MEMRISTOR: MODELING, SIMULATION AND USAGE IN NEUROMORPHIC COMPUTATION

by

Miao Hu

M.S. in Electrical Engineering,

Polytechnic Institute of New York University, 2011

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Miao Hu

It was defended on

June 9th, 2014

and approved by

Hai Li, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Luis F. Chaparro, Ph.D., Associate Professor Emeritus, Department of Electrical and Computer Engineering

Yiran Chen, Ph.D., Associate Professor, Department of Electrical and Computer Engineering

William E. Stanchina, Ph.D., Chairman and Professor, Department of Electrical and Computer Engineering

Qing Wu, Ph.D., Air Force Research Lab

Dissertation Director: Hai Li, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

MEMRISTOR: MODELING, SIMULATION AND USAGE IN NEUROMORPHIC COMPUTATION

Miao Hu, PhD

University of Pittsburgh, 2014

Memristor, the fourth passive circuit element, has attracted increased attention from various areas since the first real device was discovered in 2008. Its distinctive characteristic to record the historic profile of the voltage/current through itself creates great potential in future circuit design. Inspired by its high scalability, ultra low power consumption and similar functionality to biology synapse, using memristor to build high density, high power efficiency neuromorphic circuits becomes one of most promising and also challenging applications. The challenges can be concluded into three levels: First, at device level, memristors as emerging nano devices greatly suffer from process variations. Some materials also demonstrate stochastic switching behavior. All these impacts should be considered and efficiently modeled. Considering that memristor devices can be realized in different memristive materials, a general modeling method of process variations and stochastic switching behavior that is independent with device's physical models is preferred; Second, at circuit level, cross-point array fabrication is usually employed in realizing high density memristive circuits. However, current cross-point simulation is either not accurate enough or not efficient for large scale simulation, like 1024×1024 array. An efficient and flexible cross-point array simulation platform is necessary to explore the performance of memristive cross-point arrays in neuromorphic computation; Third, at application level, how to apply and evaluate the emerging memristor in neuromorphic circuit design is unclear. Many modification need to be carried out in neural network algorithms and circuit realizations to take memristor's advantages as well as alleviate its shortages. Generally, although memristor is widely accepted as one of

the most important keys for future high efficient, low power computation, how to design high performance realistic memristive neuromorphic circuits remains a open question, which is the major study topic in this thesis.

At device level, I designed a series of models for general memristors to study the impact of process variations and stochastic behavior on memristor device and memristive circuits. First I analyzed the impact of the geometry variations on the electrical properties of TiO_2-based memristors, including line edge roughness (LER) and thickness fluctuations (TF). A simple algorithm was proposed to generate a large volume of geometry variation-aware three-dimensional device structures for Monte-Carlo simulations. Then a statistical model is extracted accordingly. Simulations show that the statistical model is 3$\sim$4 magnitude faster than the Monte-Carlo simulation method, with only $\sim$2% accuracy degradation. In addition, the stochastic switching behaviors have been widely observed in metal oxide memristors. To facilitate the investigation on memristor-based hardware implementation, we built a stochastic behavior model of TiO_2 memristive devices based on the real experimental results. These models could be extended to general memristors to study the impact of process variations and stochastic behavior on memristor operations.

At circuit level, I integrated a full set of device models and variation models into the cross-point array structure to evaluate its performance. By efficiently formulating the entire cross-point array, the simulator is 2$\sim$3 orders of magnitude faster than traditional SPICE simulator without accuracy degradation. In this way, designers could look into the memristive cross-point array with scale and accuracy which has never been achieved before. The results firstly show how sneaking current leakage map changes in crossbar arrays, and quantitatively evaluate the impact of circuit, device and variation parameters. This simulator paves the road for further application-level studies.

At application level, I focused our study on memristor-based neuromorphic circuits with the models and tools developed in front. I first proposed a few low power neuromorphic components with memristors, including memristor-based synapse "macro cell", memristor-based stochastic neuron and memristor-based spatio-temporal(MST) synapse. These designs employee memristors to achieve ultra high power efficiency, and have high tolerance to process variations and stochastic behaviours. In addition, all these components are friendly to

cross-point array integration so they could be applied to large scale neuromorphic circuit applications.

I also extended my study to large scale neuromorphic circuit for artificial neural network (ANN) applications. In one work, I explore the potential of a memristor crossbar array that functions as an auto-associative memory and apply it to Brain-State-in-a-Box (BSB) neural networks, which belongs to traditional neural networks for spatial pattern recognition. More specifically, the recall and training functions of a multi-answer character recognition process based on BSB model are studied. The robustness of the BSB circuit is analyzed and evaluated based on extensive Monte-Carlo simulations, considering input defects, process variations, and electrical fluctuations. The results show that the hardware-based training scheme proposed in the work can alleviate and even cancel out majority of the noise issue. In another work, I further studied the memristor-based hardware realization of spiking neural network (SNN) for temporal pattern learning, which is the third generation and also the latest generation of neural networks. A temporal pattern learning application is applied to evaluate the performance of MST synapses in neural networks. The learning ability when utilizing STDP and ReSuMe learning rules were examined. The simulation results show that MST synapses can realize online spike-timing-based learning at nano-second scale with an average energy consumption of $36.7pJ$ and $64.0pJ$ per spike for STDP and ReSuMe learning rules, respectively. The energy consumption in recall process is only $14.6pJ$ per spike. The simple design structure, enhanced functionality, fast execution, and high energy efficiency of MST synapses potentially lead to highly scalable neuromorphic hardware designs.

