the eigenvalue problem of the covariance matrix. Using the PCA module in Python codes, an exact solution was obtained and used for the feature extraction of the breast cancer data to compare with the results obtained from the memristor network. After clustering of the data, a decision boundary for prediction of breast cancer was calculated by fitting the training data set with the logistic regression algorithm (Figure 2-10 (a)). In Figure 2-10 (b), the test data set was

used to validate the classification result and an accuracy of 97.6% based on the exact PCA solutions obtained in software. This result is very close to results obtained by directly solving the eigenvectors in software (97.6%), suggesting that data clustering from even a small memristor-based network based on non-ideal devices can be reliably used for efficient and effective data classification based on unsupervised, online learning.

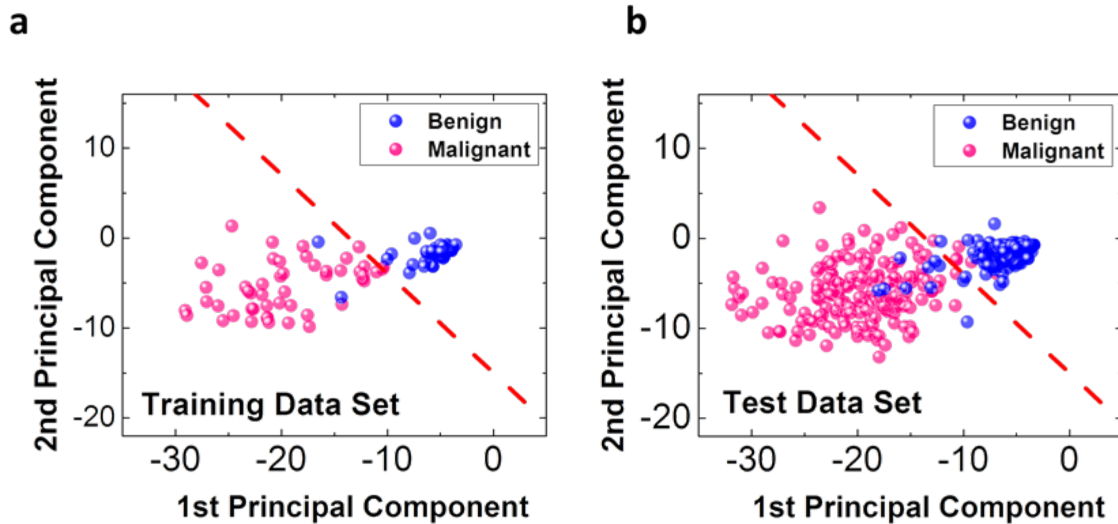


Figure 2-9 Classification based on the trained memristor network. (a) Decision boundary (purple dotted line) obtained using a supervised training process. (b) Overlay of the decision boundary obtained from (a) and the test data from PCA analysis. A cell is classified to be malignant or benign based on whether it is located to the left

or right of the decision boundary. The classification accuracy is obtained by comparing the classification with the ground truth (shown as the color label of the test data). (Reprinted from Ref.[40] with permission)

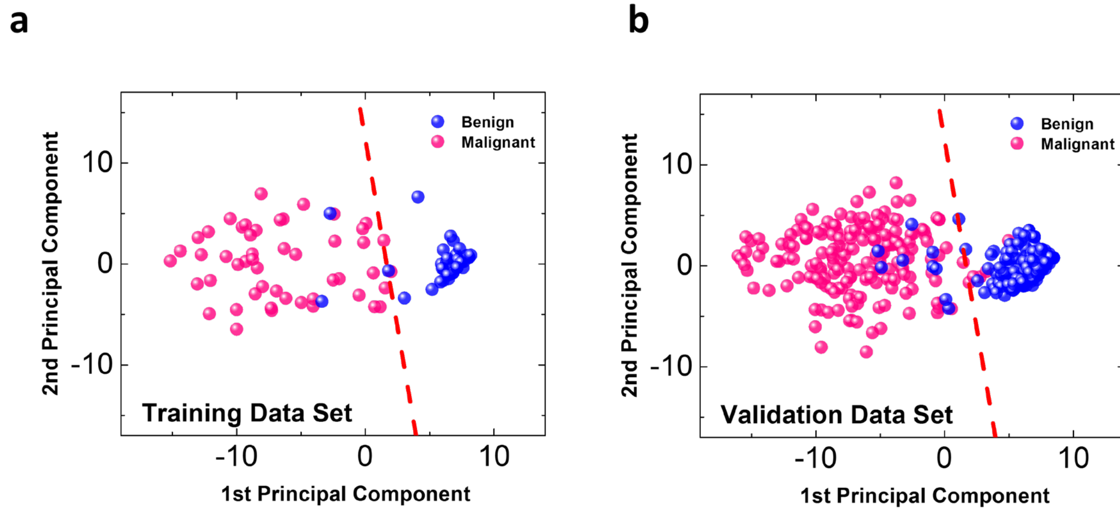


Figure 2-10 Classification results based on the exact solution of PCA. (a) Decision boundary (dotted line) obtained by fitting the training data set using logistic regression. (b) The decision boundary calculated from (a) overlaid with test data after PCA analysis. Prediction was made based on a data point's location with respect to the decision boundary. The blue and magenta color labels represent the ground truth. (Reprinted from Ref.[40] with permission)

2.8 Conclusion

In this study, we show that memristor networks can implement PCA analysis, one of the most widely used feature extraction techniques, and successfully cluster data in a real- world environment. Through online, unsupervised learning that modulates the conductance values of the memristor devices, the memristor network identifies the desired feature vectors. Our experimental studies further verify that weight normalization and orthogonality can be assured during the

unsupervised learning process. With the help of a linear classifier, the clustered data can be further used to make reliable predictions and classifications. Along with continued device optimizations and memristor-based circuit developments, this study represents a significant step toward the implementation of efficient neuromorphic hardware framework for data-intensive computing applications.

Chapter 3. Self-Limited and Forming-Free CBRAMs With Double Al_2O_3 ALD Layers

3.1 Introduction

Digital conductive-bridge random access memory (CBRAM) devices have been considered as future non-volatile memory solutions for applications such as storage class memory.[19] These devices have shown superior performance such as fast speed, low power consumption, high density ($4F^2$), high on/off ratio (>100) and CMOS-compatible fabrication that enable 3D integration process.[20]

Recently, there have been further suggestions to expand the application of digital CBRAMs for novel computing architectures. For example, CBRAM memory cells can be used to both store data and process logical or arithmetic instructions.[23], [24] In an approach termed field programmable crossbar array (FPCA), a reconfigurable computing systems for different tasks can be efficiently implemented using RRAM crossbar arrays in a modular fashion. In the FPCA system, an RRAM crossbar array can be exploited for versatile operations such as multi-bit arithmetic operations, data manipulation, vector-matrix inner product, and neuromorphic computing. Instructions listed above can be directly performed in FPCA without fetching large size of data to a processing unit to avoid the memory-wall problem. However, to realize systems such as FPCA, development and fabrication of CBRAM crossbars with uniform and robust cells is critical, and several requirements have to be met. Firstly, forming-free CBRAM devices are essential for crossbar implementation because the forming process can cause severe damage to the unselected

cells and make device variations worse.[61] In addition, self-limited programming is a desirable property because the overshoot of programming current could break down devices and also aggravate the cycle-to-cycle and device-to-device variations.[62] To reduce the effect of the sneak-current problem, low programming current and nonlinear I-V characteristics in the low resistance state (LRS) is also required.[63], [64] Although problems such as high forming voltage, over-programming, and sneak-currents can be addressed by using the one transistor and one resistor scheme (1T-1R), the large size and complex control sequence of the 1T-1R structure make the crossbar array inefficient.[65], [66] Moreover, high endurance and long retention time are also important properties for non-volatile memory applications and need to be obtained simultaneously with the other performance metrics.[67]

To meet all the requirements above, Cu-based CBRAM devices with a bilayer structure, Cu/CuO_x/ Al_2O_3(switching) /Al_2O_3(barrier)/Pd, has been developed in this study.[68] In the device structure, the ALD barrier layer serves roles as both an ionic diffusion barrier and an electronic tunneling barrier that lead to robust, uniform, and self-limited switching behaviors. The suggested CBRAM with double ALD layers (D-ALD) achieved self-current limited programming(<100nA), forming-free switching ($V_{forming} = V_{SET} = 3V$), large on/off ratio (~100), uniform cycle-to-cycle and device-to-device operation, reliable switching cycle (>1000 times), ultra-low operation current (<100nA for programming and 100pA for reading) and nonlinear I-V characteristics, using a CMOS compatible process. Optimized programming pulse scheme has also been developed to improve the operations of the D-ALD crossbar structure.

3.2 Device Fabrication

The Cu(bottom electrode)/CuO_x/LT-ALD/HT-ALD/Pd(top electrode) cells were fabricated in a $2\mu m \times 2\mu m$ two-terminal crossbar structure as shown in Figure 3-1(a). The bottom electrode (BE) was defined on top of a SiO_2/Si substrate by photolithography, followed by e-beam evaporation of 5nm/50nm of NiCr/Cu. The Cu BE was then subjected to O_2 plasma treatment for 2 minutes at 150°C to form a copper oxide CuO_x layer. 40 cycles (~45Å) of Al_2O_3 LT-ALD switching layer was then deposited in an Oxford ALD tool at 110°C using H_2O recipe, using $Al(CH_3)_3$ and H_2O as metal precursor and oxidizing agent, respectively. The ALD tool was then heated up to 250°C, and 6 cycles (~7Å) of Al_2O_3 HT-ALD layer was deposited with plasma recipe using O_2 plasma as oxidizing agent. The Pd top electrode (TE) was formed by e-beam evaporation. The devices were measured by a Keithley 4200 semiconductor parameter analyzer. All switching characteristics were obtained without external current compliance (CC).

In Figure 3-1(a), the D-ALD device with Cu/CuO_x/LT-ALD/HT-ALD/Pd structure is illustrated. The CuO_x layer was found to improve nucleation during growth of the LT-ALD switching layer. Additionally, the Cu ions in the CuO_x layer facilitates the redox processes involved in conducting filament (CF) formation and thus reduces $V_{forming}$.[69], [70] The thin HT-ALD layer is used as a barrier layer to limit CF growth and reduce the programming current. X-ray reflectance measurements (XRR) show the HT-ALD film has higher density (3.7g/cm^3) than the LT-ALD film (3.3g/cm^3). Resistive switching in HT-ALD was found to be more difficult than in LT-ALD, possibly because of the denser film makes it harder for the inclusion of water molecules to mediate Cu redox processes.[71] As a result, the HT-ALD film can act as a good Cu diffusion barrier layer.

3.3 Self-Limited and Forming-Free Resistive Switching

Figure 3-1(b) shows three consecutive resistive switching curves without CC, starting from the as-fabricated state, with voltage applied on the Cu BE and the TE grounded. Note the non-zero crossing during the negative sweep was caused by a small (~0.15pA) offset current, likely due to discharging from parasitic capacitances as the voltage is decreased. Smooth I-V curves and zero-crossing can be obtained by removing this small current offset. The first I-V sweep from the as-fabricated device leads to a switching to LRS at ~3V. This $V_{forming}$ is the same as the switching voltage in subsequent sweeps, showing forming-free characteristics. The self-limited programming process can be observed from the reliable switching curves - the switching does not lead to uncontrollable ramp up of the current, even without any external CC or added series-resistance to limit the applied voltage. Compared with previous studies on Al_2O_3-based CBRAM that require high $V_{forming}$ ~2.3×V_{SET} and external CC,[64] the forming-free and self-limited programming of D-ALD devices allow more reliable device operation and transistor-free 1S1R crossbar implementations, where low-forming voltage is necessary and external current compliance will be difficult to be applied. The I-V curves obtained from the resistive switching cycles are similar to each other, with consistent LRS and high-resistance state (HRS) values, showing good C-to-C uniformity without elaborate control circuitry. I-V nonlinearity NL_{read} and NL_{SET} of ~10 (defined as $NL_{read}=I(V_{read})/I(1/2V_{read})$ at $V_{read}=1.0V$ and $NL_{SET}=I(V_{SET})/ I(1/2V_{SET})$ at $V_{SET}=3.0V$) are also observed in the I-V curve in LRS. Due to the very low read current (0.1 nA) and the high on-state device resistance (> 1GΩ), the ground scheme can be successfully applied to arrays during read, while the nonlinearity NL_{SET} becomes attractive during write to help reduce the power consumption using the V/2 write scheme during array operation.[72], [73]

The desirable performance can be attributed to the role of the layers, CuO_x, LT-ALD, and HT-ALD in the device. Figure 3-1(c) shows forward I-V sweeps obtained from three as-fabricated control devices with different structures until their breakdown. Cu(TE)/LT-ALD(50 cycles)/Pd (BE) devices without CuO_x was used to investigate the role of the CuO_x layer. It showed not only high $V_{forming}$ ~3V, but also large D-to-D variations (gray curves) and low yield (5 out of 10 devices were initially shorted). This result can be attributed to poor nucleation of the LT-ALD layer on top of the copper BE that leads to high density of defects such as pin-holes and rough surface.[74] To address this issue, we note copper oxide has been used to improve nucleation of the Al_2O_3 layer during ALD growth since hydroxyl (OH) groups can be chemisorbed onto the CuO sites and react with the organometallic precursors during ALD.[75]–[77] Hence, Cu(BE)/CuOx/LT-ALD(50 cycles)/Pd (TE) devices with the LT-ALD layer grown on top of CuO_x were fabricated. The formation of CuO_x was conducted by 2 minutes of exposure to O_2 plasma. The LT-ALD devices grown on CuO_x layer show a low $V_{forming}$ close to 2.0V (red curves) with enhanced uniformity and high yield (all the pristine devices are initially insulating). Moreover, copper ions (Cu^{2+}) can be more easily supplied from CuO_x and diffuse into the Al_2O_3 layer compared to the Cu metal case, due to the lower Cu-O bond energy in CuO (~1.5eV) compared with that of Cu-Cu metallic bond (~2.0eV).[70], [77]–[79] Therefore, devices with CuOx as the ion source layer exhibit reduced Vforming. The low forming voltage of LT-ALD can also be attributed by the effects of H_2O molecules in the LT-Al_2O_3 film, which has been shown to be more hydrophilic than SiO_2.[80] The H_2O molecules absorbed in the LT-ALD layer can facilitate the Cu redox processes, as has been observed in SiO_2-based devices.[69]

The HT-ALD control devices were fabricated in the same structure as the LT-ALD device, i.e. Cu(BE)/CuOx/ HT-ALD(50cycles)/Pd(TE), by simply replacing the ALD growth condition

from 110°C with H_2O in the LT-ALD device to 250°C with O_2 plasma in the HT-ALD device. Since the same number of ALD cycles (50 cycles) for both the HT-ALD device and the LT-ALD device were used, resulting in similar thickness of ~55 Å, the difference of I-V the characteristics between LT-ALD devices and HT-ALD devices are mainly due to the ALD film quality. The I-V curves of HT-ALD devices (blue lines) in Figure 3-1 (c) showed very low leakage current (~10^{-14}A) and eventually broke down at a high voltage, ~ 5.5V. These results verify that the HT-ALD layer can offer very low leakage current and support high electric field. The closely packed Al_2O_3 in HT-ALD films makes it harder to contain water molecules, which have been shown to reduce the activation energy of copper ion migration.[71] These results suggest that HT-ALD can be used as an effective ion diffusion barrier to control the growth of the CF. The excellent insulating quality of the HT-ALD layer is also useful to limit the current and achieve self-compliance during the switching process, as have been verified in the D-ALD devices shown in Figure 3-1(b).

The conduction mechanism can be further explained by fitting the I-V curves, as shown in Figure 3-1(d). The I-V curve in HRS was fitted to Frenkel-Poole model in the 1.2~2.0V range (where the current is clearly above the measurement resolution limit), which implies trap-related leakage mechanism in HRS, as shown in Figure 3-1(d).[81] The LRS regime can be fitted with direct tunneling, implying the existence of a barrier between the CF and the inert electrode.[82] From the fitting, the tunneling gap and diameter of the CF is estimated to be 10Å and 16 Å, respectively. The tunneling gap of 10Å is close to the thickness of the HT-ALD barrier, ~7 Å. This analysis supports that the vertical growth of the Cu CF is stopped by the HT-ALD layer, and conduction in the LRS is dominated by tunneling through the barrier layer.

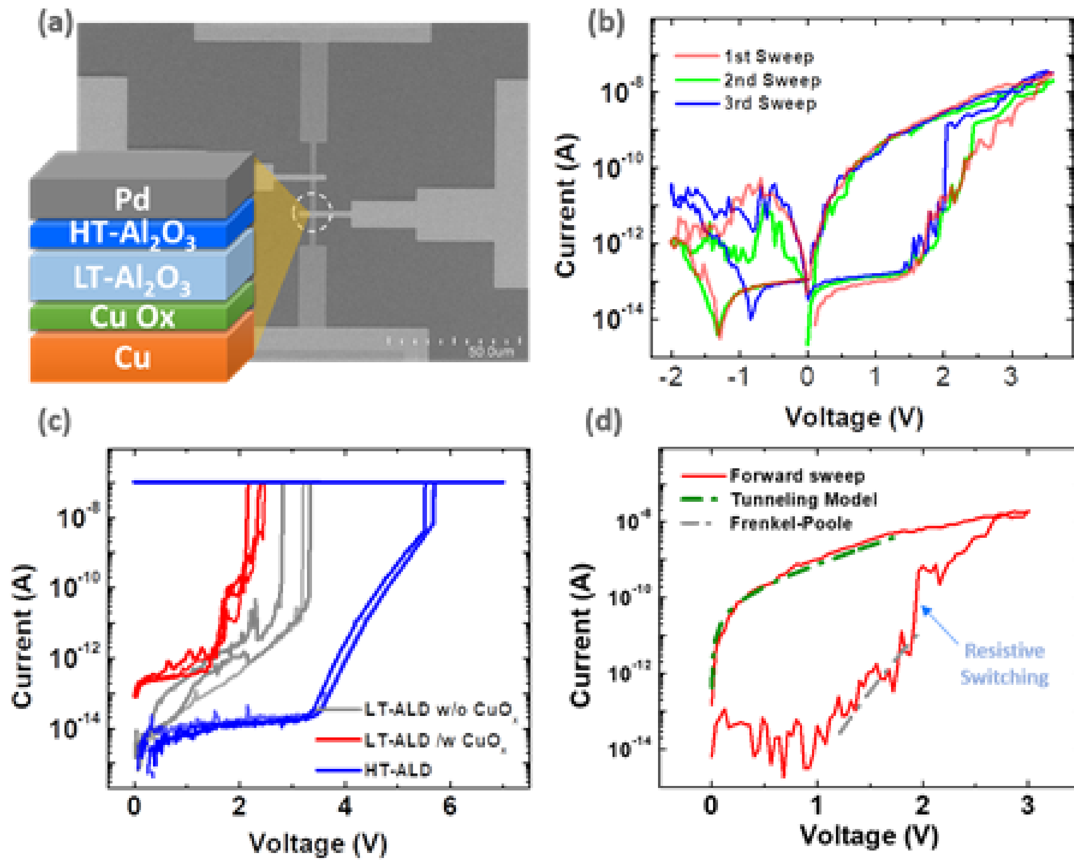


Figure 3-1 (a) SEM image of Cu/CuO_x/LT-ALD/HT-ALD/Pd crossbar devices. Inset: the schematic structure of D-ALD devices. (b) Successive I-V sweeps including SET switching of as-fabricated device (red) (c) I-V curves of LT-ALD devices without CuO_x ($V_{forming}$~3V), LT-ALD devices with CuO_x ($V_{forming}$~2V), and HT-ALD devices ($V_{forming}$~5.5V). Results from three devices are shown for each structure. (d) Fitting of the D-ALD device I-V. The fitting was limited to below 2.0V where resistive switching occurred. (Reprinted from Ref.[68] with permission)

3.4 Role of HT-ALD barrier layer

Additional evidence of reliability of the HT-ALD barrier can be observed in Figure 3-2(a). Here the applied voltage was continuously applied beyond the typical programming voltage range of 3V. At ~ 4V, Cu ion injection into the HT-ALD barrier layer occurs, leading to abruptly increased current. However, the device recovers the lower LRS current level at ~1.6V during reverse sweep without suffering from permanent damage, suggesting the partial filament in the HT-ALD layer is not stable and the HT-ALD film can restore its barrier property even if the device is accidentally exposed to high voltage. The instability of the Cu filament in the HT-ALD layer can be understood from the enhanced mechanical stress due to the high Young's modulus in the HT-ALD layer.[83], [84] Enhanced mechanical stress reduces the activation energy for Cu cluster dissolution and makes the CF volatile in the HT-ALD layer.

The resistive switching mechanism of the D-ALD device is illustrated in Figure 3-2(b). (1) In the as-fabricated device, Cu ions can be supplied from the CuO_x layer and migrate into the LT-ALD layer leading to resistive switching with forming-free behavior (2). The growth of the Cu CF in the LT-ALD layer increases the current level from ~10^{-12}A to ~10^{-9}A. The vertical growth is stopped by the HT-ALD layer. (3) Afterwards, lateral CF growth results in gradual increase in device current when the voltage is continuously applied. (4) If the device is accidentally subjected to high voltage, Cu ions may be injected into the HT-ALD layer. However, the injected Cu will not be stable and the HT-ALD layer can recover its barrier property.