In conclusion, in this thesis I introduce a complete hardware design flow of memristor-based neuromorphic circuits from device level to application level. The impact of variations, device parameters and circuit operation schemes are studied at different levels to give a accurate and complete evaluation of memristors' performance in neuromorphic circuits. Moreover, a series of modifications and designs at different levels are also carried out to optimize the performance of memristor-based neuromorphic circuits. Our result shows that memristor-based neuromorphic computation could be $\sim$ 2orders of magnitude power efficient than traditional CMOS computation, and is an ideal platform for "brain-like" high performance computation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

1.1 MOTIVATION

In the development history of microprocessors, the continuation of Moore's law was guaranteed first by the increase of transistor integration density and then by multi-core technology Moore et al. (1965). Although in von Neumann computing systems, computing efficiency of solid state circuits keeps improving by following technology scaling, the data transportation (communication) efficiency between CPU cores and storage systems starts drifting down and dominating the energy consumption of the entire system. This phenomenon is referred to as the *memory wall* Wulf and McKee (1995).

Information is generally stored in solid state circuits as the electrical charge on capacitors, *e.g.*, in SRAM and registers. Data transportation is realized by moving the charge from one location to another. Thus, the communication cost is determined by two factors, *i.e.*, the amount of the charge relocated and the distance between the source and the destination. Reducing the communication cost can be realized by decreasing these two factors, such as lowering the power supply voltage in dynamic voltage scaling Pillai and Shin (2001) and subthreshold designs Wang et al. (2006), or shortening the distance between the computing cores and memories in distributed systems Bertsekas and Tsitsiklis (1989). Notably, these methods can help alleviate the issue but can not *break* the memory wall because computation and memory are always separated in von Neumann computing systems. Moreover, as transistors reach the quantum regime, the scaling benefit slows down and may eventually diminish Haensch et al. (2006). Based on the Gedanken model from thermodynamics, once the critical device dimension reduces to $5nm$ and below, the minimum energy of a logic bit switching would dramatically increase Zhirnov et al. (2003). These facts indicate that traditional

von Neumann computing systems based on CMOS technologies will enjoy less performance increment and energy efficiency from device scaling as well as performance enhancement. To date, a lot of research efforts have been investigated on novel computing architectures for more powerful computation capability and the emerging devices offering increased functional diversity ITRS (2013).

Among various novel computing architectures, neuromorphic computation system, or say, "brain-like" computation system, is taken as one of the most promising candidate for next-generation high performance low power computation systems. In recent years, neuromorphic hardware systems built upon the conventional CPU, GPU Oh and Jung (2004), or FPGA Omondi and Rajapakse (2006) gained a great deal of attention from industry, government and academia. Such systems can potentially provide the capabilities of biological perception and cognitive information processing within a compact and energy-efficient platform Camilleri et al. (2007); Partzsch and Schuffny (2011). For example, the spike-timing-based computation (a.k.a neuromorphic computing) inspired by the working mechanism of human brains effectively reduces the data communication cost and consequently, achieves very high computation efficiency: On the one hand, since both the frequency of the spikes and their relative timing carry on the transmitted information, the spikes can be very short and sparse that minimizes the amount of the relocated electrical charge; On the other hand, neuromorphic computing also minimizes the data communication distance by distributing the data into the memories (*i.e.*, synapses) close to the associated computing units (*i.e.*, neurons) throughout the entire system. For example, *TrueNorth* – the latest spike-timing-based neuromorphic hardware prototyped by IBM achieved an extremely low energy consumption of $45pJ$ per spike in data communication Merolla et al. (2011).

Conventional CMOS implementation of synapse designs suffer from large area, high power consumption, and low-level parallelism. In TrueNorth, for example, each neurosynaptic core contains an array of 1024×256 synapses built on SRAM cells. The majority of system energy was consumed on retaining synaptic weights, programming synapses in learning process, and reading out the stored weights in recall function. Moreover, since SRAM array cannot support a truly parallel execution, TrueNorth runs at only 1KHz even though the operating frequency within each neurosynaptic core is 1MHz Seo et al. (2011). Algorithm

enhancement can alleviate the situation but cannot fundamentally resolve it. Thus, a more efficient hardware-level solution is necessary.

As one of the emerging devices, memristor has attracted great attention from various areas since it is claimed as the fourth passive basic circuit element to complete the circuit theory. The existence of the memristor was predicted in circuit theory about forty year ago Chua (1971b). In 2008, the physical realization of a memristor was firstly demonstrated by HP Lab through a TiO_2 thin-film structure Strukov et al. (2008b). Afterwards, many memristive materials and devices have been rediscovered Yang et al. (2013). Intrinsically, a memristor behaves similarly to a synapse: it can remember the total electric charge/flux ever to flow through it Di Ventra et al. (2009); Chua (2011). Moreover, memristor-based memories can achieve a very high integration density of 100 Gbits/cm^2, a few times higher than flash memory technologies Ho et al. (2009). These unique properties make it a promising device for massively-parallel, large-scale neuromorphic systems Xia et al. (2009); Jo et al. (2010).

All in all, memristor-based neuromorphic circuit is widely accepted as a promising solution to realize neuromorphic computation. However, as a technology in its infant stage, it still has many immaturities at current stage and requires careful analysis, the details of which will be explained in the following.

1.1.1 Device to device variation

As process technology shrinks down to decananometer (sub-50nm) scale, device parameter fluctuations incurred by process variations have become a critical issue affecting the electrical characteristics of devices Asenov et al. (2003). The situation in a memristive system could be even worse when utilizing the analog states of the memristors in design: variations of device parameters, e.g. the instantaneous memristance, can result in the shift of electrical responses, e.g. current. The deviation of the electrical excitations will affect memristance because the total charge through a memristor indeed is the historic behavior of its current profile. Previous works on memristor variation analysis mainly focused on its impacts on non-volatile memory design Ho et al. (2009). However, the systematic analysis and quantitative evaluation on how process variations affect the memristive behavior still needs to be done.

The device geometry variations significantly influence the electrical properties of nano-devices Roy et al. (2006). For example, the random uncertainties in lithography and patterning processes lead to the random deviation of line edge print-images from their ideal pattern, which is called line edge roughness (LER) Oldiges et al. (2000). Thickness fluctuation (TF) is caused by deposition processes in which mounds of atoms form and coarsen over time. As technology shrinks, the geometry variations do not decrease accordingly. In this work, we propose an algorithm to generate a large volume of three-dimensional memristor structures to mimic the geometry variations for Monte-Carlo simulations. The LER model is based on the latest LER characterization method for electron beam lithography (EBL) technology from top-down scanning electron microscope (SEM) measurement Jiang et al. (2009). Besides the geometry variations, other process variations such as random discrete doping (RDD) could also result in the fluctuations of the electrical properties of devices. However, because the existing memristors are all based on the thin film deposition technology, the local randomness of RDD is not as significant as geometry variations and is not covered in this work.