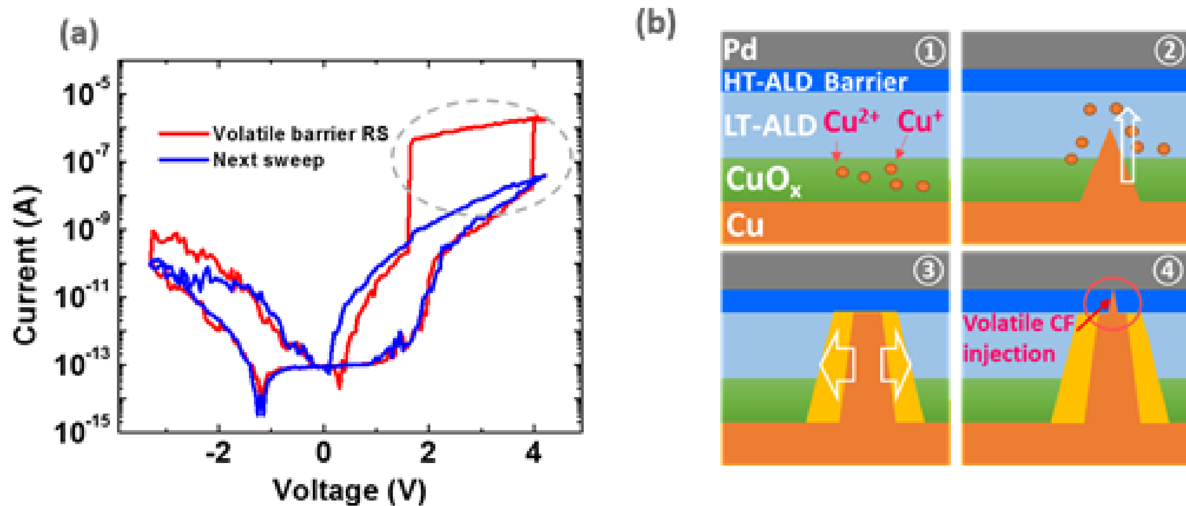


Figure 3-2 (a) An I-V sweep with larger voltage range showing over-programming (red) and the subsequent restore to the original resistive switching curve (blue) (b) Schematic illustration of the switching mechanism of D-ALD device. (Reprinted from Ref.[68] with permission)

3.5 Cycling Test Results

Figure 3-3(a) shows the cycling test results. The D-ALD device achieved 1000 cycles of resistive switching without any observable degradation of on/off ratio (~100). The C-to-C cumulative distribution in Figure 3-3(b) and its inset show uniform on-current in the 100~300pA range (at 1V), without any help of CC. D-to-D uniformity was examined in Figure 3-3(c). Box plots measured from 10 different devices show similar distribution in current levels in both LRS and HRS for all devices. The improved uniformity of the devices can be attributed by the uniformity of the deposited ALD layers, including both the LT-ALD switching layer and the HT-ALD barrier layer. In addition, retention over 10^4 seconds was obtained at 100°C (Figure 3-3(d)).

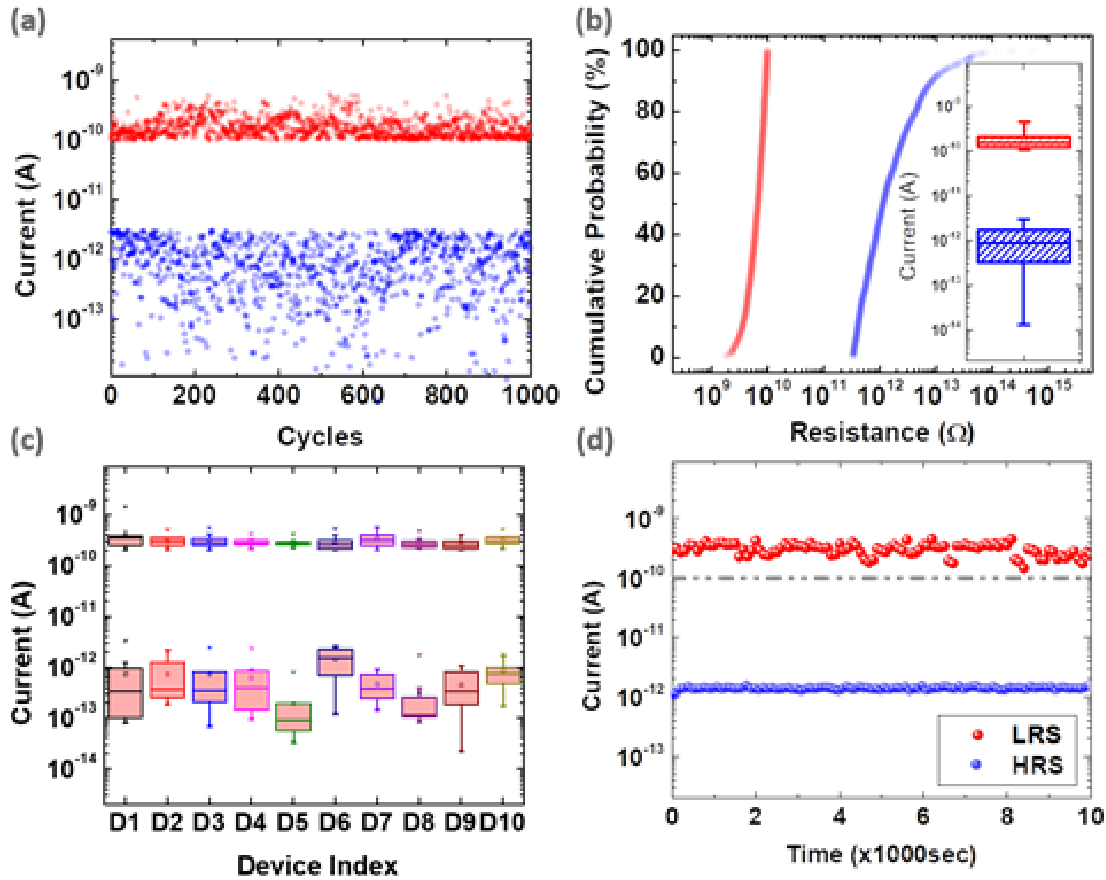


Figure 3-3 (a) Pulse cycling test (3.0V/5ms for SET pulse, -2.5V/5ms for RST pulse, and 1.0V for read pulse) (b) Cumulative probability of resistance in LRS and HRS. Inset: the box plot that summarize C-to-C variation. (c) Box plots of on/off current from 10 devices (d) retention test of LRS and HRS at T = 100°C. (Reprinted from Ref.[68] with permission)

3.6 Optimization of Pulse Programming Method

Finally, we examined the effects of the pulse programing algorithms on device operation. The standard pulse programming method is the cumulative fixed pulse (C-FP) mode, a commonly-used write-verification method, consisting of 3.0V/5ms SET pulse followed by 1.0V read pulses for verification during SET, and -2.5V/5ms RESET (RST) pulses and subsequent read pulses during RESET. (Top panel of Figure 3-4(a)) To optimize device operation, the RESET sequence

was changed following.[85] When the device was not erased after an RST pulse, a positive voltage pulse (pre-SET pulse) (2.0V/5ms) was applied prior to the subsequent RST pulse (Bottom panel of Figure 3-4(a)). This programming method is called non-cumulative fixed pulse (NC-FP) mode.[85] Compared with C-FP mode, the application of the NC-FP mode lowered the switching failure rate, defined as the probability of resistive switching not being successful after a single SET or RST pulse, as shown in Figure 3-4(b) and improved the endurance from ~200 to >1000. It is believed that the pre-SET pulses help the Cu atoms escape from meta-stable trapped locations that may favor forward migration vs. backward migration. The reduction of the gap size due to the pre-SET pulse also increases the field during the subsequent RESET pulse and facilitate the subsequent removal of the Cu atoms. This approach additionally improves SET reliability, likely due to reduction of residual Cu ions in the LT-ALD layer and the reduced stress from the reduced number of RST attempts.

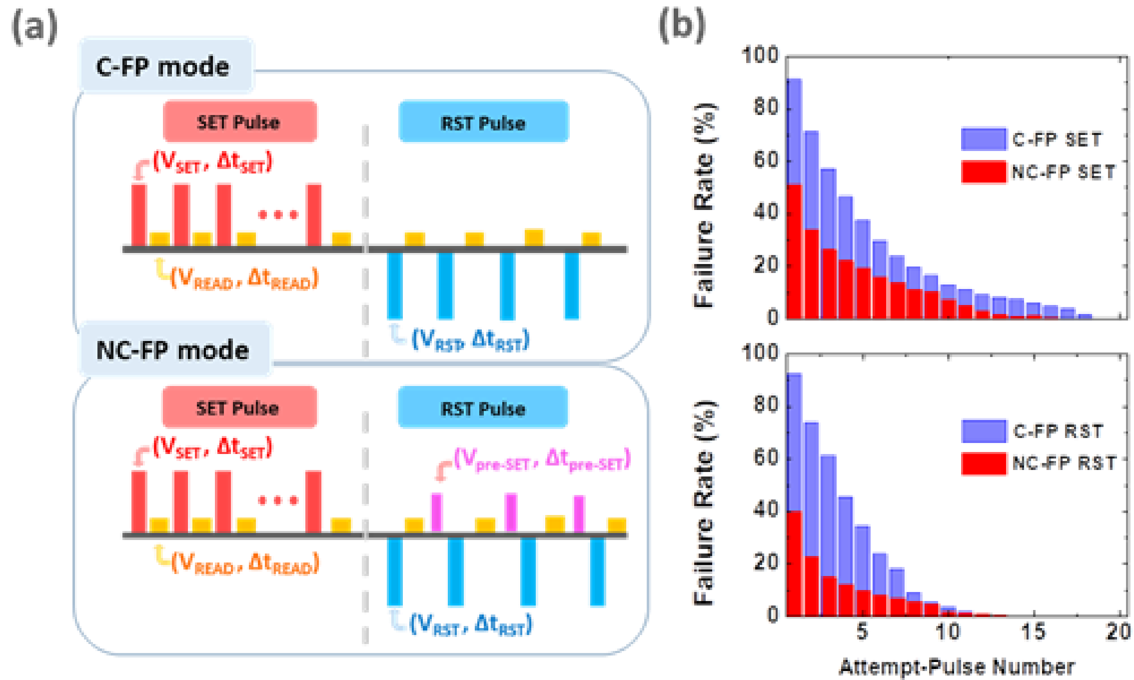


Figure 3-4 (a) Schematic pulse sequence consisting of SET pulse train and RST pulse train with verification scheme, for the cumulative fixed pulse mode (C-FP) and the non-cumulative fixed pulse mode (NC-FP). (b) Switching probability of SET (upper) and RESET (lower) for attempts using C-FP (purple) and NC-FP (red). The conditions of SET, RST, pre-SET pulses are fixed at 3.0V/5ms, -2.5V/5ms, and 2.0V/5ms, respectively. (Reprinted from Ref.[68] with permission)

3.7 Conclusion

In this project, Cu-based CBRAM devices with double Al_2O_3 ALD layers have been developed. The D-ALD devices achieved self-limited current, forming free, high on/off ratio, good uniformity, nonlinear I-V at LRS, and robust pulse switching. The roles of CuO_x and HT-ALD layers were investigated. The promising results from the D-ALD devices will help advance of the CBRAM crossbar arrays for storage and novel computing applications.

Chapter 4. Hardware Acceleration of Simulated Annealing of Spin Glass by RRAM Crossbar Array

4.1 Introduction

Combinational optimization problems (COPs) try to find globally optimal objects in a discrete space. Difficult COPs such as spin glass systems and the traveling salesmen problem are NP-hard, i.e. at least as hard as the hardest problems in NP (Non-deterministic polynomial time) problems. To solve these problems, simulated annealing (SA), a metaheuristic algorithm that effectively search global optima, has been developed and widely used.[86] However, the convergence of SA may be slow because it involves compute-intensive operations within a massively connected interaction network and stochastic search rules that require random number generation (RNG) with an exponentially decaying probability distribution. Recently, there have been significant progress in RRAM-based acceleration of numerical computation such as solving partial differential equations and neural network implementations based on vector-matrix multiplication,[87], [88] in-memory computing,[89] in-memory and stochastic computing using stochastic bit streams.[28], [29] Inspired by the ability of RRAM devices for numerical computation, in this work, we utilized vector-matrix multiplication functions of Ta_2O_5 RRAM crossbars and stochastic switching properties of Cu-based CBRAM devices to accelerate an SA algorithm that solves a spin glass problem effectively.[90]

4.2 Spin Glass Problem and Simulated Annealing

Finding the ground state of a two-dimensional (2D) spin glass, from randomly mixed states as shown in Figure 4-1 (a), is a classical problem in COP. Although the interaction between two spins is simple such that the Hamiltonian is just a multiplication between neighboring spins weighted by the coupling strength, complex interactions between arbitrary spin pairs exist in the spin glass, as illustrated in Figure 4-1 (b) and make the problem difficult to solve in polynomial time.[91] Figure 4-2 (a) shows the flowchart of conventional SA that starts from initializing the spin configuration, followed by calculating the change of Hamiltonian ΔH due to flip of randomly selected yth single spin, σ_y. The Hamiltonian of the spin glass is given as:

$$\mathrm{H} = -\mathrm{J}\sum_{<x,y>} N_{xy}\sigma_x\sigma_y = -\frac{1}{2}\mathrm{J}\sum_{x,y} N_{xy}\sigma_x\sigma_y \qquad \text{(Eq. 4-1)}$$

where J is the amplitude of the coupling strength, σ_x and σ_y are the xth and yth spin in the spin glass. <x,y> in Eq. 4-1 indicates that the spin multiplication needs to be conducted only for neighboring spins. The introduction of N_{xy}, a coupling strength (CS) matrix, makes the expression more concise. Elements in N_{xy} are '1' if σ_x and σ_y are neighbors of each other, and '0' for non-neighboring spins. If a spin flip decreases energy, e.g. inversion of σ_y leading to negative ΔH_y, SA accepts the change because it stabilizes the spin system. If ΔH_y is positive, on the other hand, the spin flip will happen with a probability proportional to the Boltzmann factor ($\mathrm{p} = \exp\left(-{}^{\Delta H_y}/_{kT}\right)$) where T is absolute temperature. After a fixed number of attempted spin flips, the temperature T is decreased following a cooling schedule, and the process is repeated at the new temperature. The stochastic hill climbing provided by the Boltzmann factor enables the spin glass to escape from local optima

as depicted in Figure 4-2 (b), and the escape probability decrease to zero as time increases and temperature cools down.

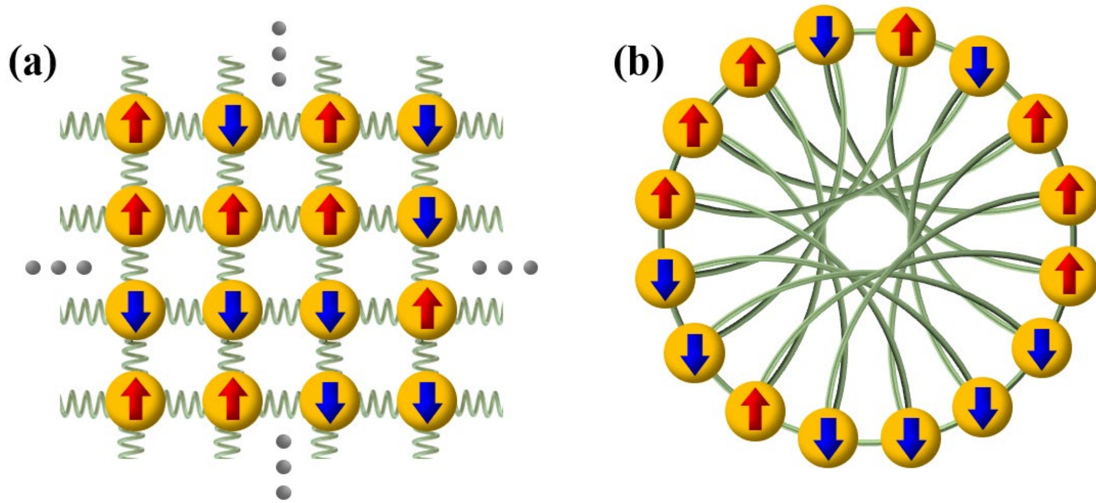


Figure 4-1 A 2D spin glass and the spin interactions represented by (a) connections to neighboring spins and (b) circular graph showing the complex couplings. (Reprinted from Ref.[90] with permission)

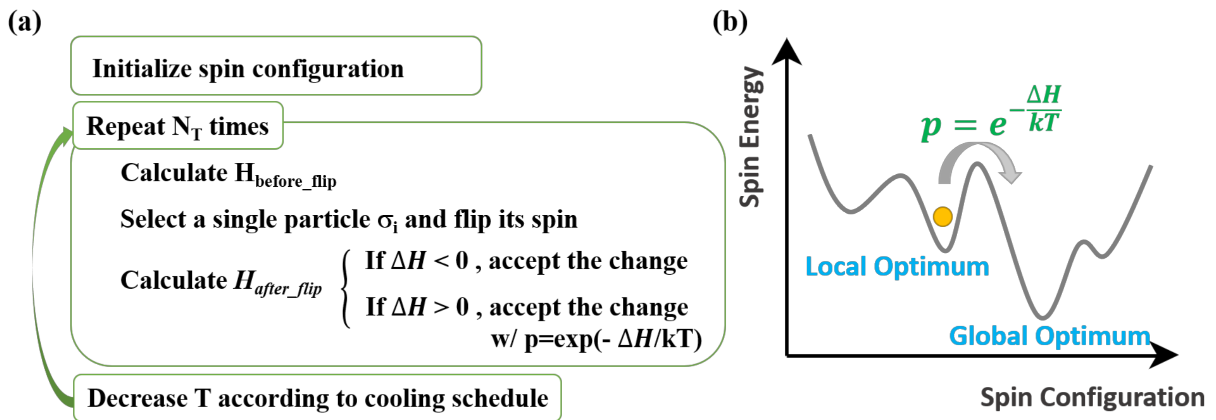


Figure 4-2 Flow chart of the SA algorithm. (b) Schematic showing finite spin flip probability even for positive ΔH can help the system escape from local optima. (Reprinted from Ref.[90] with permission)

4.3 Simulated Annealing Accelerated By RRAM Array and Stochastic CBRAM

During SA, calculations of the inner products in ΔH_y and the probability generated by the RNG function in the Boltzmann factor make the process compute-intensive. To reduce the computational cost and speed up SA, inner products between the spin vector $\vec{\sigma}$ and neighboring spins, as determined by the CS matrix, can be directly obtained in an RRAM array storing the CS matrix N_{xy}, as shown in Figure 4-3 (a), (b). For example, when the yth spin attempts to be flipped, all x^{th} row ($\forall \sigma_x \in \vec{\sigma}$) in the RRAM array in Fig 4-3 (c) are applied with a $V_x(=\sigma_x V_{read})$ pulse, and the output current I_y at the yth column is proportional to $\sum_{x,y} N_{xy}\sigma_x\sigma_y$, producing the desired value of ΔH_y. As a result, the inner-products can be readily obtained from read operations through the RRAM array.

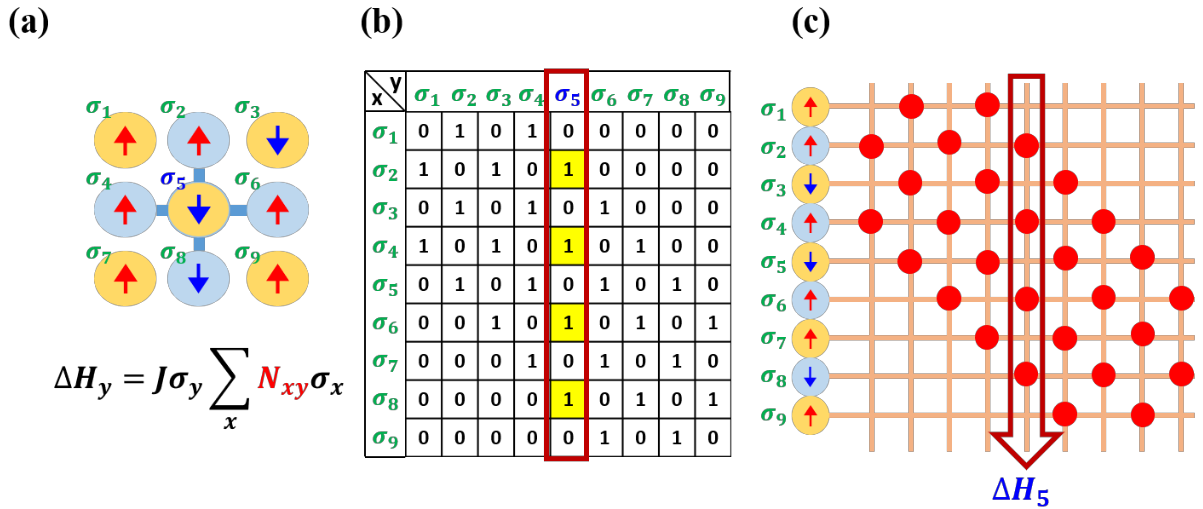


Figure 4-3 (a) ΔH_y due to the change of σ_y surrounded by its neighbor spins. (b) CS matrix where the 5th column represents interaction between 5th spin and all the other spins (c) Schematic of inner product between the 5th CS column vector and spin vector $\vec{\sigma}$ conducted by RRAM array. (Reprinted from Ref.[90] with permission)

Since only nearest neighbor interactions are non-zero, the CS matrix can be very large but sparse. The large CS matrix can be effectively mapped into smaller RRAM arrays where only the non-zero portions are stored, as illustrated in Figure 4-4. Here a 9×9 2D spin glass was chosen as an example. The 81×81 CS matrix of the spin glass represents all-to-all connection and can be divided into three groups (top-edge row, mid rows, and bottom-edge row), representing the coupling strength of a spin in the top (middle, or bottom) row with its neighbors. The groups are 9 column wide (corresponding to the 9 spins in each row) and can be further divided into sub-groups of 3 spins (3 columns), for spins at the left-edge, middle columns, and right edge, producing the patterns shown in Figure 4-5. All the possible (non-zero) sub-matrix patterns can then be stored in a three column RRAM array (11×3), as shown in Figure 4-5 (d). Experimentally, the 11×3 RRAM array was fabricated with a Pd/Ta/Ta_2O_5/Pd cell structure. The RRAM crossbar array is then wire-bonded and connected to a custom test board as shown in Figure 4-6.

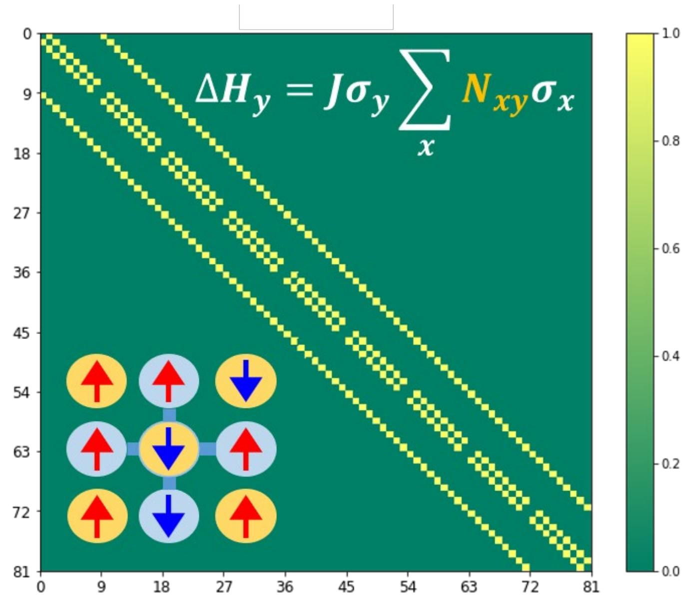


Figure 4-4 81×81CS matrix of a 9×9 2D spin array. The large but sparse CS matrix can be sliced to fit into a smaller RRAM array. (Reprinted from Ref.[90] with permission)

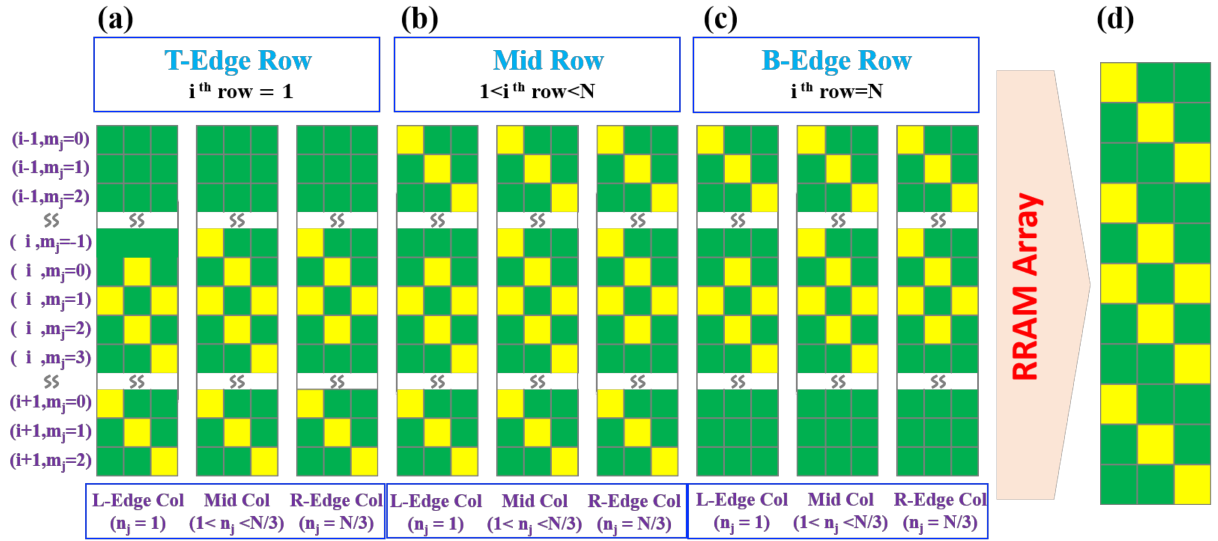


Figure 4-5 9 sub-patterns with three columns each from the 81×81 CS matrix, depending on the position of the spin in the 2D spin glass. (a) Top-Edge Row case, (b) Mid Row case, and (c) Bottom-Edge Row case. (d) All the non-zero and unique patterns in (a-c) can be stored in a single 11×3 RRAM array. (Reprinted from Ref.[90] with permission)

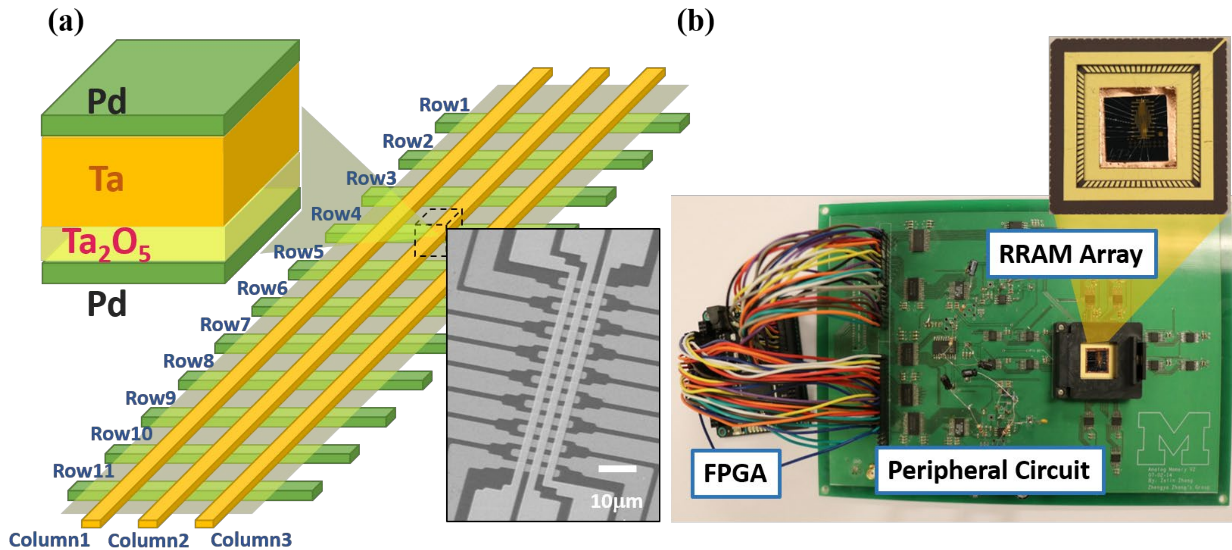


Figure 4-6 (a) Schematic of the Ta_2O_5-based RRAM cell and array structure. SEM image of the RRAM crossbar array. (b) Test board comprised of FPGA, peripheral circuit, and the RRAM array chip for experimental implementation of simulated annealing. (Reprinted from Ref.[90] with permission)

Reliable switching characteristics and tight forming, set and reset voltage distribution can be obtained from all devices in the RRAM array (Figure 4-7 (a), (b)). The cell-to-cell current variations shown in Figure 4-7 (c) can be significantly improved to be lower than 1% using a write-verify method, as shown in Figure 4-7 (d), enabling robust dot product operations to obtain ΔH_y.[24] The hill climbing probability was also obtained through hardware by using stochastic switching effects in a Cu-based CBRAM, as shown in Figure 4-8. The CBRAM device shows stochastic switching behavior switching at probability low programming voltage, with a switching probability $\mathrm{P}(\Delta t) = 1 - \exp(-\Delta t/\tau)$ for programming pulse width Δt, where τ is a time constant dependent on the voltage amplitude. A Cu/ALD Al_2O_3/Pd CBRAM structure is used in this experimental implementation, with $\tau = 24.9$ms for transition from HRS to LRS. After applying a single SET pulse, the probability of the device staying at HRS then follows the exponential decaying function $\exp(-\Delta t/\tau)$, which follows the Boltzmann factor required for SA, after converting ΔH_y to $\Delta t = \tau\left(\Delta H_y/kT(t)\right)$.

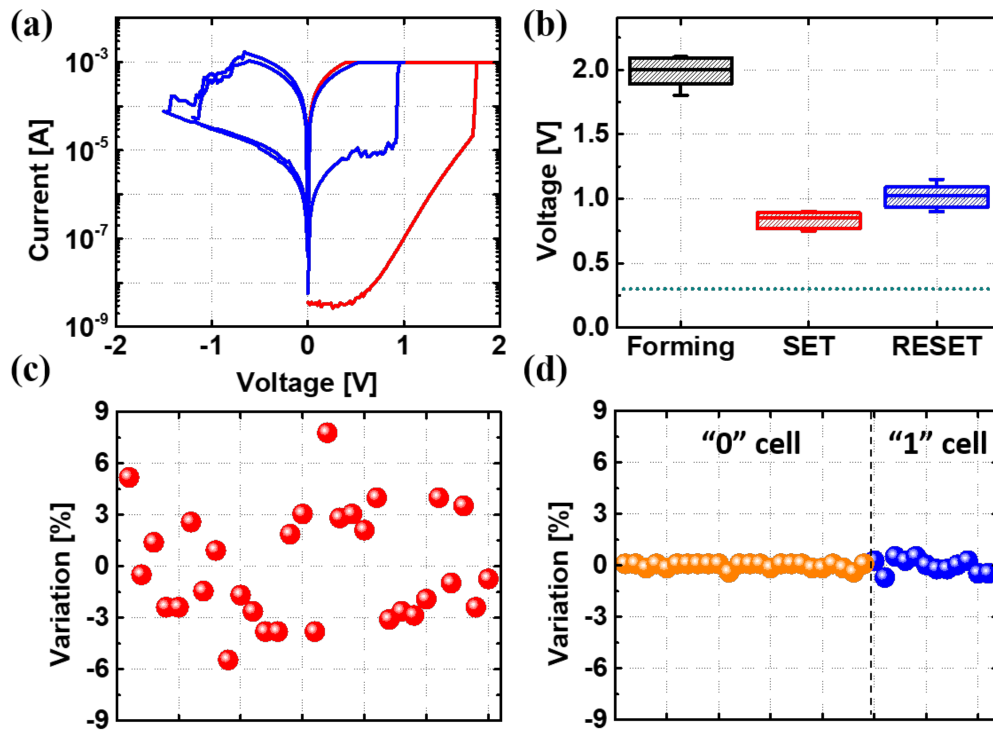


Figure 4-7 (a) I-V curves showing the forming (red) and subsequent switching (blue) processes. (b) Distribution of $V_{Forming}$, V_{SET}, V_{Reset} of the 33 cells in the RRAM array. (c-d) Variation of device current without (c) and with (d) write-verify pulse method. (Reprinted from Ref.[90] with permission)

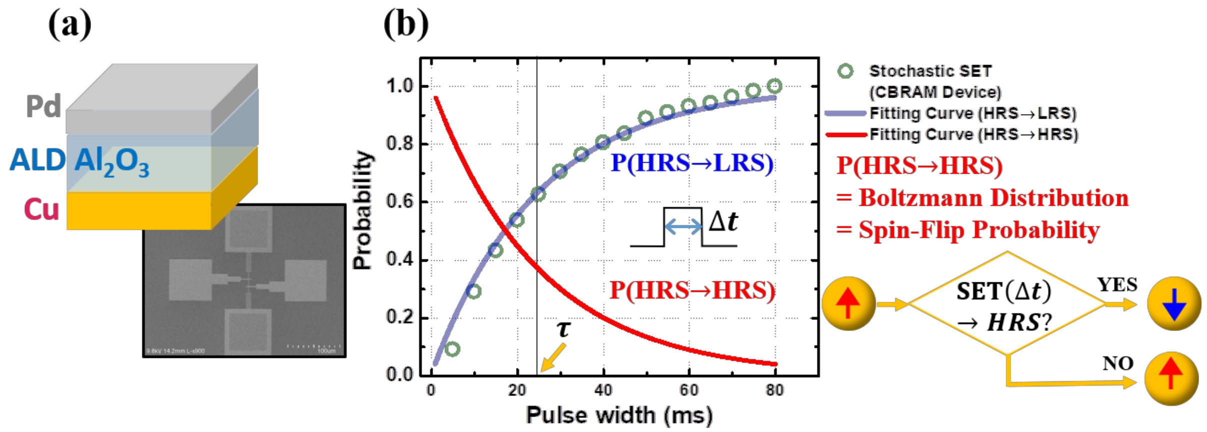


Figure 4-8 (a) Structure and SEM image of Cu-based CBRAM devices. (b) Experimentally measured probability of HRS→LRS switching (blue). The Boltzmann factor (red) can be obtained by the probability of the device staying at HRS after applying a single SET pulse with pulse width Δt. (Reprinted from Ref.[90] with permission)

4.4 Experimental Demonstration Of RRAM-Based Simulated Annealing

The flow chart of implementing SA to simulate a spin glass column in the spin glass is randomly is shown in Figure 4-9. Starting from the initial spin configuration, a spin (i^{th} row and j^{th} selected for flip-trial. The spin vector is converted as input pulse vector based on its location and applied to the 11×3 Ta_2O_5 RRAM array. After the current measurement from the selected column I_y, the sign of I_y is compared with σ_y. The flip-event of σ_y is accepted if the signs match (corresponding to negative ΔH_y. If the signs of I_y and σ_y do not match, the flip-event is only accepted if a single SET pulse on a the CBRAM does not change its original HRS state, following discussions above. The data flow is illustrated in Figure 4-10.

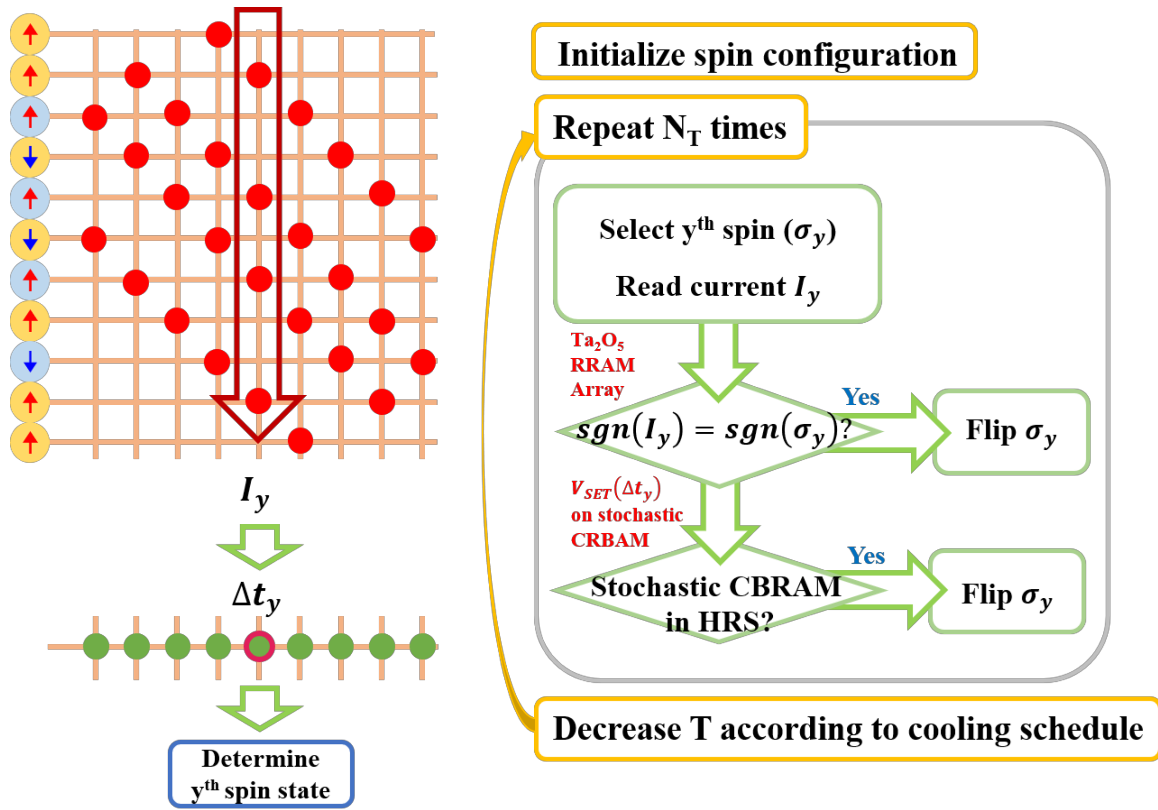


Figure 4-9 Flowchart of implementing the SA algorithm using RRAM array for the 2D spin glass problem. (Reprinted from Ref.[90] with permission)

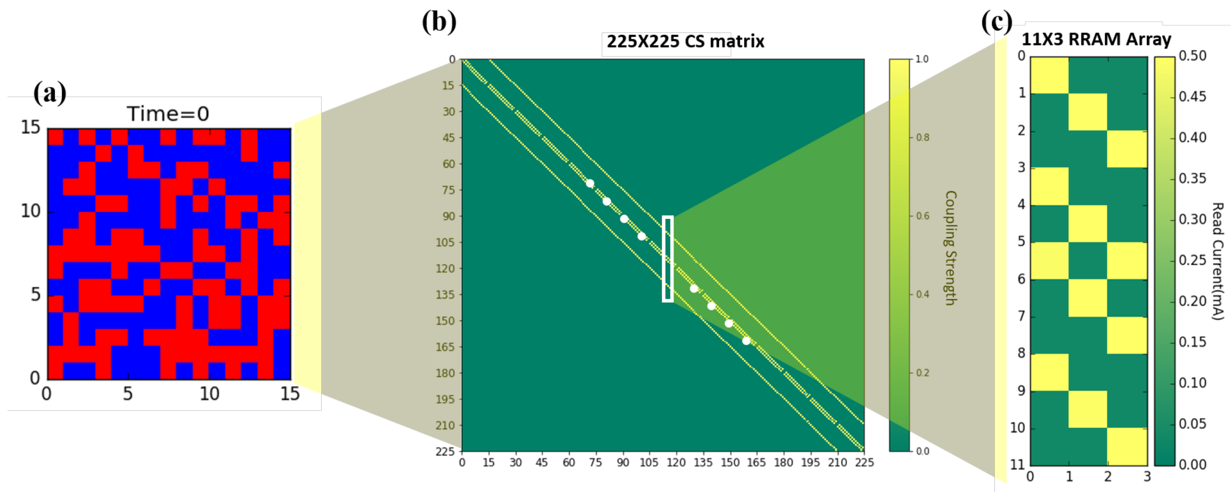


Figure 4-10 (a) Randomly initialized 15×15 spin array (with 225 spins). (b) The sparse 225×225 CS matrix. (c) Coupling strength patterns stored and measured from the RRAM array used in the experimental setup. (Reprinted from Ref.[90] with permission)

A 15×15 2D ferromagnetic spin glass was tested to prove the concept of RRAM-based SA process. Figure 4-11 shows one test case with a fixed spin edge condition, where all the edge spins are fixed at the 'up'(+1) state and the rest of the spins are initialized to 'down'(-1) state at $t = 0$. Because the edge spins are always fixed, the only possible ground state of this problem is 'all-up' configuration. The SA parameters such as J, $T(t)$, and N_T for the experiment are 1.0, $5/\sqrt[3]{t+1}$, and 100, respectively. As time flows, the initially down-spins get affected by the edge spin states due to ferromagnetic interaction that favors spins with same orientations. Note some of the down-spins surrounded by other down spins are also flipped to up-spin (e.g. at time=5), although this event increases the total energy E. This is an example of hill climbing phenomenon which can speed up the optimization process by escaping from the local optima, as discussed in SA. The ground state is achieved at time ~ 200. Other cases with multiple ground states, i.e. initially random configurations without any fixed edges, were also tested using the RRAM-based SA, as shown in Figure 4-12. Due to the existence of two possible ground states with 'all-up' and 'all-down' spin configurations, the same initial condition can evolve to opposite results, as verified by the experiments. Note that the two solutions also show similar proportions of majority spin during the evolutions (e.g. at time=150), since the SA strategy leads to similar dynamic progress towards the respective ground state. Comparison between the experimental RRAM-based SA results and software results verifies the E and magnetization (M) of both cases show similar dynamics that converge to global optima near time=200, further proving the successful experimental implementation of RRAM-based SA.