1.1.2 Device controllability and stochastic behaviour

At current stage, a large gap exists between the theoretical memristor characteristics and the experimental data obtained from real device development, raising severe concerns in feasibility of memristor-based hardware design. For instance, the memristor theory expresses a continuous and stable memristance change. This is the foundation of using memristors to represent synapses in bio-inspired system. Though an arbitrary intermediate state can be obtained by carefully setting current compliance and period in a single metal oxide memristor, *e.g.*, TiO_2 device, the corresponding realization at large scale, *e.g.*, cross-point array, is very difficult after including intrinsic design constrains, process variations, environmental fluctuations, etc. Keeping a memristor in its ON or OFF state (corresponding to R_{on} or R_{off}), on the contrary, is much more controllable. Thus, more precisely, most memristors nowadays are utilized as memristive switches Medeiros-Ribeiro et al. (2011).

Moreover, memristor behaves stochastically and hence even a single memristive device demonstrates large variations in performance. More specific, the static states of a single memristive switch, i.e., R_{on} and R_{off}, are not fixed values, but have large variations with skewed distributions and heavy tails Yi et al. (2011). The switching mechanism of a memristive switch, that is, its dynamic behavior, performs as a stochastic process Yang et al. (2009). Previous statistical analyses on memristors were limited to the binary switching as storage elements. However, as an analog device in neural network application, it is more important to understand and model memristor's stochastic characteristics.

1.1.3 Large scale cross-point array analysis

Memristor has demonstrated the similar functions as synapse. Accordingly, memristor cross-point array could be the densest realization of the synapse network among groups of neurons. In human brains, one neuron is usually connected to over thousands of other neurons, which means very large scale cross-point arrays are necessary for direct mapping of synapse network among neurons. However, to simulate the synapse behavior of large scale memristor cross-point array is extremely time-consuming in traditional circuit simulators that were designed for memory applications but not neural network applications. Thus, a circuit simulation platform specially designed for cross-point array as synapse network is an necessary bridge to connect the memristor-based synapse to the memristor-based neuromorphic circuit designs.

1.1.4 Circuit performance analysis base on applications

With enough understanding of memristor devices and memristor cross-point arrays, we can start to explore the realization of ANN algorithms in memristor-based neuromorphic circuits. The study in this area still remains nearly blank at current stage. Though some previous works have analyzed the simple learning behaviors of memristor based synaptic circuits Snider (2008); Jo et al. (2010); Ambrogio et al. (2013); Thomas (2013), the study of the input-output feature of memristor cross-point array is still missing. Moreover, the impact of process variations, signal fluctuations, defects and other physical limitations on memristor-based neuromorphic circuit performance are not yet quantitatively studied.

Furthermore, even though single memristor has been demonstrated with similar functions as synapses, they are still far from even partially mimicking a real biological system. One example is reproducing ***spatio-temporal property***, which is an important synaptic feature denoting the capabilities of not only modulating the strength and rate of an input spike (*i.e.*, spatial weighting) but also adaptively adjusting the synaptic weight according to the relative timing relationship between input and output spikes (*i.e.*, temporal weighting). The requirement of the latter capability is based on the observation that information in biological visual systems is mainly carried by the timing of the first spike and the number of spikes Keat et al. (2001); Gollisch and Meister (2008).

Nowadays, the temporal weighting property usually is realized at software level by integrating temporal synapse models into spike-timing-based learning algorithms Maass (1997); Caporale and Dan (2008); Ponulak and Kasinski (2010). We note that most of synapse designs are based on only multi-level/analog resistance states of memristors by assuming a constant synaptic weight once learning is completed. Such a design without adaptive learning ability may realize either temporal or spatial memory function separately, but cannot offer the combined spatio-temporal property. Moreover, although STDP function has been widely demonstrated on memristors theoretically and experimentally, these designs often require precise neural signals supplied by carefully designed pulse modulators Snider (2008); Jo et al. (2010); Ambrogio et al. (2013); Thomas (2013). The incurred design cost is also generally unaffordable for a large-scale neural network implementation.

1.2 CONTRIBUTIONS

To comprehensively address the difficulties mentioned in Section 1.1, my study on memristor-based neuromorphic circuits is across device, circuit and application levels. Particularly, the contributions of this thesis can be concluded in the following aspects:

1.2.1 Memristor variation model

For device to device variation, I first investigate the impacts of geometry variations on the electrical properties of memristors and explore their implications to circuit design. An algorithm for fast generation of three-dimensional memristor structures is proposed to mimic the geometry variations incurred by EBL technology. The generated samples are used for Monte-Carlo simulations. Then I develop a fast statistical model to simulate the electrical properties of TiO_2 thin-film memristors. Starting with the theoretical model of a TiO_2 thin-film memristor, I explore the influences of geometry variations on the electrical parameters of the device. On top of that, a statistical model with the merits of both high accuracy and low runtime cost is proposed. Compared to the previous model, our statistical model significantly improves the runtime cost by $3 \sim 4$ orders of magnitude; and reduces the input data set down to a few variables. The simulation accuracy maintains within $\sim 2\%$ discrepancy in the whole working region of the memristor device.

1.2.2 Memristor stochastic model

For device stochastic behavior, I build a stochastic behavior model of TiO_2 memristive devices based on the real measurement results Medeiros-Ribeiro et al. (2011); Yi et al. (2011) in order to better facilitate the exploration of memristive switches in hardware implementation. The model bypasses material-related parameters while directly linking the device analog behavior to stochastic functions. Simulations show that the proposed stochastic device model fits well to the existing device measurement results.