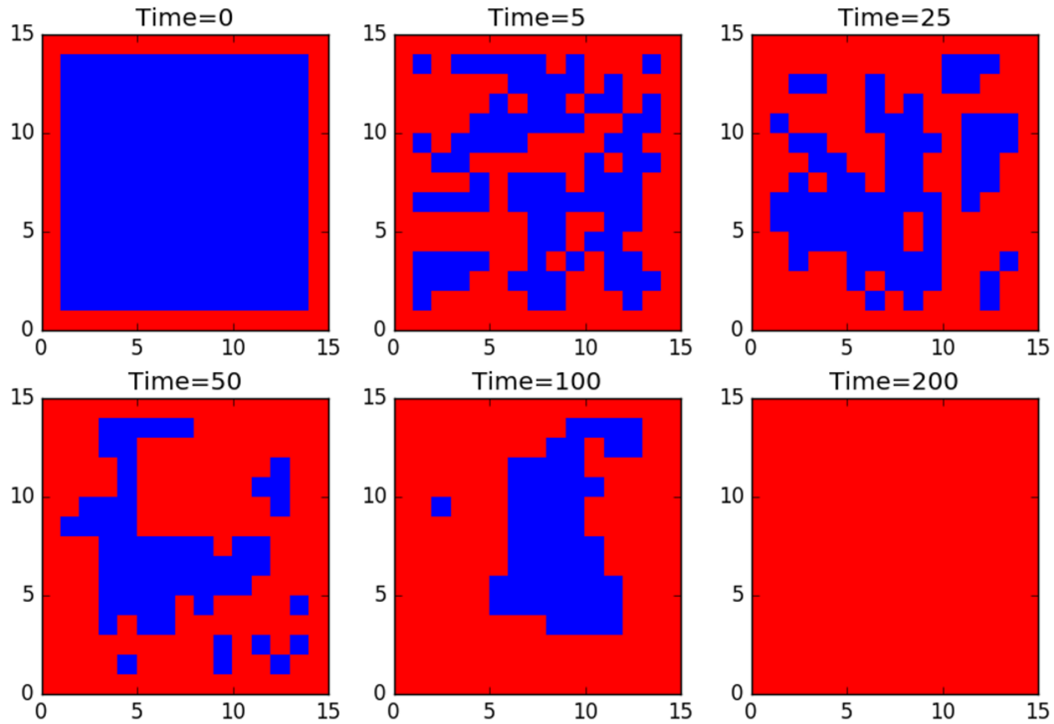


Figure 4-11 Evolution of the spin configuration at different time steps for the fixed spin-edge case. Data obtained experimentally from the RRAM array-based hardware system. (Reprinted from Ref.[90] with permission)

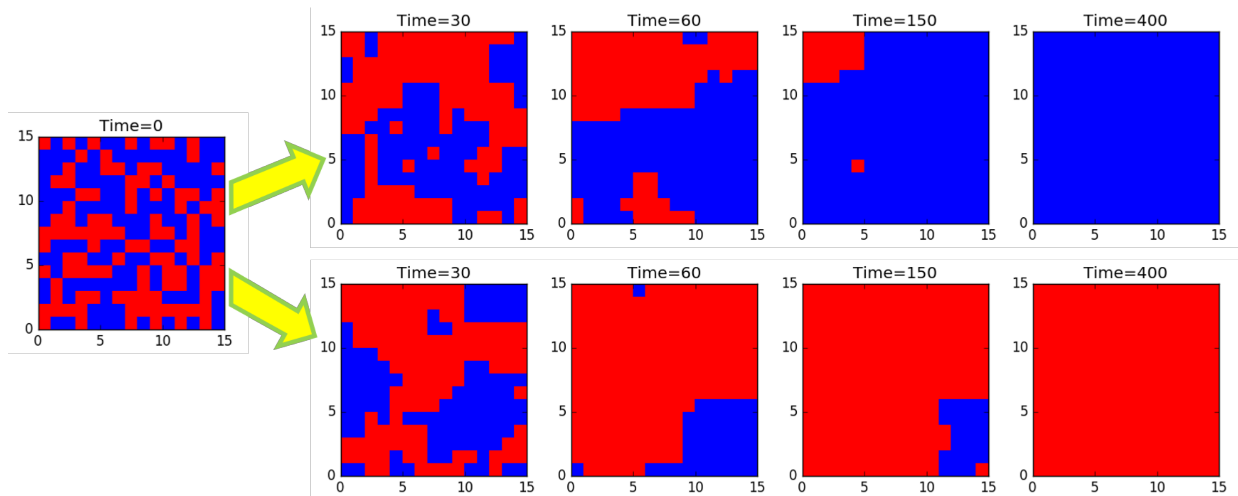


Figure 4-12 Time-dependent evolution of the spin glass system solved by the RRAM hardware, for random initial states with no fixed spins. Two ground states with global energy minima, 'all-up' state and 'all-down' states, can be generated from the same initial state in different runs. (Reprinted from Ref.[90] with permission)

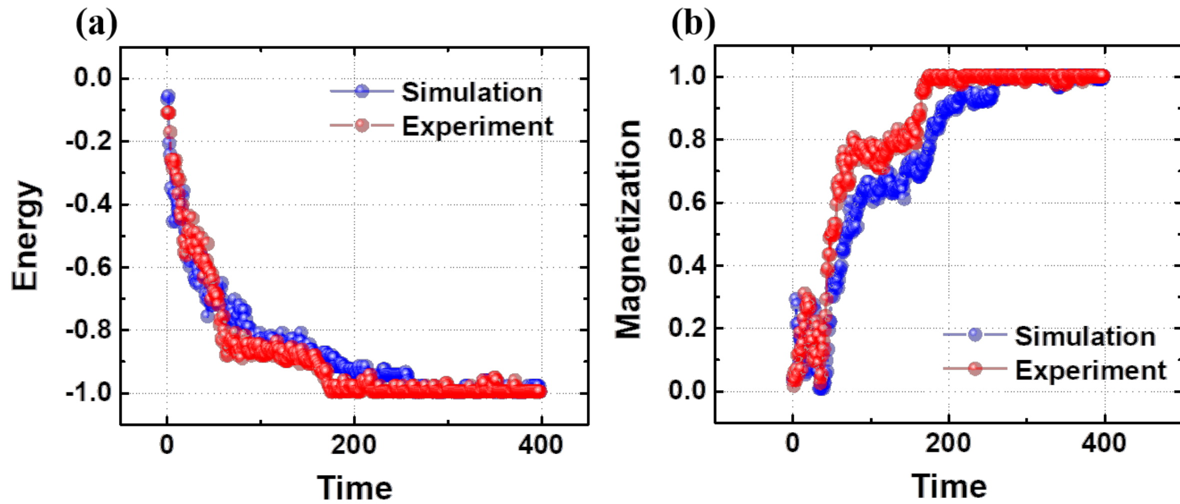


Figure 4-13 (a) Average energy and (b) magnetization as a function of cooling schedule. Conventional software version of SA (red) and experimental SA results obtained from the RRAM array (blue) are compared. (Reprinted from Ref.[90] with permission)

4.4 Parallel Spin-flip Strategy Using Memristive Simulated Annealing

To further accelerate the RRAM-based SA, it is possible to flip multiple non-neighboring spins together simultaneously to take advantage of the parallel vector-matrix multiplication (vs. vector-vector inner product) offered by RRAM arrays, as illustrated in Figure 4-14. The flipped spins must be non-neighboring to not affect the energy calculations compared with consecutive spin flips. The parallel spin-flip strategy was also implemented in the RRAM-based hardware. Comparisons of the experimental results obtained from the conventional single spin-flip and the parallel double spin-flip schemes are shown in Figure 4- 15, for the fixed edge test case. The E and M from double spin-flip scheme (red) show faster convergence than the single spin-flip scheme (blue). The single spin-flip scheme even fell into a local minimum near time=100 for a while before finally escaping, while the double spin-flip method already reached its ground state.

Since the double spin flip should be equivalent to two consecutive spin flips (at the same temperature), the results are compared with another experiment where 2 × iterations (i.e. $2N_T$=200) are attempted at each time step using the single spin flip scheme (black curves). This approach indeed produced results like those obtained from the double spin-flip experiments, and suggested possibility of further acceleration of SA with an N spin-flip scheme that can be calculated simultaneously in RRAM-based array.

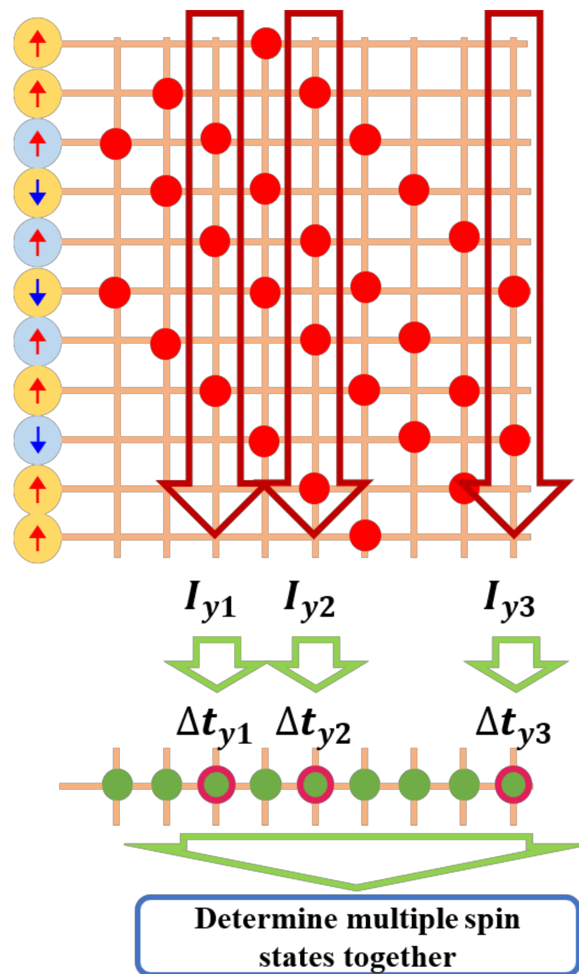


Figure 4-14 Schematic illustration of multi-spin flip method that exploits parallel vector-matrix multiplications in RRAM crossbar array. (Reprinted from Ref.[90] with permission)

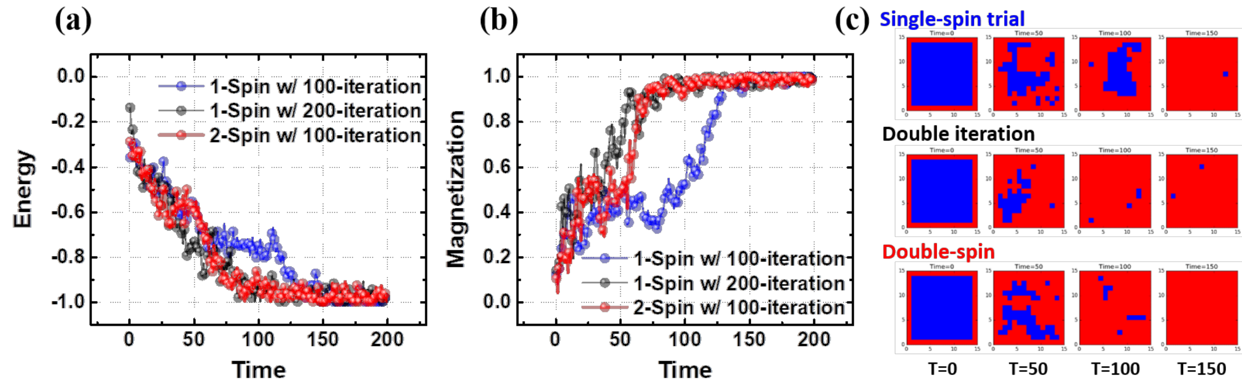


Figure 4-15 Comparison of (a) energy, (b) magnetization, and (c) spin configuration snap shots, for results obtained using the single-spin method with 100 iterations per time step (blue), single-spin method with 200 iterations per time step (black), and double-spin method with 100 iterations per time step (red). All results are obtained from the RRAM hardware setup. (Reprinted from Ref.[90] with permission)

4.5 Digital Annealing with RRAM crossbar array

Although the parallel spin-flip strategy accelerates the simulated annealing by the number of parallel spins, this method can cause error if two flipping parallel spins are the nearest neighbors to each other. As a result, the number of parallel spins cannot be larger than 50% of the number of columns in the RRAM array. Therefore, the parallel spin-flip strategy is not able to fully exploit the parallelism of the RRAM array.

Inspired by quantum annealing that examine all the possible quantum states at the same time and shrink to the ground states,[92] digital annealing with parallel pipeline using FPGA was suggested by S. Matsubara *et. al.*.[93], [94] The key idea of digital annealing is to test all-to-all interaction to try all the possible spin flip at the same time and select the most optimal spin to accelerate the spin Ising model. To implement the idea of 'parallel-trial' in digital annealing platform, ability to calculate vector-matrix multiplication needs to be maximized. Because RRAM

crossbar structure is well known as a vector-matrix multiplier,[95] not just vector-vector or multiple vector-vector operations, the parallel trial strategy is going to be a good match with RRAM crossbar array. Figure 4-16 illustrates the difference between the parallel spin-flip strategy and the parallel trial method. In parallel spin-flip strategy in left panel, only two non-neighboring columns are activated and used for calculation of changed Hamiltonian. After the estimation of Hamiltonian, both spins are flipped. On the other hand, the parallel trial method exploits all the columns in the RRAM crossbar array to find the best choice. The advantage of the parallel trial method comes from that the array use its maximal parallelism to optimize the spin configuration.

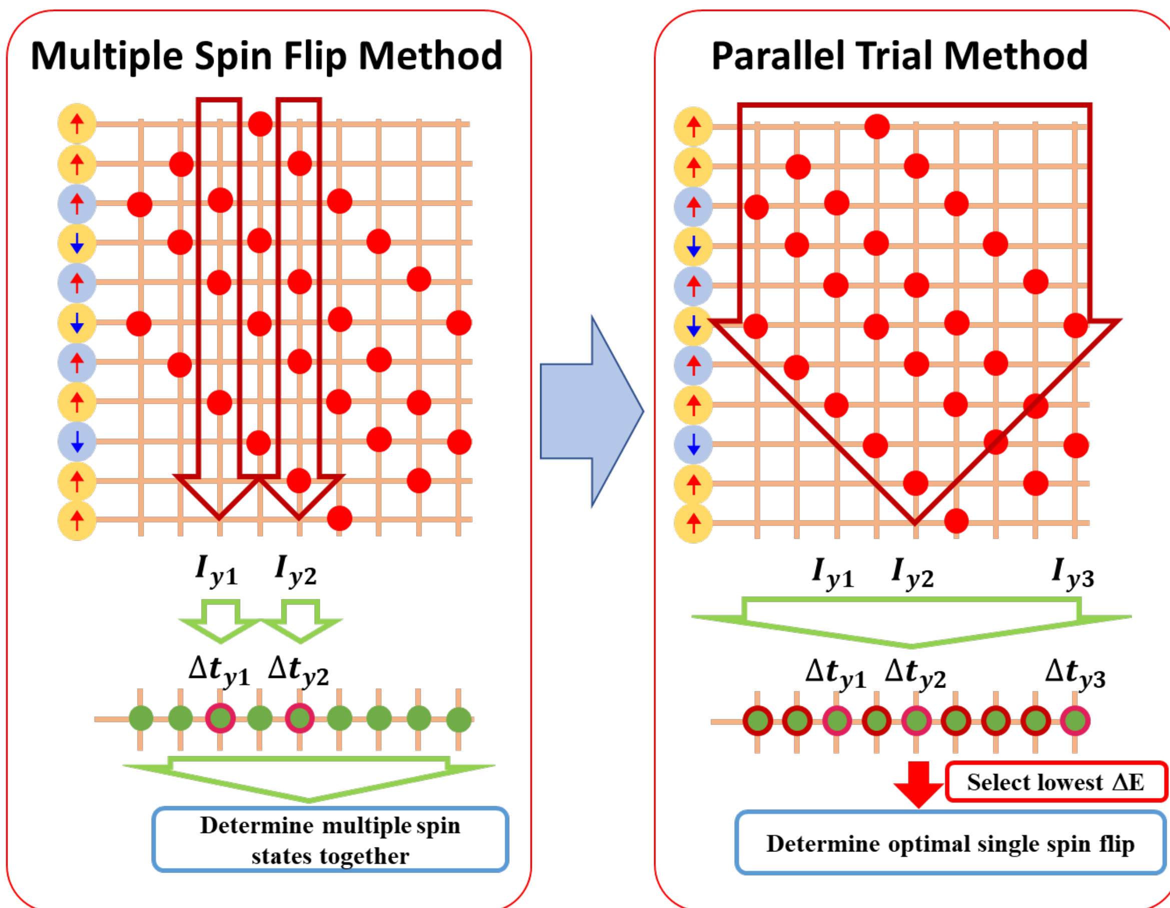


Figure 4-16 Comparison between multiple spin flip strategy and parallel trial strategy (Reprinted from Ref.[90] with permission)

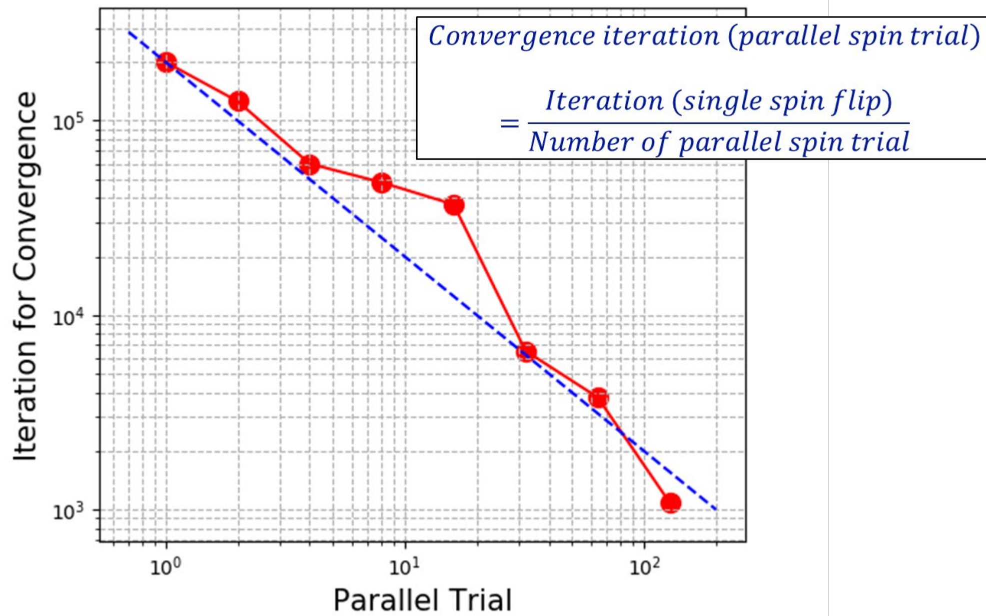


Figure 4-17 Acceleration of convergence with parallel spin trial method (Reprinted from Ref.[90] with permission)

As shown in the simulations result of spin glass optimization (Figure 4-17), the acceleration of optimization is approximately proportional to the number of parallel trial (or, number of columns of RRAM array). This tells that the parallel trial method, or digital annealing with RRAM array, has good scalability for large problem with large RRAM array.

4.6 Conclusion

The spin glass system, one of the widely analyzed NP-complete COPs, was solved by using simulated annealing in a hardware system consisting of Ta_2O_5 RRAM crossbar arrays for dot-product operations and Cu-based CBRAM devices for the emulation of stochastic events. The probability of spin flipping is first determined by vector-vector multiplications between spin vectors and the interaction vector. The event then emulated in the stochastic CBRAM by applying a programming pulse with pulse width proportional to the change of Hamiltonian. The process is repeated until the ground state is achieved. A parallel spin flip strategy and a parallel-trial method have also been developed to better utilize parallel operations in the RRAM crossbar array.

Chapter 5. Stochastic Learning of Deep Neural Network Using Stochastic RRAM Crossbar Array

5.1 Introduction

Recently, deep neural networks (DNNs) are widely used for various applications such as image processing, speech recognition, and natural language processing.[1], [96] Since the structure of a DNN is 'deep' which means that there are numerous hidden layers to represent the hierarchy of features for the task, training DNNs that offer high performance such as AlexNet,[97] GoogLeNet,[98] and ResNet-50[99] is expensive in terms of computational cost. For example, AlexNet and ResNet-50 have 61M and 25.5M parameters in the network, and require 724M and 3.9G multiplication-and-accumulation (MAC) operations respectively, as shown in Figure 5-1. For even a single training epoch of the 1.2M training data in the ImageNet dataset, the number of MAC operations is as large as 10^{15}, which will take more than a few weeks to compute with conventional CPUs. As researchers realized that larger, deeper, and wider models perform better, graphic processing units (GPUs) have been used for the acceleration of deep learning tasks, and even further hardware acceleration is required for more complex tasks such as dynamic video analysis for self-driving cars. The efforts to improve deep learning performance include not only enhanced computational throughput, but also to compress the scale of the network to reduce computation while not significantly sacrificing the network's performance.[100]

	AlexNet	GoogLeNet(v1)	ResNet-50
Top-error	16.4	6.7	5.3
# of Conv layers	5	21	49
# FC layers	3	1	1
Total Parameters	61M	7M	25.5M
Total MACs	724M	1.43G	3.9G

Figure 5-1 Properties of deep neural networks for the ImageNet dataset, reproduced using data from Ref.[97]–[99] As the number of the convolution layers increases the top-error rates reduce while the total number of parameters and the number of MAC operations for a single input increase.

Weight quantization is one of the promising strategies to compress the network by reducing the bitwidth of the parameters.[101]–[105] According to Horowitz et. al.[106], reduced bitwidth and integer arithmetic operations are more efficient than high precision and floating-point numbers in conventional computing architectures, as shown in Figure 5-2. Especially, compared to 32bit floating point MAC operations, 8bit integer type MAC operations require 20 times cheaper energy cost and 37 times smaller area. To push the limit of the efficiency of low precision operation, M. Courbariaux et. al. proposed BinaryConnect to train a DNN with binarized weights.[105]

In general, however, a neural network needs to maintain high precision information implement commonly used learning algorithms such as backpropagation, where small learning rates lead to infinitesimal weight updates that need to be accumulated to the original weights. If the bitwidth of the weights is too small, the updated weights will be rounded to their original values and learning will fail. This is the reason why attempts to directly train binary neural networks could not achieve competitive performances.[107], [108]

Operation Type	Energy (pJ)	Area (mm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A

Figure 5-2 Rough energy cost and chip area for various operation and bit-width precisions in 45nm CMOS technology, Reproduced using data from Ref.[106]. As precision increases from 8bits to 32bits and the integers change to floating point numbers, the energy and area consumption increases accordingly. The last 2 rows are cost for data fetching from memory.

To maintain the high precision weights and simultaneously exploit the benefits of quantized weights, BinaryConnect divides the commonly used back propagation algorithm into two steps, propagation step and update step.[105] In propagation step, the input x propagates forward to obtain an estimate of the output activation $\mathrm{o}(x)$, and the error $\delta(x)$ in the output layer propagates backward to calculate all the gradients of the error with respect to weights $\frac{\mathrm{d}\delta(x)}{\mathrm{d}w}$. This step can be implemented using the quantized weights to both reduce computation cost and help inject noise to the system. Then, in the update step, the calculated gradients are multiplied by the

(small) learning rate λ and added to the current weights w. This step still needs to be implemented using the high precision weights.

In BinaryConnect described in Figure 5-3, these conditions are met by using two networks: one storing the original high precision weights, and the other storing the binarized weights obtained from the high-precision weights after weight updates. Specifically, the high precision weights $w \in R$ of a neural network (right panel of Figure 5-3) are first used to generate binarized weight $w_b \in \{-1, +1\}$ through binarization process. The generated binary weights are then used for the propagations step (left panel of Figure 5-3) to estimate the output activation $o_b(x)$ and the gradients $\frac{d\delta_b(x)}{dw_b}$. The desired weight update $-\lambda\frac{d\delta_b(x)}{dw_b}$ calculated from the binary network are then accumulated to the high precision weight *w* to yield the updated weight *w'* (right panel of Figure 5-3). This cycle is then repeated until the training of the neural network is completed. Despite of the noisy weight update due to binary weights, BinaryConnect demonstrated competitive classification results close to state-of-the-art deep learning techniques. The high performance of BinaryConnect is addressed by averaging out of the noisy steps from binarization process like the stochastic gradient descent method, because the average of binarized weights throughout the training process is likely close to the high precision weights. Finally, by dividing the backpropagation algorithm into the propagation step with binarized network and the update step with high precision network as the schematic in Figure 5-3, BinaryConnect can make the training process more cost effective due to the cheap 1-bit MAC operations.

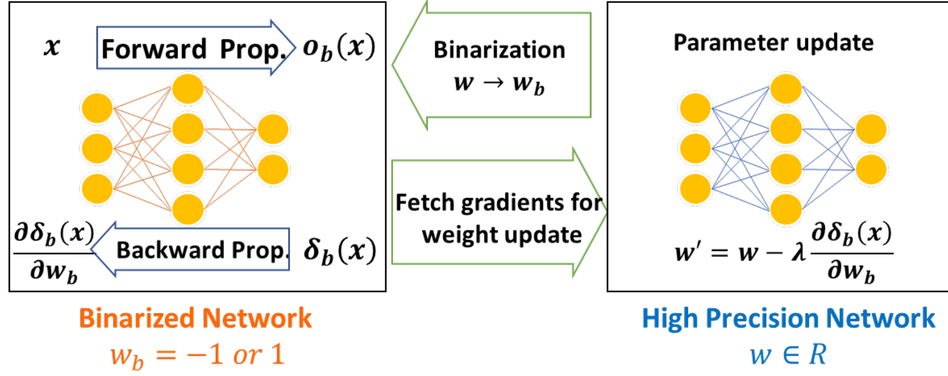


Figure 5-3 Schematic of BinaryConnect (Reproduced using data from Ref.[105]). At the first stage of network training, a binarized network (left panel) is created from the high precision network (right panel). The binary network is used in the propagation step to obtain gradients with respect to weights, which are then fetched to the high precision network for weight update.

However, binarization process is a main bottleneck for BinaryConnect with high performance. Between two kinds of binarization process, deterministic binarization and stochastic binarization, stochastic binarization outperforms deterministic one. Deterministic binarization simply decides the binary weights as $w_b = +1$ if $w \geq 0$ and $w_b = +1$ if $w > 0$. Otherwise, stochastic binarization first clips the weights to $-1 \leq w_{clip} \leq +1$. The clipped weights are then linearly converted to binarization using probability listed in (Eq. 5-1) and (Eq. 5-2) so that $w_{clip} = +1$ has 100% probability to be $w_b = +1$ (0% to be $w_b = -1$) and $w_{clip} = -1$ has 0% probability to be $w_b = +1$ (100% to be $w_b = -1$).

$$p_{+1}(w) = p_{+1}(w_{clip}) = {(1 + w_{clip})}/{2} \qquad \text{(Eq. 5-1)}$$

$$p_{-1}(w) = p_{-1}(w_{clip}) = {(1 - w_{clip})}/{2} \qquad \text{(Eq. 5-2)}$$

The advantage of stochastic binarization is that the average of binarized network generated by stochastic binarization more precisely reproduce the high precision network than the average of binarized network generated by deterministic binarization.[105], [109], [110] However, even though stochastic binarization provides better accuracy compared to deterministic binarization (Error rate in CIFAR-10: 8.27% with stochastic binarization and 9.90% with deterministic binarization)[105], only deterministic binarization method has been used for hardware implementation of quantized neural network with inevitable sacrifice of classification performance.[111]–[113] The reason is that the stochastic binarization requires the expensive Monte Carlo method to generate binary numbers following the Bernoulli distribution with the desired binarization probability.

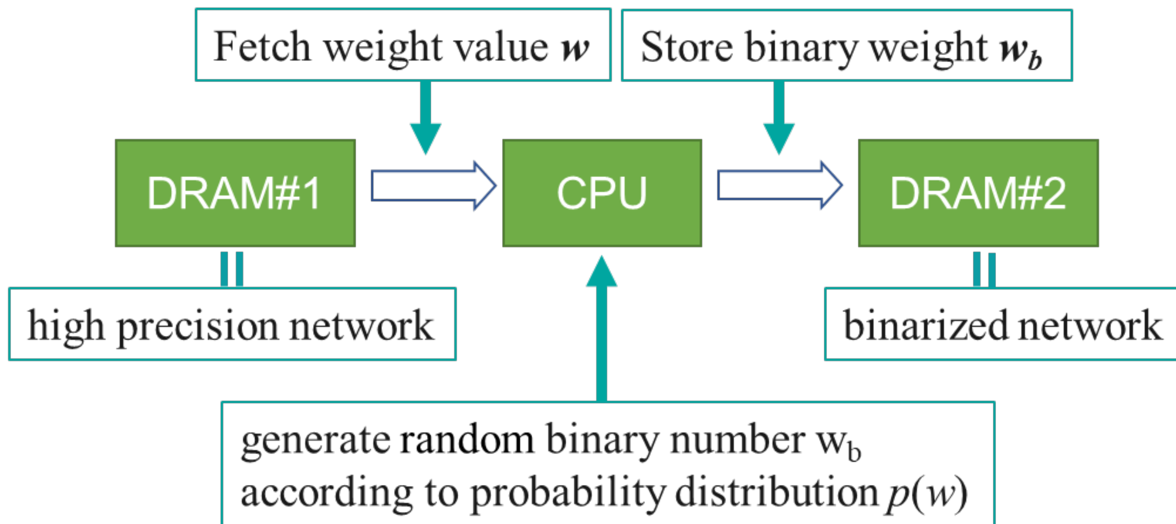


Figure 5-4 Data flow of stochastic binarization process. Weights of the high precision network (stored in DRAM #1) are fetched to the CPU, the binarization probability is calculated, and compared to random numbers to determine the binary values. The generated binary numbers are then transferred to a different part of DRAM (DRAM#2) to store the weights of the binary network.

Data flow of stochastic binarization process in the von-Neumann architecture is described in Figure 5-4. For stochastic binarization process, the weights of the high precision network stored in DRAM need to be fetched and converted to a probability according to Eq. 5-1 and Eq. 5-2. Then, the probability is compared to random numbers ranging from 0.0 to 1.0 to decide the binary weight value. The obtained binary weights are then transferred to another DRAM block to store the weights for the binary network. Because data read from DRAM is 100 times more expensive than MAC operations, the von-Neumann bottleneck makes the stochastic binarization very inefficient.

In this study, we discuss the implementation of stochastic learning (BinaryConnect with stochastic binarization) using the native stochasticity of RRAM devices for deep neural networks. Stochastic binarization using RRAM array eliminates data transfer between the memory and the processing unit. The RRAM crossbar array also makes the stochastic binarization process parallel and energy-efficient. Effects of device non-idealities such as probability precision and device-to-device variability will also be discussed by analyzing the convolutional stochastic learning for the MNIST dataset.

5.2 Stochastic resistive switching with adjustable probability

The basic mechanism of RRAM is the electrochemical reactions of oxygen vacancies or active metal atoms and ionic migration by overcoming the associated energy barriers. These reactions are inherently stochastic, and the stochastic nature can be clearly observed in resistive switching processes.[27], [28], [114] For example, Gaba. *et. al.* has reported this phenomenon in Figure 5-5 (a-b) and suggested applications for stochastic computing.[28] Stochastic computing aims to replace arithmetic operations with operations on bitstreams from different sources.[115] Similar observations have now been reported by other groups, and other applications have been

proposed, including stochastic computing using fast stochastic bitstreams,[29] physical unclonable functions (PUFs) for security device,[116] true random number generators (RNGs),[117] and neuromorphic applications.[30], [118]

As discussed in Chapter. 3, we can reduce the SET voltage to improve the reliability of the Cu-based ALD CBRAM device using non-cumulative pulse scheme.[68] At low SET or RST voltages, the SET or RST resistive switching become stochastic. The probability of the device being SET or RESET during each SET or RST pulse can be described by an exponentially decaying distribution, also known as the Poisson distribution, as shown in Figure 5-5 (c-d). The switching probability can be modeled as $\mathrm{P}(\Delta t) = 1 - \exp(-\Delta t/\tau)$ for programming pulse width Δt, where the time constant τ_{SET} and the RST pulse time constant τ_{SET} depend on the voltage amplitude. One can then decide the pulse width to turn-on (HRS to LRS) or turn-off (LRS to HRS) the device with a specific switching probability according Eq. 5-3

$$\Delta t = -\tau \log(1 - P) \qquad \text{(Eq. 5-3)}$$

Therefore, the probability for stochastic binarization can be implemented by stochastic switching of RRAM devices with low SET or RST pulses (i.e., stochastic SET or RST pulses), where different binarization probability can be achieved by adjusting the pulse width Δt. For example, to achieve 50% SET switching (transition from LRS to HRS) of Cu-based ALD CBRAM studied in chapter 3, one only needs to apply a single SET pulse with 3V amplitude and 17.9ms pulse width. For 50% RST resistive switching (transition from HRS to LRS) of the device, a single 2.5V/15.7ms RST pulse can be applied to the device.

In addition, since the energy barrier for the electrochemical reactions or ion migration is modulated by the applied voltage, the characteristic time τ can be readily adjusted. For example, the reaction rate equation (Eq. 5-4)[26], [119] implies that the characteristic time (Eq. 5-5) is exponentially dependent on the applied voltage through a conversion factor α. Therefore, the pulse width for 50% switching probability calculated above can be shortened or increased for faster or slower stochastic binarization processes.

$$\Gamma = {}^{1}/_{\tau} = \upsilon e^{-(E_a-\alpha \mathrm{eV})/k_B T} \qquad \text{(Eq. 5-4)}$$

$$\tau = \tau_0 e^{-\beta V} \qquad \text{(Eq. 5-5)}$$

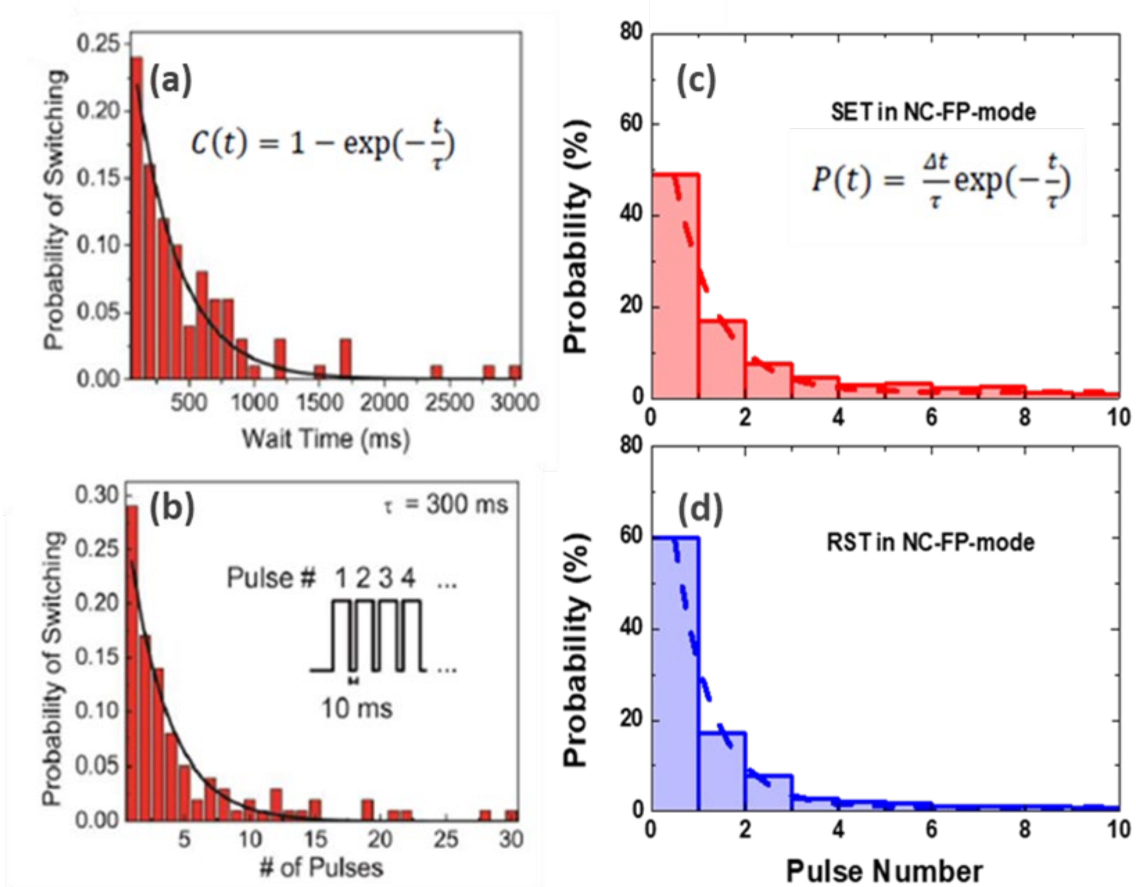


Figure 5-5 (a-b)Cumulative probability of the SET switching with (a) continuous pulse and (b) discrete pulses, obtained from Ref. [28]. (c-d) Stochastic switching observed in D-ALD CBRAM devices. Top panel (c) is a histogram of switching events for SET pulses (3.0V/10ms), and bottom panel (d) is a histogram of switching events for RST pulses (-2.5V/10ms). The histograms can be modeled with Poisson distributions [68]

5.3 Stochastic Binarization Mapping in RRAM Crossbar Array

5.3.1 Deterministic Data Migration Strategy

In addition to control the switching probability of a single stochastic device, a systematic method is required to map multi-bit values to stochastic binary values following the desired Bernoulli distribution. We note a RRAM crossbar array has the ability to implement in-situ data migration needed for in-memory computing.[24] As shown in Figure 5-6, a RRAM array with two columns with floated rows can migrate all the data in the data column from the left side (filled with data 1 or 0 represented by LRS or HRS, respectively) to the target column at the right side (initialized with HRS states) with a single programming pulse, by applying 0V on the right column and the SET pulse on the left column. Before data migration, cells in the target column are initialized to HRS. If a left-side device is in HRS, the SET voltage is divided between the data cell and the target cell and cannot change the state of the target cell, allowing it to remain at HRS. Otherwise, if the data cell is in LRS, essentially all the SET voltage will be dropped on the target cell to program it to LRS. Note that a single pulse on the left column will be applied to all the rows and enable parallel data migration between the two columns, without having to accessing external memory to temporarily store the data. This method can be extended to an RRAM array with multiple columns by using 1T1R structure to avoid sneak paths that can lead to incorrect data transfer.

5.3.2 Single-Bit Stochastic Binarization Processes

The next question is then what will happen if we modify the data migration strategy for a RRAM array with stochastic SET (or RST) pulse with 50% SET (or RST) probability for the target devices. The programming voltage of the stochastic SET or RST pulse is below the deterministic switching threshold of the target devices, and the pulse width is adjusted to make the SET or RST

switching probability 50%. To minimize the disturbance of the data column due to the stochastic pulse, the data column needs be comprised of RRAM devices with higher SET and RST voltages than the devices in the target column. Moreover, unlike the deterministic data migration strategy discussed in the last chapter, here the devices in the target column will also be reset with a pre-determined probability (e.g. 50%). To reliably deliver both SET and RST stochastic pulse to the target column devices, the resistance of the data column devices in LRS need to be at least two-orders lower than the resistance of the target column devices in LRS to minimize the voltage divider effect during RST when the target column device is already in LRS. A desired situation is $\mathrm{R_{data,LRS}} \ll \mathrm{R_{target,LRS}} \ll \mathrm{R_{data,HRS}} \approx \mathrm{R_{target,HRS}}$. In this case, applying a stochastic SET pulse with 50% probability will cause the the LRS states (data '1') stored in the data column to be copied to the target column with 50% probability, while no change will happen to the target column device if the data column device is at HRS state (data '0') because of the voltage divider effect, as shown in Figure 5-7. Similarly, when the target column devices are already at LRS, applying a stochastic RST pulse with 50% probability will have a 50% chance to turn off the target column devices to HRS if the data column device is at LRS state (data '1'), while no change will happen to the target column device if the data column device is at HRS state (data '0'), as sown in Figure 5-8. These two cases are termed as single-bit stochastic binarization processes, and they will be used as a building block for the multi-bit stochastic binarization process.

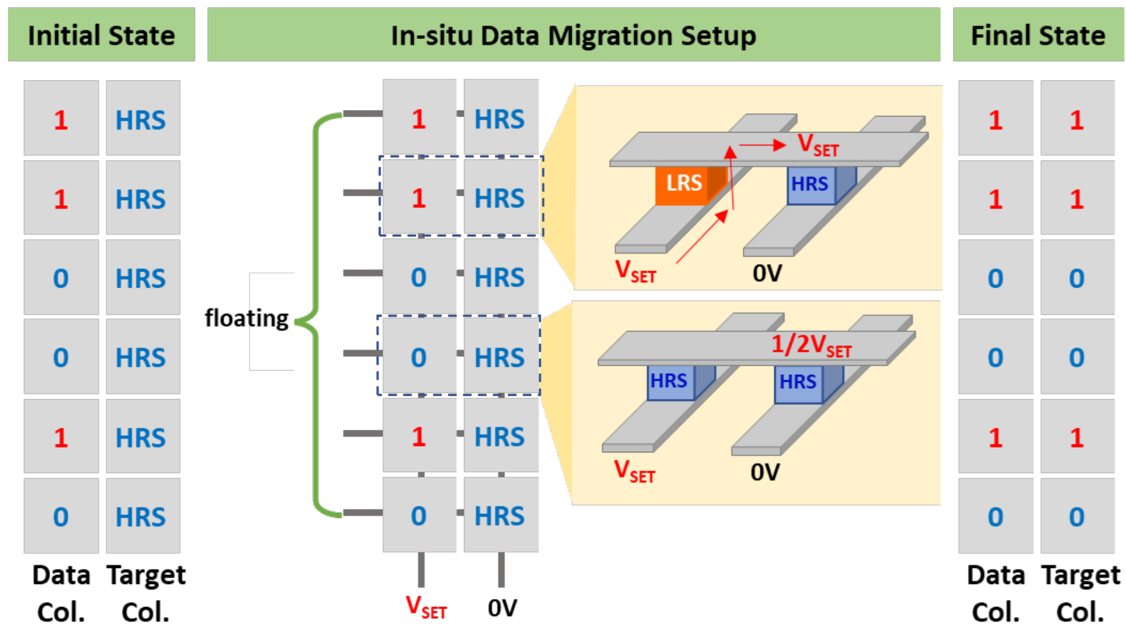


Figure 5-6 Schematic of in-situ data migration as suggested in Ref.[24]. Resistive states of an RRAM array is tracked during in-situ data migration process. 1 and 0 are represented by the LRS and the HRS of the RRAM devices. The left column of the array contains data devices, and the right column contains target devices. With this data migration setup, when the data device is at LRS essentially all V_{SET} is delivered to the top electrode of the target device, while when the data device is at HRS only half of V_{SET} is delivered to the top electrode of the target device, allowing 1 and 0 to transferred to the target device.

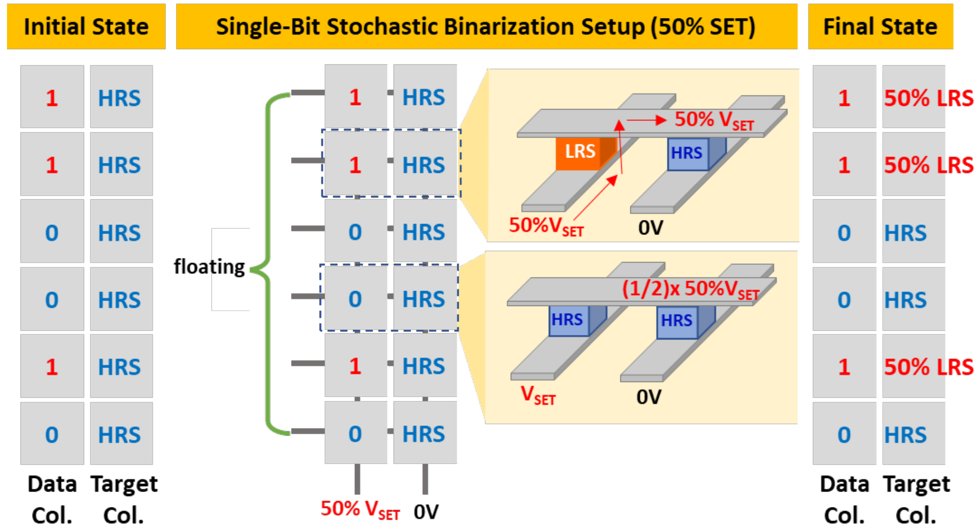


Figure 5-7 Schematic of single-bit stochastic binarization for stochastic SET pulse with 50% SET probability. A stochastic SET pulse with 50% switching probability is applied to the target column. The stochastic pulse is delivered to target devices only if the associated devices is in LRS. As a result, the data '1' stored in the data column is copied to the target column by 50% probability.

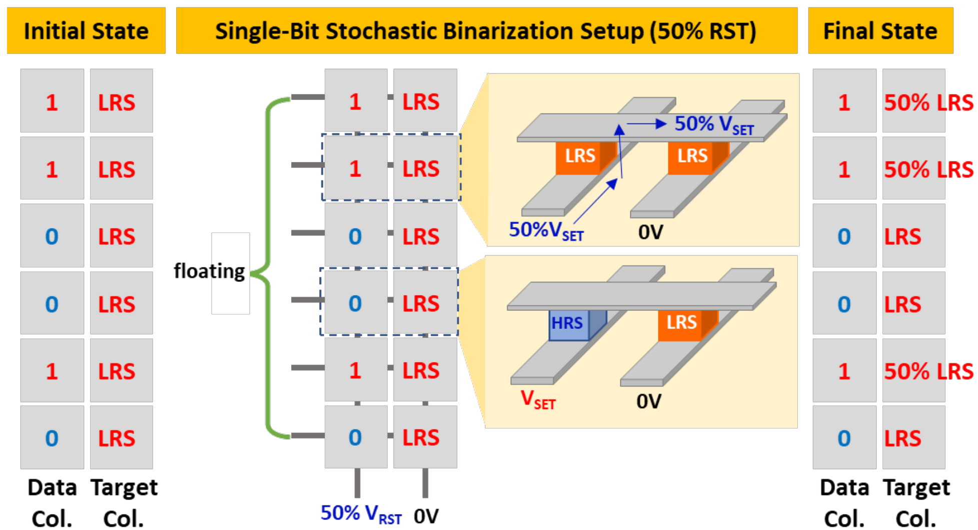


Figure 5-8 Schematic of single-bit stochastic binarization for stochastic RST pulse with 50% RST probability. The initialization condition of the target column is all LRS in this case. The 50% RST stochastic pulse is applied to target column devices and turn them off by 50% only if the associated data column devices are in LRS.