1.2.3 Efficient memristor cross-point array simulator

For circuit simulations, I firstly proposed a full dynamic simulation platform of nonlinear memristor crossbar memory, where experimental verified memristor model, nonlinear selector model, stochastic variation model and possible process variations are considered. After taking the regular shape of crossbar array, the simulation process can be speed ed up by $2\sim3$ orders of magnitudes faster than in SPICE simulation via solving the nonlinear model in

MATLAB environment and pre-calculating its Jacobian matrix. All of these contribute to a simulation study on memristor cross-point array with accuracy and scale which has not been shown before. It can also give us a much deeper insight on the impact of sneak path current leakage, the impact of writing scheme, and the relationship between single memristor device performance and the corresponding crossbar performance.

1.2.4 Memristor-based neuromorphic circuit components

For memristor-based neuromorphic circuit applications, a few memristor-based neuromorphic circuit components are conducted, including the "macro cell" synapse design, memristor-based stochastic neuron and memristor-based spatio-temporal (MST)synapse.

"Macro cell" synapse: Macro cell is a practicable neuromorphic circuit design to overcome the gap between the theoretical and real characteristics of memristive devices. A macro cell design is composed of a group of parallel connected memristive switches. It obtains multiple logic states by leveraging device's stochastic behavior. The macro cell sacrifices the design density for better feasibility. However, it is still more efficient than traditional CMOS implementations through floating gates or capacitors Srinivasan et al. (2005). As a weight storage unit, macro cell synapse can be easily adapted into a cross-point array for high density weight storage. With aid of a feedback attempt scheme, the conductance of a macro cell quickly converges to the desired value.

Memristor-based stochastic neuron: A binary stochastic neuron design can be realized with a single memristive switch, using an external voltage to control its ON-OFF probability. The design can be extended to continuous stochastic neuron by replacing the memristive switch with the proposed macro cell. The number of switches in macro cell and the external voltage signal together control the mean and deviation of noise.

Memristor-based spatio-temporal (MST)synapse: Instead of simplifying synapse designs, we tend to enrich the functionalities of synapses and reduce the complexity of neurons, as illustrated in Fig. 1. The new approach is closer to the fact in biological nervous system and enables a more efficient hardware implementation. In particular, a *memristor-based spatio-temporal* (MST) synapse design was invented with the following features:

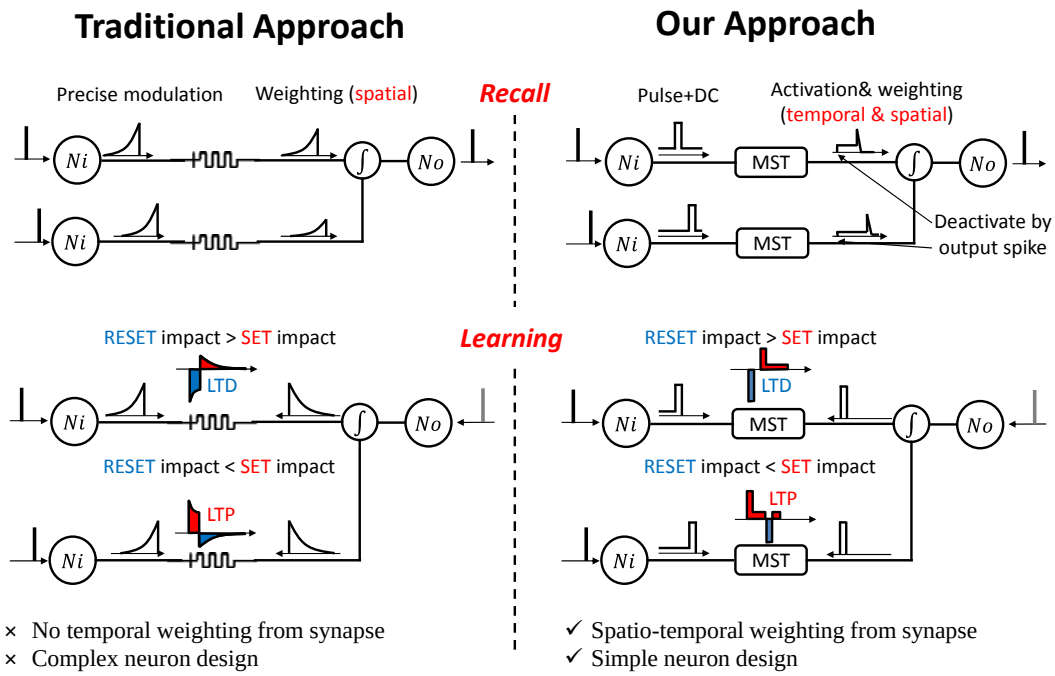

Figure 1: Comparison of traditional approach with MST synapse approach.

- *A simple and dense structure* that can be easily integrated in cross-point array for massive connections.
- *Relaxed requirements on the control precision of spike signals* so that design complexity of neurons reduces and communication energy efficiency improves.
- *Good support of typical synaptic properties*, including spike-timing-based recall and synaptic weight tuning.
- *Successful demonstration of spiking-timing-based learning ability* by using fundamental STDP Caporale and Dan (2008) and ReSuMe Ponulak and Kasinski (2010) learning rules.

1.2.5 Memristor-based neurmorphic circuit for spatial pattern recognition

In studying the input-output feature of memristor cross-point arrays, we found this typical array structure can naturally provide the capability of connection matrix storage and matrix-vector multiplication. Moreover, it offers a huge number of connections. Therefore, we exploit the application of the memristor cross-point arrays in neuromorphic hardware design

and use the Brain-State-in-a-Box (BSB) model Hu et al. (2012, 2013b), an auto-associative neural network, to illustrate the potential of memristor crossbars in complex and large scale pattern associations and classifications.

Two methods are proposed to configure the memristance of memristor cross-point arrays. The mapping method converts a software generated connection matrix into hardware memristor crossbars when memristors can be precisely configured. We present a fast approximation technique so that a matrix can be mapped to pure circuit element relations. Key design parameters and physical constraints have been extracted and carried into the study of the accuracy and robustness of the BSB circuit. Interestingly, the large random process variations of the memristive devices Hu et al. (2011b); Medeiros-Ribeiro et al. (2011) have little adverse impact, due to the inherent random noise tolerance of the BSB model.

The training method mimics the training process in the software algorithm and iteratively adjusts the memristor cross-point arrays to the required status. Many physical constraints in circuit implementations have been considered, including limited data access, limited accuracy of signal detection, non-ideal memristor characteristics Strukov et al. (2008b), process variations and defects. Our design generates the programming pattern for iterative training using the sign of input signals and the magnitude differences between the output signals and the expected outputs. By avoiding directly reading the memristance values of cross-point arrays, the proposed scheme significantly reduces the design complexity and avoids analog-to-digital converter (ADC) circuits. We demonstrate the effectiveness of our training scheme by performing the recall operation with the BSB recall circuit and comparing the results with those from software algorithms Lillo et al. (1994); Perfetti (1995); Park (2010).

1.2.6 Memristor-based spiking neurmorphic circuit for temporal pattern recognition

With the MST synapse which can mimic the spatio-temporal feature of synapse, we are able to build efficient memristor-based neuromorphic circuit to realize the spiking neural network for temporal learning algorithms. we studied the usage of MST synapses in spiking neural networks for a temporal pattern learning task with both STDP and ReSuMe learning rules.

Our result shows that the learning task can be accomplished at nano-second scale with on average $36.7pJ$ and $64.0pJ$ per learning spike for STDP and ReSuMe learning rules, respectively. And the recall energy consumption is only $14.6pJ$ per recall spike. Even with All the simulations are conducted with the latest experimentally-grounded TaO_x physical model from HP Labs Strachan et al. (2013), making great promise of the credibility of the work. Even the MST synapse design based on TaO_x devices has limited synaptic conductance tuning range, our result shows that it can still provide sufficient learning ability in neural network applications with assist of appropriate learning rules. Further increase of memristance range will alleviate the situation and makes more learning rules practical. Note that the micro model of TaO_x devices used in this work has a low resistance range of $70\Omega \sim 670\Omega$ Strachan et al. (2013). Applying nano-scale devices can significantly increase the memristance value to $100K\Omega \sim 1M\Omega$ Jo et al. (2010) and further reduce energy per spike to the sub-pJ region.

1.3 THESIS ROADMAP

The rest of the paper is organized as follows: Chapter 2 introduces preliminary knowledge on memristor, synapse and neural networks Chapter 3 describes the modeling work of memristors on device variation and stochastic behavior. Chapter 4 briefly explains the MATLAB-based memristor cross-point array simulator and verifies the result. Chapter 5 introduces the memristor-based neuromorphic circuit components. Chapter 6 details the implementation of BSB neural network with memristor-based neuromorphic circuit for spatial pattern recognition. Chapter 7 describes the implementation of spiking neural network with memristor-based neuromorphic circuit for temporal pattern recognition. Chapter 8 concludes the thesis.

2.0 PRELIMINARY

2.1 MEMRISTOR

2.1.1 Theoretical memristors

The original definition of the memristor is derived from circuit theory: besides resistor, capacitor and inductor, there must exist the fourth basic two-terminal element that uniquely defines the relationship between the magnetic flux (φ) and the electric charge (q) passing through the device Chua (1971a), or

$$d\varphi = M \cdot dq. \tag{2.1}$$

Considering that magnetic flux and electric charge are the integrals of voltage (V) and current (I) over time, respectively, the definition of the memristor can be generalized as:

$$\begin{cases} V = M(\omega, I) \cdot I \\ \dfrac{d\omega}{dt} = f(\omega, I) \end{cases} \tag{2.2}$$

Here, ω is a state variable; $M(\omega, I)$ represents the instantaneous memristance, which varies over time. For a "pure" memristor, neither $M(\omega, I)$ nor $f(\omega, I)$ is an explicit function of I Strukov et al. (2009).

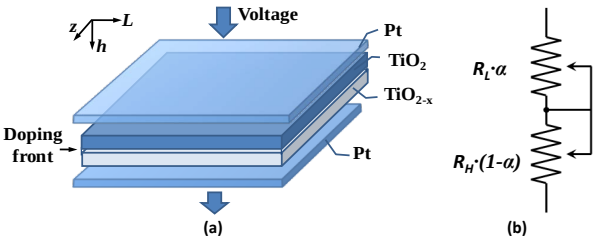

Figure 2: TiO_2 thin-film memristor. (a) structure, and (b) equivalent circuit.

2.1.2 Memristor devices

2.1.2.1 TiO_2 thin-film memristor In 2008, HP Lab demonstrated the first intentional memristive device by using a Pt/TiO_2/Pt thin-film structure Strukov et al. (2008a). The conceptual view is illustrated in Fig. 2(a): two metal wires on Pt are used as the top and bottom electrodes, and a thick titanium dioxide film is sandwiched in between. The stoichiometric TiO_2 with an exact 2:1 ratio of oxygen to titanium has a natural state as an insulator. However, if the titanium dioxide is lacking a small amount of oxygen, its conductivity becomes relatively high like a semiconductor. We call it oxygen-deficient titanium dioxide (TiO_{2-x}) Niu et al. (2010). The memristive function can be achieved by moving the doping front: A positive voltage applied on the top electrode can drive the oxygen vacancies into the pure TiO_2 part and therefore lower the resistance continuously. On the other hand, a negative voltage applied on the top electrode can push the dopants back to the TiO_{2-x} part and hence increase the overall resistance. For a TiO_2-based memristor, R_L (R_H) is used to denote the low (high) resistance when it is fully doped (undoped).

Fig. 2(b) illustrates a coupled variable resistor model for a TiO_2-based memristor, which is equivalent to two series-connected resistors. The overall resistance can be expressed as

$$M(w) = R_L \cdot \omega + R_H \cdot (1 - \omega). \tag{2.3}$$

Here ω ($0 \leq \omega \leq 1$) is the relative doping front position, which is the ratio of doping front position over the total thickness of TiO_2 thin-film.