5.3.3 Multi-Bit Stochastic Binarization Process

A strategy was then developed to perform stochastic binarization of multi-bit weights by combining a few single-bit stochastic binarization processes, as illustrated in Figure 5-9. The RRAM array consists of three different parts that are connected to shared rows (top electrodes) with 1T1R structure to avoid sneak path problem. The first and second parts are 'weight array' (left, red box) and 'inversion array' (center, blue box) that store the data of the high precision weights and their inverted values, respectively. The weight array is working as a storage of high precision network as well, and it will be used for inference task when the training of the network is completed as proposed in FPCA framework.[24] The third part is 'binary column' (right, orange box) that is made of RRAM devices with lower SET and RST voltage than the devices in the weight array and the inversion array. With the device conditions for SET/RST voltage range, the stochastic SET/RST pulses with 50% probability for the binary column devices hardly disturb the weight array and the inversion array. Moreover, unlike the data migration tasks shown in Figure 5-6, during the stochastic binarization process the devices in the binary column can be at LRS after some steps, then needs to be RST with certain probability (instead of 100% as in the simple data migration case). To allow RST with the desired probability, the LRS resistance in the weight array and the inversion array need to be at least two orders of magnitude lower than the LRS resistance of the binary column. This effect can be obtained by limiting the programming current, e.g. by using high resistance bottom electrodes for the binary column. An ideal resistance configuration is $\mathrm{R_{weight,LRS}} \approx \mathrm{R_{inversion,LRS}} \ll \mathrm{R_{binary,LRS}} \ll \mathrm{R_{weight,HRS}} \approx \mathrm{R_{inversion,HRS}} \approx \mathrm{R_{binary,HRS}}$, where "<<" represents 2 orders of magnitude difference. At these conditions, the voltage amplitude of the stochastic SET/RST pulses can be delivered to the binary column correctly within 1% from the desired value. With this RRAM array structure, stochastic binarization of high precision

weights can be implemented with the following steps, as shown in Figure 5-9 with a few example weights. In addition, the history of probability distribution of the binary column through the stochastic binarization process is summarized in Figure 5-10.

(STEP.1) All the weight values are clipped by a clip function in Eq. 5-6 first.

$$w_{clip} = \begin{cases} w \ if \ |w| \leq 1 \\ +1 \ if \ w > 1 \\ -1 \ if \ w < -1 \end{cases} \qquad \text{(Eq. 5-6)}$$

A clipped multi-bit weight value w_{clip} $(-1 \leq w_{clip} \leq 1)$ is stored in the weight array as following. For positive weights, "1" is used to represent the positive sign at the most-significant-bit (MSB) of the weight array, followed by the other bits representing the absolute weight value. For negative weights, "0" is used to represent the negative sign at the MSB, followed by the other bits representing the 2's complement of the negative weight value. And, $w_{clip} = -1$ is represented by all "0" bits. For instance, w_1 = 0.75 and w_2 = -0.75 with 3-bit precision are converted to 111 and 001, respectively. The converted binary values are stored in the left-most part of the system, which is called 'weight array'.

(STEP.2) The stored weight in the weight array is copied to the 'inversion array' which is adjacent of the weight array, after inversion of every binary value. For example, w_1 = 0.75 and w_2 = -0.75 are converted to 000 and 110, respectively, and written into the inversion array.

(STEP.3) Apply SET pulse with 50% programming probability (termed 50% SET pulse) between the least-significant-bit (LSB) column of the weight array and the 'binary column', which is located at the right side of the system.

(STEP. 4) Apply RST pulse with 50% programming probability (termed 50% RST pulse) between the LSB column of the inversion array and the 'binary column', which is located at the right side of the system.

(STEP.5) Repeat steps 3-4 to transfer probability of the 2nd, 3rd, ... Nth LSB of the weight to the same binary column until binarization is completed.

The stochastic binarization process using a RRAM array can be proved as below. In this proof, the RRAM array is assumed to have 2N+1 columns where N is the number of columns of both the weight array and the inversion array. The right-most single column is assigned as the binary column. In fact, the initialization method in STEP.1 first linearly stores the clipped weights w_{clip} ($-1 \leq w_{clip} \leq 1$) into the weight array as a form of unsigned binarization probability values p ($0 \leq p \leq 1$). If we consider that the values of the weight array are unsigned significand of unit interval $[0,1)$ or unsigned fraction value with N-bit precision, a positive sign of w_{clip} converted as "1" and stored at MSB of the weight array ensures that the unsigned fraction value is larger than 0.5. Moreover, the absolute value of the positive weight written in the rest of the columns is an addition to 0.5, and the range of positive clipped weights ($0 \leq w_{\mathrm{pos},clip} \leq 1$) are compressed to unsigned fraction value ranging from 0.5 to 1.0 in linear fashion. Otherwise, a negative sign of w_{clip} stored as "0" at MSB of the weight array will confine the range of the fraction value from 0 to 0.5, and the 2's complement inverts the order of the absolute value of the negative weights upside down. Thus, the negative clipped weights ($-1 \leq w_{\mathrm{neg},clip} < 0$) changed to unsigned fraction value within $[0.0,0.5)$ Finally, the fraction values in the weight array can be interpreted as expected N-bit probability value because they follow the weight-probability conversion equations, Eq.5-1 and

Eq.5-2. In the example above, 111 and 001 converted from w_1 = 0.75 and w_2 = -0.75 by STEP.1 can be interpreted as expected probability P of 0.875 and 0.125.

After STEP.1 and STEP.2 for initialization of the weight array and the inversion array, we can use the N-bit probability number and its inversion for stochastic binarization. Assume that one already knows the stochastic binarization process of k-bit probability numbers (k<N). Then, the stochastic binarization process of a (k+1) bit probability number p_{k+1} can be conducted by first stochastic binarization of the k-bit probability number p_k ($\mathrm{p}_k = \mathrm{p}_{k+1}\ mod\ 0.5$) representing the least-significant k-bits, followed by additional operations to adjust the LRS/HRS probability distribution in the binary column based on the most-significant bit (MSB) digit, as discussed below. Through the stochastic binarization of the k-bit binary, the binary column devices will be programmed to LRS with a p_k probability Bernoulli distribution and HRS with a (1 - p_k) probability Bernoulli distribution. Afterwards, one can calculate the final probability distribution for the two different cases (MSB = '1' or '0'). The goal is to change the LRS probability distribution from p_k to (0.5+ p_k/2) for MSB = 1, and to (p_k/2) for MSB = 0. In the cases shown in Figure 5-11 (a) and (b), if the MSB of the (k+1)-bit is '1', p_{k+1} should be 0.5+ p_k/2 that results to p_{k+1} =87.5% for p_k =75% and p_{k+1} =50% for p_k =0%,. The change of probability distribution in the binary column can be implemented by connecting the MSB column of the weight array with the grounded binary column using a 50% SET pulse. With k-bits, the device in the binary column has a probability p_k in LRS, and a probability $(1 - p_k)$ in HRS. If the MSB of the weight is '1' (LRS), then applying a 50% SET pulse will produce a 50% probability for the device in HRS in the binary column to switch to LRS. As a result, the probability of finding the device in LRS after the MSB operation is $p_k + (1 - p_k)/2$. If the MSB of the weight is '0' (HRS), a 50% RST pulse is applied so that the binary column device will have a 50% to be reset if it is already in LRS. As a result, the probability

of finding the device in LRS after the MSB operation is (p_k/2). Examples of these operations are as shown in Figure 5-11 (a) and (c).

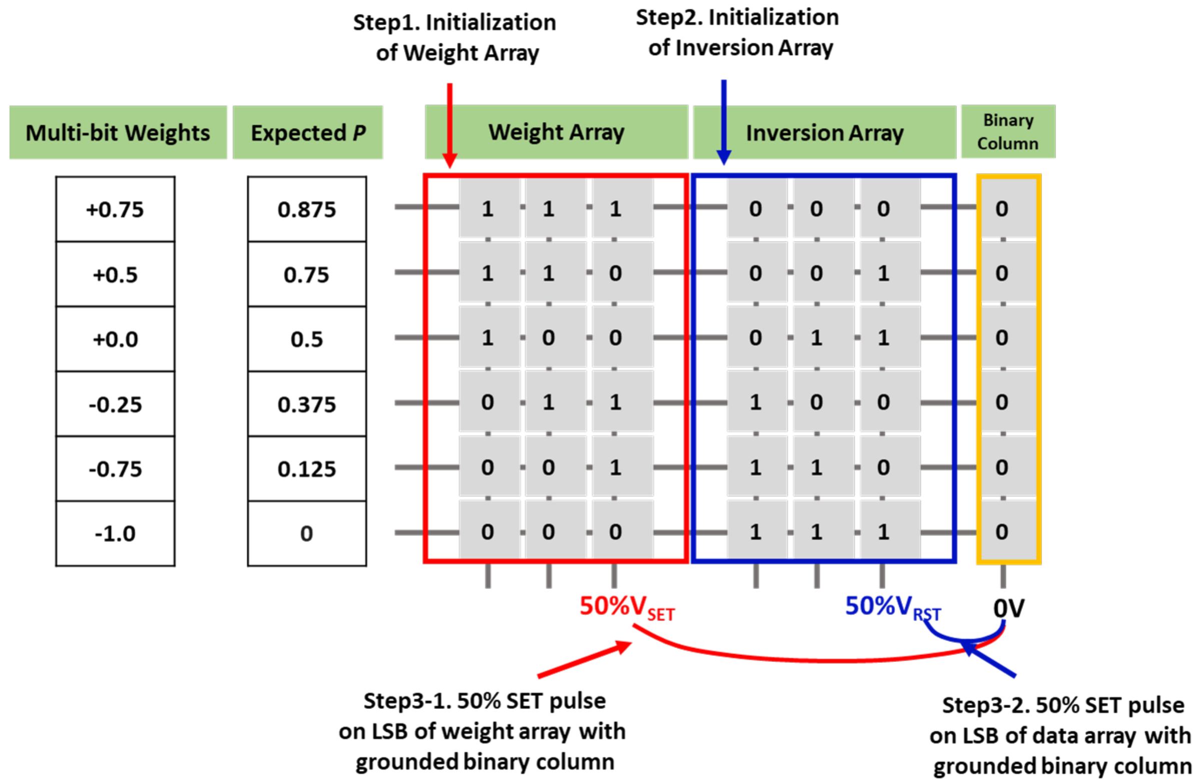
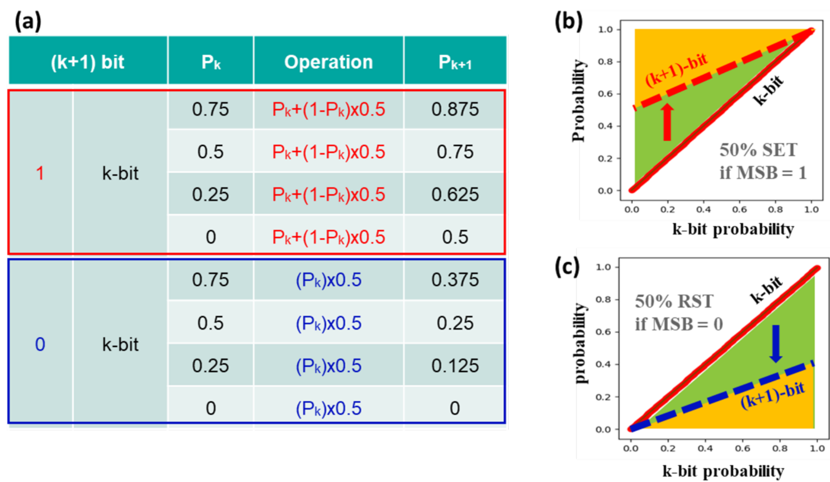


Figure 5-9 Schematic of multi-bit stochastic binarization process using an RRAM array. The system includes a weight array, an inversion array, and a binary column. Initialization of the weight array and the inversion array are explained in STEP1 and STEP2. The grounded binary column is connected to the least-significant-bit (LSB) of the weight array with 50% SET pulse first in STEP3, then the LSB of the inversion array with 50% RST pulse in STEP4. The processes, STEP3 and STEP4 are repeated after moving to the next bit until binarization is complete.

Weights	Exp. Prob.	Initial	1st LSB		2nd LSB		3rd LSB		Final Bin. Prob.
			Weight	Inv.	Weight	Inv.	Weight	Inv.	
+0.75	87.5%	0%	50%	50%	75%	75%	87.5%	87.5%	87.5%
+0.5	75%	0%	0%	0%	50%	50%	75%	75%	75%
+0.0	50%	0%	0%	0%	0%	0%	50%	50%	50%
-0.25	37.5%	0%	50%	50%	75%	75%	75%	37.5%	37.5%
-0.75	12.5%	0%	50%	50%	50%	25%	25%	12.5%	12.5%
-1.0	0%	0%	0%	0%	0%	0%	0%	0%	0%

Figure 5-10 The probability history of the binary column devices to be '1' or LRS from initial state to end of the stochastic binarization step. The log starts from 'initial' column in the table. For each nth LSB, STEP3 and STEP4 is conducted by connection of the grounded binary column to nth LSB in the weight array and the inversion array, respectively. The stochastic binarization proceed from the 'initial' column to the 'Final Binary Probability' column through the table. Weights from -1.0 to +0.75 are successfully transformed to binary states following expected probability distribution.



(a)

(k+1) bit		P_k	Operation	P_{k+1}
1	k-bit	0.75	$P_k+(1-P_k)x0.5$	0.875
		0.5	$P_k+(1-P_k)x0.5$	0.75
		0.25	$P_k+(1-P_k)x0.5$	0.625
		0	$P_k+(1-P_k)x0.5$	0.5
0	k-bit	0.75	$(P_k)x0.5$	0.375
		0.5	$(P_k)x0.5$	0.25
		0.25	$(P_k)x0.5$	0.125
		0	$(P_k)x0.5$	0

Figure 5-11 The generalization of stochastic binarization of k-bit probability to (k+1)-bit probability. (a) For both '1' MSB and '0' MSB cases, the operations required for p_{k+1} obtained from p_k is described in the upper red box and lower blue box, respectively. In the listed operation, (b) The k-bit probability (red solid line) should be converted by MSB = '1' to red dashed line by changing the lowest probability from 0.0 to 0.5. (c) The k-bit probability (blue solid line) should be lowered by MSB = '0' to blue dashed line by changing the highest probability from 1.0 to 0.5.

In this way, we can generate random binary numbers (stored in the binary column) with the Bernoulli distribution corresponding to weights in the high precision network (stored in the weight array). The major benefit of this approach is the elimination of data communication between memory and CPU needed to apply the Monte Carlo method. Unlike the von-Neumann architecture shown in Figure 5-6, all the operations in the proposed RRAM-based stochastic binarization happen in a RRAM array without having to read the data in and out of the array, as shown in Figure 5-12. The proposed stochastic in-memory computing saves energy cost from data migration, calculation of probability from weight value, and random number generation.

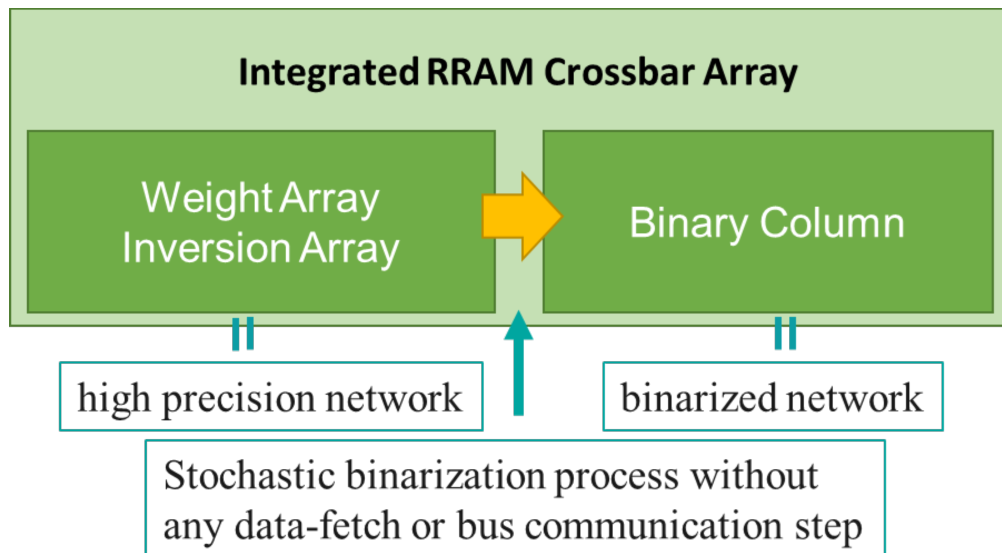


Figure 5-12 Schematic of the proposed in-memory computing approach for multi-bit stochastic binarization of a neural network. The binarization process is much more simplified compared with the von Neumann architecture implementation depicted in Figure 5-4.

Figure 5-13 shows Monte Carlo simulation results of the proposed RRAM-based stochastic binarization processes. By running the stochastic binarization process 1000 times, statistical distributions can be obtained from the binarized results in the binary column, for different weight precision cases (e.g. 2, 4, and 6 bit weights). These simulation results verify that using the proposed approach, the expected probability stored in the weight array can be reliably transferred to the probability of getting "1" in the binary column through the stochastic binarization process.

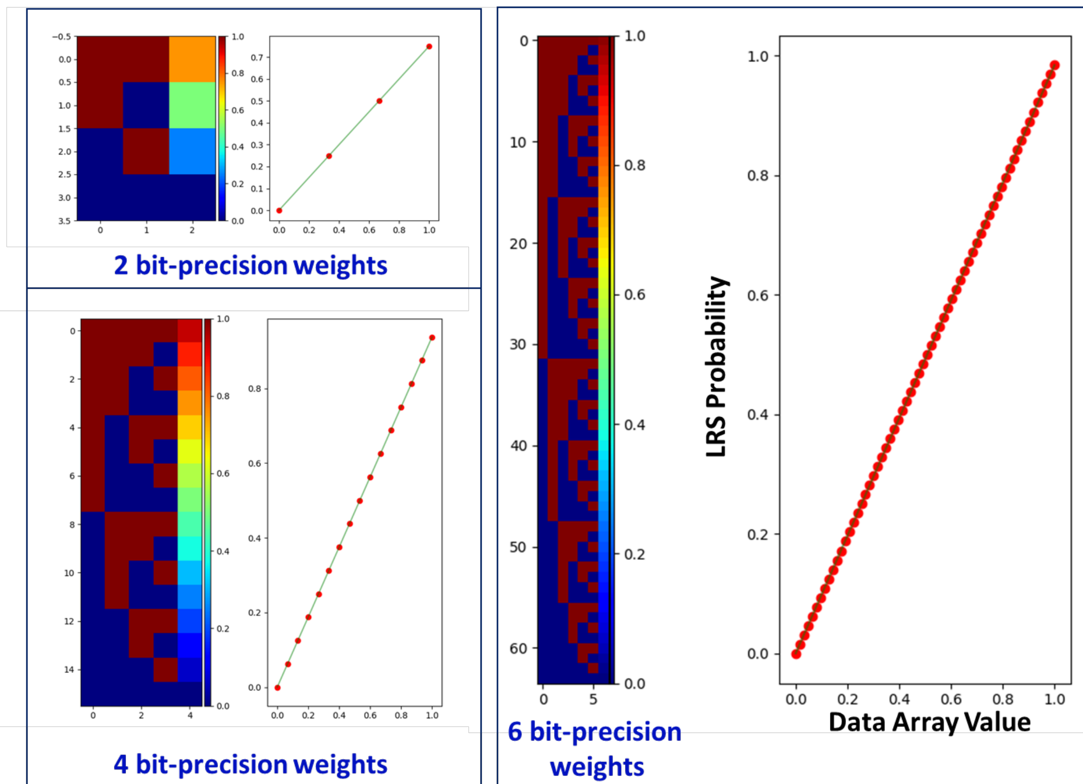


Figure 5-13 Monte Carlo simulation results of stochastic binarization for 2 bit, 4 bit, and 6bit weights. Left panel of each case shows the weight array and the binary column. Red and blue colors in the weight array represent LRS and HRS, respectively. Color in the binary column representing the probability of having '1' (LRS) from the 1000 tests, based on data in the weight array. The right panel plots the measured LRS probability in the binary column with respect to the weight array value.

For the stochastic binarization process, the required number of programming pulses is just twice the number of bits in weights, without requiring any memory read process. The absence of read process makes the process very attractive by eliminating the need of analog-to-digital converters and sense-amplifiers. Overall the parallel stochastic dataflow provides significant benefit compared to conventional architectures. For instance, in von Neumann architecture, converting a thousand 8bit weights would require a thousand cycles of DRAM access for input weights, random number generations, arithmetic operations, comparison operations, and writing-back to the DRAM. In the proposed implementation, the conversion can theoretically be achieved in a RRAM crossbar array with a thousand rows, using only 16 cycles of stochastic binarization pulses, as illustrated in Figure 5-12, since all the weights are binarized in parallel.

5.4 Stochastic Learning of Convolutional Neural Network

To highlight the benefit of stochastic binarization using RRAM crossbar arrays, a convolutional neural network (CNN) for classification of handwritten digits (e.g. MNIST)[120] was tested by using a high precision network (baseline), BinaryConnect with deterministic binarization (deterministic binarization model), and BinaryConnect with stochastic binarization (stochastic binarization model).

The architecture of the CNN is (16C3)-MP2-(32C3)-MP2-(64C3)-(300FC)-10SM, where C3 is a 3×3 convolution layer with ReLU activation functions, MP2 is a 2×2 max-pooling layer, FC is a fully connected layer, and SM is a softmax output layer. For the training and the inference of MNSIT dataset, 55k training examples and 10k test examples are used, respectively. Figure 5-14 (a) and (b) shows the error rate and cost function of the softmax output layer from the three different training algorithms. At 30k iterations of minibatch training with batch size of 100 training

examples, the error rate of the baseline, the deterministic binarization model, and the stochastic binarization model is 0.85%, 1.94%, and 0.85%, respectively. Note that the error rate from the deterministic binarization model is larger than the twice of the error rate from the stochastic binarization. In Figure 5-14 (b), the trace of the cost function of the baseline model and the stochastic binarization model matches well, although the deterministic binarization model maintains one order higher cost than the stochastic binarization model. The high performance of the stochastic binarization model, despite of using binary information stored in the network, is due to the statistical cancellation of binarized weights (+1 or -1) through the whole training process to make the average effect of weights on output values to be same as the weights with high precision.[105]

Moreover, stochastic binarization model has another advantage of regularization effect over deterministic binarization model.[105] Due to the large number of parameters, high complexity of the multi-layered structure, and nonlinear neurons, CNN is very good at learning the distribution of training data. However, if the learning is too precise then the network can even learn the noise in the training data, besides actual features. As a result, test results using samples not in the training set can be degraded, due to this "overfitting" problem. Various regularization methods such as L2 regularization, dataset augmentation, early-stopping, and drop-out to reduce effects from noisy information in training examples have been developed to minimize the overfitting problem.[121] Drop-out, one of the most widely used regularization method, drops random parts of the neurons along with their connections in the network to make the weights noisy and the network more general to unseen data.[122] During training step, dropout actually training a number of randomly "thinned" network by dropped units. At inference step, all the noisy predictions from randomly thinned networks are averaged by simply using a single unthinned

network. The benefit of drop-out is that the computational overhead from training the deep and wide neural network can be reduced and serious overfitting problem can be avoided, simultaneously. Likewise, theoretical analysis of stochastic binarization model show that it can offer similar regularization effects with reduced computation by averaging the binarized network (instead using thinned network in drop-out), which is a main motivation for us to develop the stochastic binarization systems.[105]

Since the overfitting problem is more easily observed in small training datasets, 10k training examples randomly selected from the MNSIT dataset (so-called reduced-MNIST dataset, or RMNIST) are tested to check the regularization effect of stochastic binarization model, as shown in Figure 5-15. The left panel of Figure 5-15 is the error rate for the training data (blue solid line) and the test data (orange solid line), obtained from the high-precision network (baseline) trained with RMNIST dataset. The right panel is the result from stochastic binarized model. Although both networks succeeded to learn the RMNIST training dataset perfectly without any wrong prediction for training examples, the inference error of the test dataset from the baseline model is 15% higher than the stochastic binarization model. Although the inference error is almost the same (~0.85%) for both the baseline and the stochastic model when they are trained using the MNIST dataset with 55000 training examples, reduced training samples (1000 training examples) in the RMNIST cause apparent overfitting in the high-precision network, while the noise injected by the stochastic binarization can apparently mitigate the overfitting problem in this case. The better performance of stochastic learning implies that the stochastic learning not only improves computational efficiency, but also boosts deep learning performance by adding regularization effects. It is notable that dropout algorithms using the Monte Carlo method (to decide which weights are activated or inactivated), like stochastic binarization using the von Neumann

architecture, will also suffer from the memory bottleneck problem. On the other hand, by generating Bernoulli distribution in parallel, drop-out also can be accelerated by the proposed RRAM-based stochastic binarization method. In short, we expect stochastic binarization based on RRAM crossbar arrays can be generally compatible with regularization algorithms.

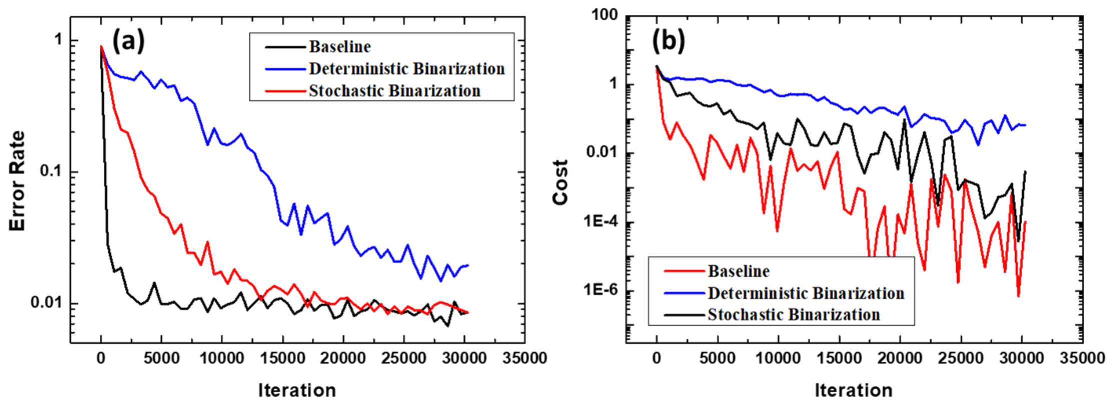


Figure 5-14 Comparison of (a) error rate and (b) cost function of the softmax output layer among the baseline model (high precision network training), the deterministic binarization model, and the stochastic binarization model.

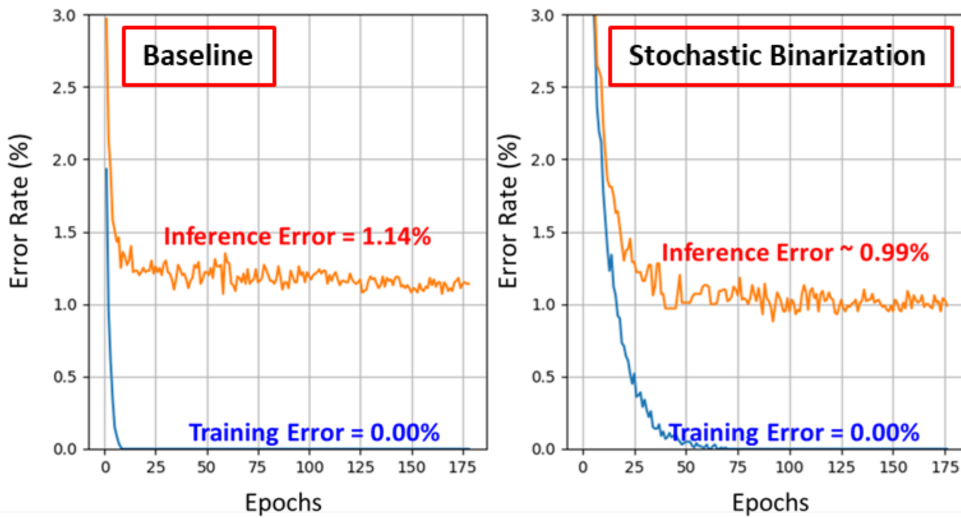


Figure 5-15 Reduced MNIST training results, using the baseline model (left panel) and the stochastic binarization model (right panel)

5.5 Optimization and Nonideality of RRAM-based Stochastic Binarization

Several other factors need to be considered for the implementation of RRAM-based stochastic binarization. As explained in section 5.3, RRAM-based stochastic binarization is composed of a weight array, an inversion array, and a binary column. The number of columns in the weight array and the inversion array is a critical factor for optimization of the stochastic binarization process, because the number of pulses needed for the binarization process is twice the weight length, which equals the number of columns in the weight array. In addition, the number of rows in the RRAM array determines the acceleration factor of the binarization process, which is in turn limited by device properties such as device-to-device variability. In this section, we discuss the effects of both weight precision and device-to-device variability on stochastic learning.

5.5.1 Effects of Weight Precision on Stochastic Binarization

The weights used in the high precision network need to have sufficient bits to retain the accuracy during training to implement the gradient descent algorithm, although the change of weights determined by the propagation step can include some noise like those used in the stochastic gradient method. However, increasing the bit length lowers the energy-efficiency and the speed of stochastic binarization process due to the increased number of programming pulses that are required. Therefore, there is trade-off between energy-efficiency and precision in the weight array. To optimize the precision of weights, error rate of the baseline model and stochastic binarization model with different weight precisions (32-bit floating point weights (black), fixed point weights with 4(magenta)/3(orange)/2(green)/1(blue)-bits) are examined, as shown in Figure 5-16 (a). Surprisingly, 4-bit weights (magenta solid line) show comparable performance of error rate below 1.0% to the baseline model and 32-bit FP weights, and 3-bit weights still have better performance than deterministic binarization. Therefore, in the weight array and the inversion array, 4 or 5 columns will be optimal size for handwritten digit classification. Note that the training with stochastic binarization saves 75% of bitwidth in the final weights when compared to conventional training of CNN which requires more than 16 bit precision.[110]

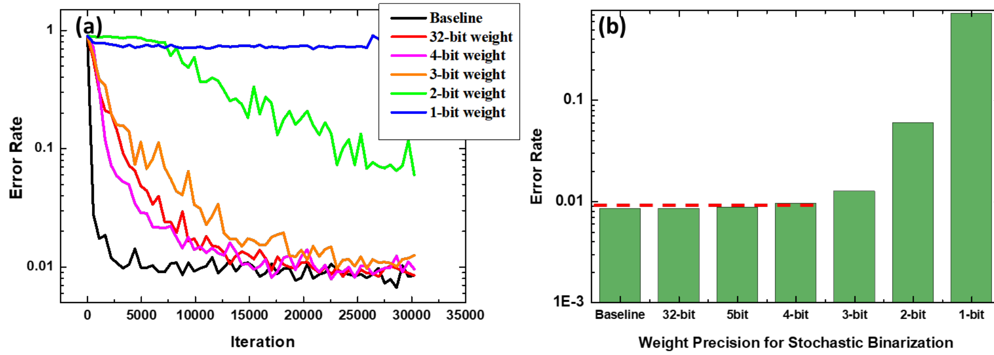


Figure 5-16 Error rate for MNIST classification, for different weight precisions using stochastic binarization.

5.5.2 Device-to-Device Variability Effect

The nonideality of stochastic RRAM devices can affect the deep learning performance. Cycle-to-cycle variability of a single stochastic RRAM devices is not a severe problem because it is already included in the device's stochastic behavior. However, device-to-device variability can degrade the network performance. For example, if each device has a different characteristic switching time, then the stochastic pulse applied for 50 % binarization probability can induce binary states that deviate from 50% for different devices. To test this effect, we analyzed various variability conditions (0.0% ~ 5.0% resistance variations) and tested these effects on MNIST dataset classification with CNN. The device-to-device variation is defined here by the range of differences of switching probability within RRAM devices in the binary column from 50%, when the devices is applied with stochastic SET/RST pulse designed for 50% switching probability. Specifically, device-to-device variability *var* is defined as the fluctuation range of the difference between the desired probability $p_{desired}$ and the actual binarization probability p_{RRAM} of from the weights, i.e. $p_{RRAM} = (1 + var)p_{desired}$. For example, a binary column with 1.0% device-to-device variation have switching probability ranging from 49.0% to 51.0% for 50% stochastic

SET/RST pulses. As can be seen Figure 5-17 (a), the error rate increases as the variability increases, as expected. Interestingly, the error rate for high variability cases seems to have a common feature where the network learns fast at the early training stage, lose information at moderate iteration cycles, and then gets trained again as the training iteration further increases. In other words, the device variability causes the network to fall in some local minima during training, but with longer training the network can escape from the local minima slowly. If we use the error rate from deterministic binarization model (1.94%) as the threshold to gauge the performance of stochastic learning, then 3.0% of device variability (with 1.68% error rate) is acceptable for practical applications of RRAM-based stochastic binarization model.

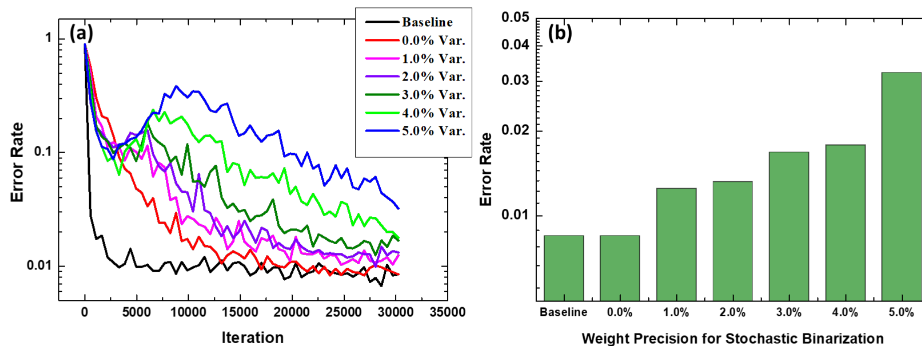


Figure 5-17 Effect of device-to-device variability on error rate of stochastic learning of CNN

5.5. Conclusion

In conclusion, we investigated exploiting the stochasticity of RRAM devices for deep learning. The stochastic binarization method using RRAM crossbar arrays developed in this study can generate Bernoulli distribution without any memory fetch, random number generation, CPU

arithmetic operations, and write-back processes. The benefit of better performance compared to deterministic binarization, regularization effect of stochastic learning and successful 4-bit weight precision training are verified. The effect of device-to-device variability was investigated and guidelines for practical implementations were developed.

Chapter 6. Future Work and Summary

6.1 Solving general combinatorial optimization problems

Despite the promise of combinatorial optimization problems (COPs), the time complexity of NP-complete or NP-hard problems, which exponentially increases as the size of the problem increases, makes many useful COPs unaffordable.[123] As a practical approach to solve difficult COPs, meta-heuristic algorithms like simulated annealing has been widely studied.[86] Chapter 4 in this thesis proposed a specialized method to utilize RRAM arrays for simulated annealing of the two-dimensional spin glass problem.[90] Since the two-dimensional spin glass model in the Ising model formulation can be transformed to any other NP-complete/hard COPs according to Ref.[124], the usage of RRAM array designed in chapter 4 can theoretically be generalized to solve other difficult problems such as scheduling problem, satisfiability problem, clique cover, and travelling sales man problem (TSP).

Among the difficult COPs, TSP is one of the most recognized. The goal is for a travelling salesman to find the shortest path to visit all the listed cities and return to his starting point making a Hamiltonian cycle. Although the problem is easy to understand, the salesman has to count all the possible number of Hamiltonian cycles which increases as exponentially as the number of cities increases. The Ising version of TSP (Eq. 6-1) is introduced in Ref.[124].

$$\mathrm{H} = \mathrm{A}\left(\sum_{v=1}^{n}\left(1-\sum_{j=1}^{N} x_{v,j}\right)^2 + \sum_{j=1}^{n}\left(1-\sum_{v=1}^{N} x_{v,j}\right)^2 + \sum_{(uv)\notin 1}^{n}\sum_{v=1}^{N} x_{u,j}x_{v,j+1}\right)$$

$$+\mathrm{B}\left(\sum_{(uv)\in E} W_{uv}\sum_{j=1}^{N} x_{u,j}x_{v,j+1}\right) \quad \text{(Eq.6-1)}$$

The Hamiltonian that describes the TSP is made of two parts. The terms in the first parenthesis is used to ensure that the path will include each city once and return to the starting location. $x_{v,j}$ is the binary spin vector of a travelling spin vector $\overrightarrow{x_{v,J}}$ that describes the travelling path. *v* and *j* represent the index of each city and its order in a prospective cycle, respectively. If the salesman visited city *v* among *N* cities at the *j*-th order in his travelling path, $x_{v,j}$ is assigned to '1'. Otherwise, $x_{v,j}$ is '0'. Note that the dimension of the spin vector $\overrightarrow{x_{v,J}}$ is N^2 for *N* cities and *N* possible visits. Because all the coefficients used in the first part are +1, -1, or 0, its computational cost from simple integer arithmetic operations (counting 1's and accumulating its squared integer) is relatively cheap. On the other hand, W_{uv} in the second part represents the distance between city *u* and city *v*, which is an analog value. The need of floating-point operations to calculate the second part in the Hamiltonian dominates the computational cost.

We note the Hamiltonian in Eq. 6-1 can be mapped to an RRAM crossbar system, as shown in Figure 6-1. Since the dimension of a travelling spin vector $\overrightarrow{x_{v,J}}$ is N^2, the interaction coefficients between $\overrightarrow{x_{u,J}}$ and $\overrightarrow{x_{v,k}}$ in the second part of the Hamiltonian can be represented by a $N^2 \times N^2$ matrix $\left[W_{(u,j)(v,k)}\right]$, where $W_{(u,j)(v,k)} = W_{uv}\delta_{j,k-1}$, and $[W_{uv}]$ is a $N \times N$ matrix comprised of the distance values between city *u* and city *v* and δ is the Kronecker Delta function of two indices which is 1 for same index values, and 0 for different index values. The $\left[W_{(u,j)(v,k)}\right]$ matrix is thus very large but sparse. Its non-zero elements are formed by repeating patterns of $[W_{uv}]$, which can

be mapped to a $N \times N$ RRAM crossbar array. As a result, computation of the 2nd term in the Hamiltonian can be performed through a $N \times N$ RRAM crossbar array, instead of mapping the full matrix which will requires a $N^2 \times N^2$ array.

As an extension of our work on hardware acceleration of simulated annealing, we expect the TSP problem can be efficiently solved using RRAM based architecture. A starting point may be using a 25×25 array to solve a TSP problem for 25 cities, which is finding the most optimal path among huge number of possible travelling paths, the factorial of 25. Such demonstrations will allow the RRAM crossbar based hardware to be expanded to solve general COPs, and bring this technology closer to practical applications.

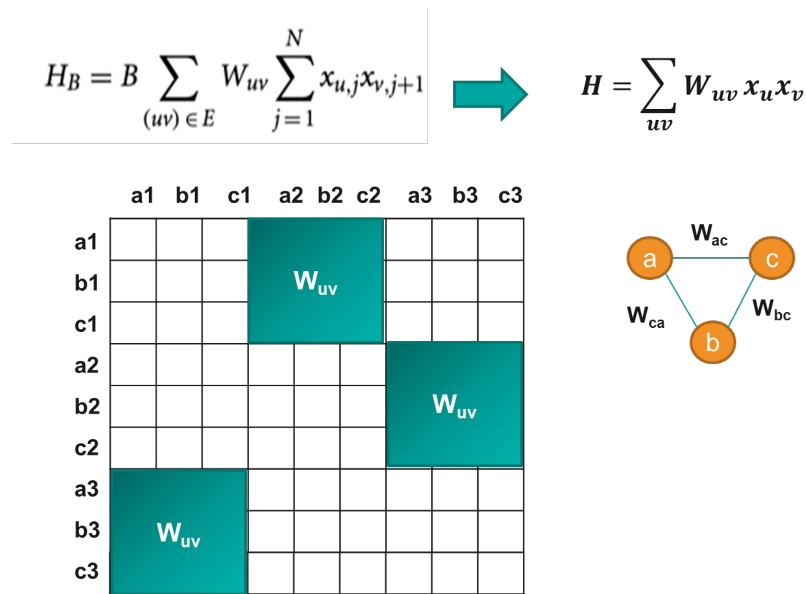


Figure 6-1 Schematic of implementing the travelling salesman problem (TSP). The nonzero elements of the $[W_{(u,j)(v,k)}]$ matrix are formed by repeating patterns of the $[W_{uv}]$ sub-array.

6.2 Summary

In chapter 1, we first introduced the memory wall problem of the conventional von Neumann architecture for data-intensive tasks. To alleviate the memory wall problem, RRAM devices are used as synaptic devices for neuromorphic computing and in-memory computing. The main advantages of RRAMs are their abilities to store weights and to perform vector-matrix operations directly through physics. Moreover, the diverse switching behaviors such as analog, digital, and stochastic resistive switching were explained, along with an introduction of their optimization and applications for computing beyond von Neumann architecture.

In chapter 2, we experimentally demonstrated that analog TaOx RRAM array can be used to perform principal component analysis for feature extraction and dimensionality reduction of the breast cancer dataset. To reliably initialize the TaOx RRAM crossbar array, we optimized the forming voltage from ~2.5V to ~1.1V. Using Sanger's rule, the principal components were obtained as the RRAM device conductances in the network after training. During the training process, the RRAM crossbar array was controlled by periphery circuitry, FPGA, and computer. The network was then successfully used to analyze sensory data from a standard breast cancer screening database with a high classification success rate (97.1%).

In chapter 3, we optimized the digital Cu-based CBRAM devices to achieve self-limiting current and low forming voltage for very low power computing applications. In this study, copper oxide layer and high-temperature Al_2O_3 layer were inserted as a copper ion supplier and diffusion barrier, respectively. The optimized device structure (Cu/CuOx/LT-ALD/HT-ALD/Pd) with double ALD (D-ALD) layer achieved low forming voltage ($V_{forming} \sim V_{SET} \sim 3.0V$), self-limited resistive switching with very low programming current (~10 nA), high ON/OFF ratio (>100), and nonlinear I-V (NL_{read} and $NL_{SET} \sim 10$) at low resistance state.

In chapter 4 and chapter 5, we proposed utilization of stochastic resistive switching of digital RRAM devices for hardware acceleration of simulated annealing (SA) and stochastic deep learning, respectively. For SA of the spin glass problem investigated in chapter 4, which is one of the typical NP-hard combinatorial optimization problems, we utilized vector-matrix multiplication functions in analog Ta_2O_5 RRAM crossbar array and stochastic switching properties in Cu-based CBRAM devices to accelerate an SA algorithm that solves a spin glass problem efficiently. In this RRAM-based SA accelerator, the change of Hamiltonian of the spin system and the probability of stochastic spin flipping event are calculated natively by Ohm's law and the stochastic resistive switching property of the RRAM device with Boltzmann distribution. In chapter 5, digital RRAM array with stochastic resistive switching was utilized for stochastic binarization for binary neural networks. The stochastic binarization using RRAM array accelerated the generation of binary random numbers with specific probability corresponding to high precision weights. Specifically, the stochastic binarization can be achieved in parallel without having to read the data in and out of the array, thus eliminating data communication between memory and CPU needed to apply the Monte Carlo method. Finally, in chapter 6, solving travelling salesman problem with RRAM crossbar array was proposed as a future work, where the stochasticity in RRAMs can be used to make deep learning more accurate and affordable.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[3] D. A. Reed and J. Dongarra, "Exascale Computing and Big Data," *Commun ACM*, vol. 58, no. 7, pp. 56–68, Jun. 2015.