The velocity of doping front movement $v(t)$, which is driven by the voltage applied across the memristor $V(t)$ can be expressed as

$$v(t) = \frac{d\omega}{dt} = \mu_v \cdot \frac{R_L}{h^2} \cdot \frac{V(t)}{M(\omega)} \tag{2.4}$$

where, μ_v is the equivalent mobility of dopants, h is the total thickness of the TiO_2 thin-film; and $M(\omega)$ is the total memristance when the relative doping front position is ω.

Bulk model is a general model derived from the mathematical definition of memristor, which assumes a flat doping front moving up or down. However, in reality, filamentary conduction has been observed in nano-scale semiconductors: the current travels through some high conducting filaments rather than passes the device evenly Kim et al. (2006)Kim et al. (2007). The doping front is formed so randomly that a few filaments dopes much faster than others, observed as hot spots on the device. This is called as *filament conduction phenomenon*. The way we solved the conflict between the bulk and filament models in this work can be explained as follows: when cutting the device into many tiny tiny filaments as we shall describe in Section 3.1.1, it is reasonable to assume a small flat doping front exists in each filament. Therefore, bulk model can be used for each small flat doping front movement.

Recent experiments showed that μ_v is not a constant but grows exponentially when the bias voltage goes beyond certain threshold voltage Strukov and Williams (2009). Nevertheless, the structure of TiO_2 memristor model, *i.e.*, Eq. (2.4), still remains valid.

2.1.2.2 Spintronic memristor The spintronic memristor relies on the existence of domain wall in MTJ free layer Lou et al. (2008) and properly controlling the domain wall. There have been many research activities on manipulating domain walls Parkin (2009)Tatara and Kohno (2004)O. Boulle and Klaui (2011). Very recently, NEC Lab reported the free layer switching though the domain wall movement Matsunaga et al. (2011), which indeed is a spintronic memristor.

Among all the spintronic memristive devices , the one based on magnetic tunneling junction (MTJ) could be the most promising one because of its simple structure Wang et al. (2009)Wang and Chen (2010). The basic structure of magnetic memristor could be either giant magneto-resistance (GMR) or tunneling magneto-resistance (TMR) MTJs. We choose

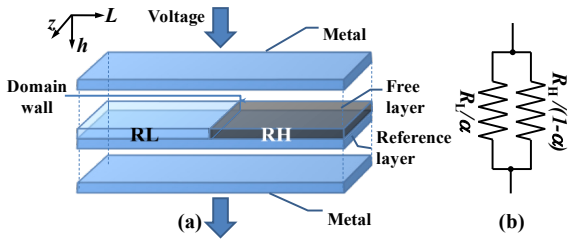

Figure 3: TMR-based spintronic memristor. (a) structure, and (b) equivalent circuit.

TMR-based structure shown in Fig. 3(a) as the objective of this work because it has a bigger difference between the upper and the lower bounds of total memristance (resistance).

An MTJ is composed of two ferromagnetic layers and an oxide barrier layer, e.g. MgO. The bottom ferromagnetic layer is called reference layer, of which the magnetization direction is fixed by coupling to a pinned magnetic layer. The top ferromagnetic layer called free layer is divided into two magnetic domains by a domain-wall: the magnetization direction of one domain is parallel to the reference layer's, while the magnetization direction of the other domain is anti-parallel to the reference layer's.

The movement of the domain wall is driven by the spin-polarized current, which passes through the two ferromagnetic layers. For example, applying a positive voltage on free layer can impel the domain wall to increase the length of the magnetic domain with a magnetization direction parallel to the reference layer's and hence reduce the MTJ resistance. On the other hand, applying a positive voltage on reference layer will reduce the length of the magnetic domain with a magnetization direction parallel to the reference layer's. Therefore, the MTJ resistance increases. If the width of the domain with the magnetization direction anti-parallel (parallel) to the reference layer's is compressed to close to zero, the memristor has the lowest (highest) resistance, denoted as R_L (R_H).

As shown in Fig. 3, the overall resistance of a TMR-base spintronic memristor can be modeled as two parallel connected resistors with resistances R_L/ω and $R_H/(1-\omega)$, respectively Wang and Chen (2010). This structure has also been experimentally proved Lou et al. (2008). Here ω ($0 \leq \omega \leq 1$) represents the relative domain wall position as the ratio of the

domain wall position (x) over the total length of the free layer (L). The overall memristance can be expressed as

$$M(\alpha) = \frac{R_L \cdot R_H}{R_H \cdot \omega + R_L(1-\omega)}. \tag{2.5}$$

How fast the domain-wall can move is mainly determined by the strength of spin-polarized current. More precisely, the domain-wall velocity $v(t)$ is proportional to the current density J Li and Zhang (2004). We have

$$J(t) = \frac{V(t)}{M(\omega) \cdot L \cdot z}, \tag{2.6}$$

and

$$v(t) = \frac{d\omega(t)}{dt} = \frac{\Gamma_v}{L} \cdot J_{eff}(t), J_{eff} = \begin{cases} J, J \geq J_{cr} \\ 0, J \leq J_{cr}. \end{cases} \tag{2.7}$$

Here Γ_v is the domain wall velocity coefficient, which is related to device structure and material property. L and z are the total length and width of the spintronic memristor, respectively. The domain wall movement in the spintronic memristor happens only when the applied current density (J) is above the critical current density (J_{cr}) Li and Zhang (2004); Bazaliy and et al (1998); Zhang and Li (2004); Liu et al. (2005); Tatara and Kohno (2004).

2.1.3 Characteristics of realistic memristive devices

Compared to theoretical characteristics of ideal devices, many non-ideal features have been revealed in real memristive devices. Large device to device variation is one of the issue since memristor is still in its infancy and its compact nano-size structure also aggravate the problem. Since memristor is using the same fabrication process as traditional CMOS technology, the CMOS process variation analysis method will be adopted in our following analysis scheme.

More importantly, although a single memristor can be carefully tuned to arbitrary analog states, this approach cannot be generalized to a large-scale implementation, *e.g.*, a cross-point array with a large number of memristors. We have to face the unfortunate reality that only memristive switches presenting binary states are available in memristive system design Ha and Ramanathan (2011).

Moreover, the stochastic behavior in dynamic switching process and large variations in static states have been widely observed in experimental results of metal oxide materials, such as TiO_2 based memristive switches. In brief, the time to successfully change the state of a single memristive switch is not deterministic but follows a wide lognormal distribution Medeiros-Ribeiro et al. (2011). And its resistance in ON or OFF state (R_{on} or R_{off}, respectively) is not a fixed value, but follows a skewed, heavy tail distribution Yi et al. (2011). These non-ideal characteristics shall be considered in hardware implementations built with memristive switches.

There are many physical memristor models based on insight mechanisms Yang et al. (2009); Miao et al. (2009); Pickett et al. (2009). These models are deterministic so cannot reflect the large variation induced by stochastic switching behavior. Stochastic models can better link the statistical measurement data to probability functions. However, the existing stochastic models are limited to only the binary switching behaviors Medeiros-Ribeiro et al. (2011); Yi et al. (2011). Without considering the stochastic switching of memristor, the intermediate analog state cannot be captured.

2.2 SYNAPSE THEORY

Originally *synapse* is used in biological systems, representing the communication channel of electrical/chemical signals. Later on, the use of synapse was expended to computer science and neuromorphic hardware in developing brain-like computing systems. The use of synapse was then expended to neuromorphic hardware in developing brain-like computing systems. Though the meanings of synapse in these areas share high similarity, there are many subtle differences.

2.2.1 Biological Synapse

The weighting function of biological synapse has been widely taken as a *spatio-temporal* process, that is, a synapse starts participating in the weighting function only after it is

activated by the input spikes from pre-neurons Gollisch and Meister (2008). For instance, the low response variability of retinal ganglion cells shows that the most important info of a firing event by visual neurons is reserved by the timing of the first spike and the number of spikes Keat et al. (2001). Experiments also show that the timing of the first spike after stimulus onset carries the majority of the visual info Gollisch and Meister (2008).

The tuning of synaptic strength has two important properties: *long term potentiation* (LTP) represents a long lasting strength potentiation once a synapse receives strong and positive stimulus from active connections, and *long term depression* (LTD) is an opposite process. The effects of LTP and LTD are strongly related to the spiking time, initial synaptic strength, and type of post-synaptic cells Bi and Poo (1998). More importantly, biological synapses have self-learning abilities, *e.g.*, *spike-timing-dependent plasticity* (STDP) Bi and Poo (1998). Experiment of cultured hippocampal neurons showed that within a critical correlation time window, the post-synaptic spiking that peaked after synaptic activation caused LTP, whereas the post-synaptic spiking before synaptic activation resulted in LTD Bi and Poo (1998).

2.2.2 Artificial Neural Network (ANN) Synapse

ANNs are computational models that mimic nervous systems for machine learning and pattern recognition. ANN synapses are expressed by highly abstracted models of the biological version. Usually here synaptic *strength* is converted to synaptic *weight*, since it could be negative in ANN meanings.

There are different ways to employ ANNs in recall process. For instance, feedforward neural network can be considered as spatial mapping, in which synapses maintain constant weights and provide consistent weighting functions, while a neuron output is determined by its net-input at each time step. In contrast, spiking neural networks can provide temporal mapping: a neuron accumulates the net-input and fire a spike once the accumulation reaches the threshold. One of these procedures is refined as the *leaky-integrate-and-fire* (LIF) model [16]. Synaptic weights are either keep consistent, result in *rate-coded* recall where information is carried by rate of spikes, or updated dynamically depending on the timing of spikes,

lead to *temporal-coded* recall that information is carried by precise timing of spikes For the latter part, STDP learning rule is a well-known example, which is abstracted from the biology observation; The *hierarchical temporal memory* (HTM) model HTM (2011)also employee the temporal feature of synapse in a highly abstracted manner: each synapse has a permanence value that consistently decreases over time, and the permanence value increases only when the synapse receives an input spike that activate it. Synaptic weight changes to 1 if permanence value is beyond threshold, otherwise it is 0. However, spatial and temporal weighting properties of synapse in ANNs are separated since it is usually considered as a single number.

The synaptic weight modification can integrate into the learning process. Among a variety of learning algorithms in ANN family, *Hebbian rule* and *Delta rule* are two fundamental methods. In Hebbian rule, the synaptic weight between two neurons increase (decreases) when they activate simultaneously (separately). Delta rule improves Hebbian rule's learning quality by consider output signal as feedback to minimize the spatial error between target signal and output signal. Note that basic Hebbian and Delta rules are spatial learning algorithms because their weight modifications are determined solely by the signals at the current time step. For spiking neural networks, STDP rule can be regarded as an improved version of Hebbian rule in temporal space as it considers the time correlation between pre-spike and post-spike to modify the synaptic weight. Similar as Delta rule, ReSuMe improves STDP rule's performance by minimizing the temporal difference between target spikes and output spikes. Both STDP and ReSuMe are temporal learning algorithms and they will be detailed in Section II.B.

2.2.3 Electronic Synapse

ANN algorithms have obtained great achievement in many applications. However, the performance of such pure software-level applications is severely limited by the traditional von Neumann architecture, as the "memory wall" problem we mentioned before.