[4] J. von Neumann, "First draft of a report on the EDVAC," *IEEE Ann. Hist. Comput.*, vol. 15, no. 4, pp. 27–75, 1993.

[5] S. A. McKee, "Reflections on the memory wall," in *Proceedings of the first conference on computing frontiers on Computing frontiers - CF'04*, Ischia, Italy, 2004, p. 162.

[6] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[7] C.-S. Poon and K. Zhou, "Neuromorphic Silicon Neurons and Large-Scale Neural Networks: Challenges and Opportunities," *Front. Neurosci.*, vol. 5, 2011.

[8] G. Indiveri *et al.*, "Neuromorphic Silicon Neuron Circuits," *Front. Neurosci.*, vol. 5, 2011.

[9] S. Yu, Ed., *Neuro-inspired Computing Using Resistive Synaptic Devices*. Cham: Springer International Publishing, 2017.

[10] P. A. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, p. 668, Aug. 2014.

[11] C.-H. Kim *et al.*, "Emerging memory technologies for neuromorphic computing," *Nanotechnology*, vol. 30, no. 3, p. 032001, Jan. 2019.

[12] L. Chua, "Memristor-The missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[13] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

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

[15] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, May 2015.

[16] C. Li *et al.*, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," *Nat. Commun.*, vol. 9, no. 1, Dec. 2018.

[17] G. Bi and M. Poo, "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, Dec. 1998.

[18] S. Kim, S. Choi, and W. Lu, "Comprehensive Physical Model of Dynamic Resistive Switching in an Oxide Memristor," *ACS Nano*, vol. 8, no. 3, pp. 2369–2376, Mar. 2014.

[19] J. R. Jameson *et al.*, "(Invited) Conductive Bridging RAM (CBRAM): Then, Now, and Tomorrow," *ECS Trans.*, vol. 75, no. 5, pp. 41–54, Sep. 2016.
[20] R. Fackenthal *et al.*, "19.7 A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, San Francisco, CA, USA, 2014, pp. 338–339.
[21] Y. Yang and W. Lu, "Nanoscale resistive switching devices: mechanisms and modeling," *Nanoscale*, vol. 5, no. 21, p. 10076, 2013.
[22] Y. Yang, P. Gao, S. Gaba, T. Chang, X. Pan, and W. Lu, "Observation of conducting filament growth in nanoscale resistive memories," *Nat. Commun.*, vol. 3, no. 1, Jan. 2012.
[23] B. Chen, F. Cai, J. Zhou, W. Ma, P. Sheridan, and W. D. Lu, "Efficient in-memory computing architecture based on crossbar arrays," in *2015 IEEE International Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2015, pp. 17.5.1-17.5.4.
[24] M. A. Zidan, Y. Jeong, J. H. Shin, C. Du, Z. Zhang, and W. D. Lu, "Field-Programmable Crossbar Array (FPCA) for Reconfigurable Computing," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 4, pp. 698–710, Oct. 2018.
[25] S. Yu *et al.*, "Binary neural network with 16 Mb RRAM macro chip for classification and online training," in *2016 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2016, pp. 16.2.1-16.2.4.
[26] S. H. Jo, K.-H. Kim, and W. Lu, "Programmable Resistance Switching in Nanoscale Two-Terminal Devices," *Nano Lett.*, vol. 9, no. 1, pp. 496–500, Jan. 2009.
[27] S. Yu, Ximeng Guan, and H.-S. P. Wong, "On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, monte carlo simulation, and experimental characterization," in *2011 International Electron Devices Meeting*, Washington, DC, USA, 2011, pp. 17.3.1-17.3.4.
[28] S. Gaba, P. Sheridan, J. Zhou, S. Choi, and W. Lu, "Stochastic memristive devices for computing and neuromorphic applications," *Nanoscale*, vol. 5, no. 13, p. 5872, 2013.
[29] P. Knag, W. Lu, and Z. Zhang, "A Native Stochastic Computing Architecture Enabled by Memristors," *IEEE Trans. Nanotechnol.*, vol. 13, no. 2, pp. 283–293, Mar. 2014.
[30] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "Stochastic learning in oxide binary synaptic device for neuromorphic computing," *Front. Neurosci.*, vol. 7, 2013.
[31] Ö. Türel, J. H. Lee, X. Ma, and K. K. Likharev, "Neuromorphic architectures for nanoelectronic circuits," *Int. J. Circuit Theory Appl.*, vol. 32, no. 5, pp. 277–302, Sep. 2004.
[32] G. Indiveri and S.-C. Liu, "Memory and Information Processing in Neuromorphic Systems," *Proc. IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015.
[33] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.
[34] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*, Fourth edition. Stamford, CT, USA: Cengage Learning, 2015.
[35] H. Moon and P. J. Phillips, "Computational and Performance Aspects of PCA-Based Face-Recognition Algorithms," *Perception*, vol. 30, no. 3, pp. 303–321, Mar. 2001.
[36] M. Ringnér, "What is principal component analysis?," *Nat. Biotechnol.*, vol. 26, no. 3, pp. 303–304, Mar. 2008.
[37] D. Reich, A. L. Price, and N. Patterson, "Principal component analysis of genetic data," *Nat. Genet.*, vol. 40, no. 5, pp. 491–492, May 2008.
[38] J. Khan *et al.*, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nat. Med.*, vol. 7, no. 6, pp. 673–679, Jun. 2001.

[39] J. de Haes, F. van Knippenberg, and J. Neijt, "Measuring psychological and physical distress in cancer patients: structure and application of the Rotterdam Symptom Checklist," *Br. J. Cancer*, vol. 62, no. 6, pp. 1034–1038, Dec. 1990.

[40] S. Choi, J. H. Shin, J. Lee, P. Sheridan, and W. D. Lu, "Experimental Demonstration of Feature Extraction and Dimensionality Reduction Using Memristor Networks," *Nano Lett.*, vol. 17, no. 5, pp. 3113–3118, May 2017.

[41] A. Adamatzky and L. O. Chua, Eds., *Memristor networks*. Cham ; New York: Springer, 2014.

[42] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," *Nat. Nanotechnol.*, vol. 12, no. 8, pp. 784–789, Aug. 2017.

[43] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nat. Commun.*, vol. 4, no. 1, p. 2072, Dec. 2013.

[44] S. Yu, X. Guan, and H.-S. P. Wong, "Conduction mechanism of TiN/HfO_x/Pt resistive switching memory: A trap-assisted-tunneling model," *Appl. Phys. Lett.*, vol. 99, no. 6, p. 063507, Aug. 2011.

[45] T. Chang, S.-H. Jo, K.-H. Kim, P. Sheridan, S. Gaba, and W. Lu, "Synaptic behaviors and modeling of a metal oxide memristive device," *Appl. Phys. A*, vol. 102, no. 4, pp. 857–863, Mar. 2011.

[46] M.-J. Lee *et al.*, "A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta2O5−x/TaO2−x bilayer structures," *Nat. Mater.*, vol. 10, no. 8, pp. 625–630, Aug. 2011.

[47] Y. Yang, P. Sheridan, and W. Lu, "Complementary resistive switching in tantalum oxide-based resistive memory devices," *Appl. Phys. Lett.*, vol. 100, no. 20, p. 203112, May 2012.

[48] D. S. Jeong, B. J. Choi, and C. S. Hwang, "Electroforming Processes in Metal Oxide Resistive-Switching Cells," in *Resistive Switching*, D. Ielmini and R. Waser, Eds. Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, 2016, pp. 289–316.

[49] S. Choi, J. Lee, S. Kim, and W. D. Lu, "Retention failure analysis of metal-oxide based resistive memory," *Appl. Phys. Lett.*, vol. 105, no. 11, p. 113510, Sep. 2014.

[50] S. Kim, S. Choi, J. Lee, and W. D. Lu, "Tuning Resistive Switching Characteristics of Tantalum Oxide Memristors through Si Doping," *ACS Nano*, vol. 8, no. 10, pp. 10262–10269, Oct. 2014.

[51] S. Kim *et al.*, "Physical electro-thermal model of resistive switching in bi-layered resistance-change memory," *Sci. Rep.*, vol. 3, no. 1, p. 1680, Dec. 2013.

[52] A. Prakash *et al.*, "Improvement of Uniformity of Resistive Switching Parameters by Selecting the Electroformation Polarity in IrOx/TaOx/WOx/W Structure," *Jpn. J. Appl. Phys.*, vol. 51, pp. 4-6, Apr. 2012.

[53] M. Azzaz *et al.*, "Endurance/Retention Trade Off in HfOx and TaOx Based RRAM," in *2016 IEEE 8th International Memory Workshop (IMW)*, Paris, France, 2016, pp. 1–4.

[54] W. Kim *et al.*, "Nonlinearity analysis of TaOx redox-based RRAM," *Microelectron. Eng.*, vol. 154, pp. 38–41, Mar. 2016.

[55] J. T. Qiu, S. Samanta, M. Dutta, S. Ginnaram, and S. Maikap, "Controlling Resistive Switching by Using an Optimized MoS_2 Interfacial Layer and the Role of Top Electrodes on Ascorbic Acid Sensing in TaO_x -Based RRAM," *Langmuir*, vol. 35, no. 11, pp. 3897–3906, Mar. 2019.

[56] Y. Pan *et al.*, “Microscopic origin of read current noise in TaOx-based resistive switching memory by ultra-low temperature measurement,” *Appl. Phys. Lett.*, vol. 108, no. 15, p. 153504, Apr. 2016.
[57] S. Choi, P. Sheridan, and W. D. Lu, “Data Clustering using Memristor Networks,” *Sci. Rep.*, vol. 5, no. 1, Sep. 2015.
[58] T. D. Sanger, “Optimal unsupervised learning in a single-layer linear feedforward neural network,” *Neural Netw.*, vol. 2, no. 6, pp. 459–473, Jan. 1989.
[59] E. Oja, “Simplified neuron model as a principal component analyzer,” *J. Math. Biol.*, vol. 15, no. 3, pp. 267–273, Nov. 1982.
[60] W. H. Wolberg and O. L. Mangasarian, “Multisurface method of pattern separation for medical diagnosis applied to breast cytology.,” *Proc. Natl. Acad. Sci.*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990.
[61] J. Guy *et al.*, “Experimental and theoretical understanding of Forming, SET and RESET operations in Conductive Bridge RAM (CBRAM) for memory stack optimization,” in *2014 IEEE International Electron Devices Meeting*, San Francisco, CA, USA, 2014, pp. 6.5.1-6.5.4.
[62] D. Ielmini, “Filamentary-switching model in RRAM for time, energy and scaling projections,” in *2011 International Electron Devices Meeting*, Washington, DC, USA, 2011, pp. 17.2.1-17.2.4.
[63] S. Gaba, F. Cai, J. Zhou, and W. D. Lu, “Ultralow Sub-1-nA Operating Current Resistive Memory With Intrinsic Non-Linear Characteristics,” *IEEE Electron Device Lett.*, vol. 35, no. 12, pp. 1239–1241, Dec. 2014.
[64] J. Zhou, F. Cai, Q. Wang, B. Chen, S. Gaba, and W. D. Lu, “Very Low-Programming-Current RRAM With Self-Rectifying Characteristics,” *IEEE Electron Device Lett.*, vol. 37, no. 4, pp. 404–407, Apr. 2016.
[65] H. D. Lee *et al.*, “Integration of 4F2 selector-less crossbar array 2Mb ReRAM based on transition metal oxides for high density memory applications,” in *2012 Symposium on VLSI Technology (VLSIT)*, Honolulu, HI, USA, 2012, pp. 151–152.
[66] C. Xu *et al.*, “Overcoming the challenges of crossbar resistive memory architectures,” in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Burlingame, CA, USA, 2015, pp. 476–488.
[67] Y. Y. Chen *et al.*, “Endurance/Retention Trade-off on HfO2/Metal Cap 1T1R Bipolar RRAM,” *IEEE Trans. Electron Devices*, vol. 60, no. 3, pp. 1114–1121, Mar. 2013.
[68] J. H. Shin, Q. Wang, and W. D. Lu, “Self-Limited and Forming-Free CBRAM Device With Double Al_2O_3 ALD Layers,” *IEEE Electron Device Lett.*, vol. 39, no. 10, pp. 1512–1515, Oct. 2018.
[69] T. Tsuruoka, K. Terabe, T. Hasegawa, I. Valov, R. Waser, and M. Aono, “Effects of Moisture on the Switching Characteristics of Oxide-Based, Gapless-Type Atomic Switches,” *Adv. Funct. Mater.*, vol. 22, no. 1, pp. 70–77, Jan. 2012.
[70] B. G. Willis and D. V. Lang, “Oxidation mechanism of ionic transport of copper in SiO2 dielectrics,” *Thin Solid Films*, vol. 467, no. 1–2, pp. 284–293, Nov. 2004.
[71] T. Tsuruoka *et al.*, “Redox Reactions at Cu,Ag/Ta_2O_5 Interfaces and the Effects of Ta_2O_5 Film Density on the Forming Process in Atomic Switch Structures,” *Adv. Funct. Mater.*, vol. 25, no. 40, pp. 6374–6381, Oct. 2015.

[72] Jiantao Zhou, Kuk-Hwan Kim, and Wei Lu, "Crossbar RRAM Arrays: Selector Device Requirements During Read Operation," *IEEE Trans. Electron Devices*, vol. 61, no. 5, pp. 1369–1376, May 2014.

[73] Sungho Kim, Jiantao Zhou, and W. D. Lu, "Crossbar RRAM Arrays: Selector Device Requirements During Write Operation," *IEEE Trans. Electron Devices*, vol. 61, no. 8, pp. 2820–2826, Aug. 2014.

[74] Y. Zhang, J. A. Bertrand, R. Yang, S. M. George, and Y. C. Lee, "Electroplating to visualize defects in Al2O3 thin films grown using atomic layer deposition," *Thin Solid Films*, vol. 517, no. 11, pp. 3269–3272, Apr. 2009.

[75] S. M. George, "Atomic Layer Deposition: An Overview," *Chem. Rev.*, vol. 110, no. 1, pp. 111–131, Jan. 2010.

[76] Youqin. Xie and C. O. Huber, "Electrocatalysis and amperometric detection using an electrode made of copper oxide and carbon paste," *Anal. Chem.*, vol. 63, no. 17, pp. 1714–1719, Sep. 1991.

[77] H. B. Lv *et al.*, "Improvement of Endurance and Switching Stability of Forming-Free CuxO RRAM," in *2008 Joint Non-Volatile Semiconductor Memory Workshop and International Conference on Memory Technology and Design*, Opio, France, 2008, pp. 52–53.

[78] H. Häkkinen, M. Moseler, and U. Landman, "Bonding in Cu, Ag, and Au Clusters: Relativistic Effects, Trends, and Surprises," *Phys. Rev. Lett.*, vol. 89, no. 3, p. 033401, Jun. 2002.

[79] D.-Y. Cho, S. Tappertzhofen, R. Waser, and I. Valov, "Bond nature of active metal ions in SiO2-based electrochemical metallization memory cells," *Nanoscale*, vol. 5, no. 5, p. 1781, 2013.

[80] M. Takeuchi, G. Martra, S. Coluccia, and M. Anpo, "Evaluation of the Adsorption States of H_2O on Oxide Surfaces by Vibrational Absorption: Near- and Mid-Infrared Spectroscopy," *J. Infrared Spectrosc.*, vol. 17, no. 6, pp. 373–384, Dec. 2009.

[81] M. Specht, M. Städele, S. Jakschik, and U. Schröder, "Transport mechanisms in atomic-layer-deposited Al2O3 dielectrics," *Appl. Phys. Lett.*, vol. 84, no. 16, pp. 3076–3078, Apr. 2004.

[82] R. Stratton, "Volt-current characteristics for tunneling through insulating films," *J. Phys. Chem. Solids*, vol. 23, no. 9, pp. 1177–1190, Sep. 1962.

[83] S. Ambrogio, S. Balatti, S. Choi, and D. Ielmini, "Impact of the Mechanical Stress on Switching Characteristics of Electrochemical Resistive Memory," *Adv. Mater.*, vol. 26, no. 23, pp. 3885–3892, Jun. 2014.

[84] P. Auerkari, "Mechanical and physical properties of engineering alumina ceramics," VTT Manuf. Technol. Res. Notes, Tech. Res. Center Finland, Espoo, Finland, 2016, vol. 1792.

[85] A. Belmonte, A. Fantini, A. Redolfi, M. Houssa, M. Jurczak, and L. Goux, "Optimization of the write algorithm at low-current in Cu/Al2O3-based conductive-bridge RAM," in *2015 45th European Solid State Device Research Conference (ESSDERC)*, Graz, Austria, 2015, pp. 114–117.

[86] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[87] M. A. Zidan *et al.*, "A general memristor-based partial differential equation solver," *Nat. Electron.*, vol. 1, no. 7, pp. 411–420, Jul. 2018.

[88] C.-C. Chang *et al.*, "Challenges and opportunities toward online training acceleration using RRAM-based hardware neural network," in *2017 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, USA, 2017, pp. 11.6.1-11.6.4.
[89] W.-H. Chen *et al.*, "A 16Mb dual-mode ReRAM macro with sub-14ns computing-in-memory and memory functions enabled by self-write termination scheme," in *2017 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, 2017, pp. 28.2.1-28.2.4.
[90] J. H. Shin, Y. J. Jeong, M. A. Zidan, Q. Wang, and W. D. Lu, "Hardware Acceleration of Simulated Annealing of Spin Glass by RRAM Crossbar Array," in *2018 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, 2018, pp. 3.3.1-3.3.4.
[91] F. Barahona, "On the computational complexity of Ising spin glass models," *J. Phys. Math. Gen.*, vol. 15, no. 10, pp. 3241–3253, Oct. 1982.
[92] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model," *Phys. Rev. E*, vol. 58, no. 5, pp. 5355–5363, Nov. 1998.
[93] S. Matsubara *et al.*, "Ising-Model Optimizer with Parallel-Trial Bit-Sieve Engine," in *Complex, Intelligent, and Software Intensive Systems*, vol. 611, L. Barolli and O. Terzo, Eds. Cham: Springer International Publishing, 2018, pp. 432–438.
[94] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-Inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer," *Front. Phys.*, vol. 7, p. 48, Apr. 2019.
[95] M. Hu *et al.*, "Dot-product engine for neuromorphic computing: programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proceedings of the 53rd Annual Design Automation Conference on - DAC '16*, Austin, Texas, 2016, pp. 1–6.
[96] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[97] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
[98] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1–9.
[99] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.
[100] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient Processing of Deep Neural Networks: A Tutorial and Survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[101] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *ArXiv151000149 Cs*, Oct. 2015.
[102] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1135–1143.
[103] Q. He *et al.*, "Effective Quantization Methods for Recurrent Neural Networks," *ArXiv161110176 Cs*, Nov. 2016.
[104] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations," p. 30.

[105] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3123–3131.
[106] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014, pp. 10–14.
[107] N. Brunel, F. Carusi, and S. Fusi, "Slow stochastic Hebbian learning of classes of stimuli in a recurrent neural network," *Netw. Comput. Neural Syst.*, vol. 9, no. 1, pp. 123–152, Jan. 1998.
[108] W. Senn and S. Fusi, "Convergence of stochastic learning in perceptrons with binary synapses," *Phys. Rev. E*, vol. 71, no. 6, p. 061907, Jun. 2005.
[109] M. Höhfeld and S. E. Fahlman, "Probabilistic rounding in neural network learning with limited precision," *Neurocomputing*, vol. 4, no. 6, pp. 291–299, Dec. 1992.
[110] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep Learning with Limited Numerical Precision," Proceedings of the 32nd International Conference on Machine Learning, in PMLR 37:1737-1746, 2015
[111] X. Sun, S. Yin, X. Peng, R. Liu, J. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1423–1428.
[112] Z. Dong *et al.*, "Convolutional Neural Networks Based on RRAM Devices for Image Recognition and Online Learning Tasks," *IEEE Trans. Electron Devices*, vol. 66, no. 1, pp. 793–801, Jan. 2019.
[113] Y. Zhou, S. Redkar, and X. Huang, "Deep learning binary neural network on an FPGA," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 281–284.
[114] S. Menzel, P. Kaupmann, and R. Waser, "Understanding filamentary growth in electrochemical metallization memory cells using kinetic Monte Carlo simulations," *Nanoscale*, vol. 7, no. 29, pp. 12673–12681, 2015.
[115] J. P. Hayes, "Introduction to stochastic computing and its challenges," in *Proceedings of the 52nd Annual Design Automation Conference on - DAC '15*, San Francisco, California, 2015, pp. 1–3.
[116] A. Chen, "Utilizing the Variability of Resistive Random Access Memory to Implement Reconfigurable Physical Unclonable Functions," *IEEE Electron Device Lett.*, vol. 36, no. 2, pp. 138–140, Feb. 2015.
[117] H. Jiang *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nat. Commun.*, vol. 8, no. 1, Dec. 2017.
[118] G. Pedretti *et al.*, "Stochastic Learning in Neuromorphic Hardware via Spike Timing Dependent Plasticity With RRAM Synapses," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 8, no. 1, pp. 77–85, Mar. 2018.
[119] R. Naous, M. Al-Shedivat, and K. N. Salama, "Stochasticity Modeling in Memristors," *IEEE Trans. Nanotechnol.*, vol. 15, no. 1, pp. 15–28, Jan. 2016.
[120] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[121] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016.

[122] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," p. 30.
[123] M. R. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, 27. print. New York [u.a]: Freeman, 2009.
[124] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, vol. 2, 2014.