The use of synapse has been expended to neuromorphic hardware in developing brain-like computing systems. Neuromorphic circuits tend to mimic biological models and employ

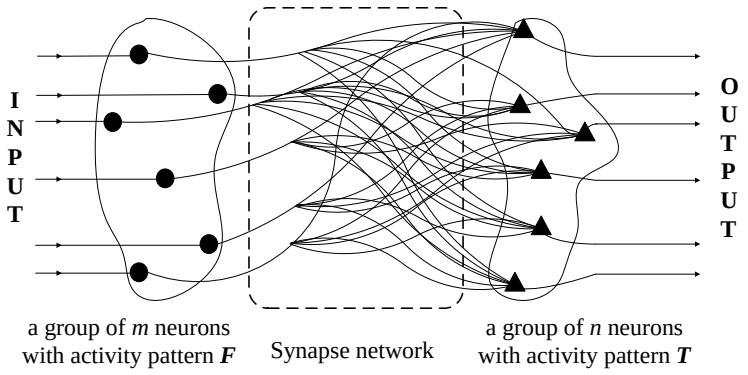

Figure 4: A simple example of neural network.

the same operation flows in weighting, tuning, and learning processes. Previously, SRAM, floating gates, capacitors have been used in developing electronic synapses Mead and Ismail (1989). As aforementioned, these designs are severely constrained by parallelism scale and implementation size Merolla et al. (2011); Seo et al. (2011). Another prominent trend is designing electronic synapses and neurons in analog or mixed-signal format. The process variations and signal fluctuation can dramatically affect the system performance, which however can be partially amortized by synapse's learning ability.

The recently re-discovered *memristor devices* at nanometer scale Jo et al. (2010) demonstrate synapse-alike behaviors, offering a more efficient way to implement electronic synapses. For instance, Snider *et al.* successfully realized LTP, LTD, and STDP in a TiO_2 device controlled by *pulse width modulators* (PWM) Snider (2008). The similar functions were also obtained in Ag/Si memristor synapse by using *time division multiplexing* (TDM) Ambrogio et al. (2013). Very recently, a 1T1R-based HfO_2 synapse was demonstrated with pulse shape filters in pre- and post-neurons Thomas (2013). These demonstrations remain at single device level and require complicated neuron circuits. *To the best knowledge of authors, the spatio-temporal feature has never been presented in electronic synapse designs.*

2.3 NEURAL NETWORK

Figure 4 illustrates a simple example of a neural network, in which two groups of neurons are connected by a set of synapses. We define $a_{i,j}$ as the synaptic strength of the synapse connecting the j^{th} neuron in the input group and the i^{th} neuron in the output one. The relationship of the activity patterns $\mathbf{F}$ of input neurons and $\mathbf{T}$ of output neurons can be described in matrix form:

$$\mathbf{T}_n = \mathbf{A}_{n\times m} \times \mathbf{F}_m, \tag{2.8}$$

where matrix $\mathbf{A}$, denoted as the *connection matrix*, consists of the synaptic strengthes between the two neuron groups. The matrix-vector multiplication of Eq. (2.8) is a frequent operation in neural network theory to model the functionally associated with neurons in brains.

2.3.1 Brain-state-in-a-Box neural network for spatial learning

The BSB model is an auto-associative neural network with two main operations: training and recall Anderson et al. (1977). The mathematical model of BSB recall function can be represented as Wu et al. (2011):

$$\mathbf{x}(t+1) = S(\alpha \cdot \mathbf{A}\mathbf{x}(t) + \beta \times \cdot\mathbf{x}(t)), \tag{2.9}$$

where, $\mathbf{x}$ is an N dimensional real vector, and $\mathbf{A}$ is an N-by-N connection matrix. $\mathbf{A}\mathbf{x}(t)$ is a matrix-vector multiplication, which is the main function of the recall operation. α is a scalar constant feedback factor. β is an inhibition decay constant. $S(y)$ is the "squash" function defined as follows:

$$S(y) = \begin{cases} 1, & \text{if } \; y \geq 1 \\ y, & \text{if } \; -1 < y < 1 \\ -1, & \text{if } \; y \leq -1 \end{cases} . \tag{2.10}$$

For a given input pattern $\mathbf{x}(0)$, the recall function computes Eq. (2.9) iteratively until *convergence*, that is, when all the entries of $\mathbf{x}(t+1)$ are either '1' or '-1'.

The most fundamental BSB training algorithm is given in Figure 5, which bases on the extended Delta rule Anderson et al. (1977). It aims at finding the weights so as to minimize

Algorithm 1. BSB training algorithm using Delta rule.

Step 0. Initialize weights (zero or small random values).
Initialize learning rate α.

Step 1. Randomly select one prototype pattern $\gamma^{(k)} \in B^n$, $k=1,\ldots,m$. B^n is the n-dimension binary space $(-1, 1)$.
Set target output to the external input prototype pattern $\gamma^{(k)}$: $t_i=\gamma_i$.

Step 2. Compute net inputs:

$$y_{in_i} = \sum_j \gamma_j a_{ij}$$

(Each net input is a combination of weighted signals received from all units.)

Step 3. Each unit determines its activation (output signal):

$$y_i = S(y_{in_i}) = \begin{cases} 1, & y_{in_i} \geq 1 \\ y_{in_i}, & -1 < y_{in_i} < 1 \\ -1, & y_{in_i} \leq -1 \end{cases}$$

Step 4. Update weights: $\Delta a_{ij}=\alpha \cdot (t_j - y_j) \cdot \gamma_i$.

Step 5. Repeat *Steps 1-4* until the condition $|t(i) - y(i)|<\theta$ is satisfied in m consecutive iterations.

Figure 5: BSB training algorithm using Delta rule.

the square of the error between a target output pattern and the input prototype pattern. A large α helps improve the training speed but may cause non-convergence. On the other hand, if α is too small, the convergence needs many iterations. The best α usually can be obtained through experimentation. In neuromorphic hardware, the training terminates once its error drops below a predefined threshold, represented by a reference voltage in this work.

A typical application of the BSB model is *optical character recognition*(OCR) for printed text Schultz (1993). A multi-answer character recognition method based on the BSB model has been developed to improve reliability and robustness for noisy or occluded text images Wu et al. (2011). The method first performs a training (pattern memorization) operation on all BSB models such that they "remember" different character patterns. Once trained, input character images can be processed through multiple BSB modules in parallel for the recall (pattern recognition) operation. When all recalls are complete, a set of candidates is selected based on the convergence speed and provided for cogent confabulation-based word or sentence completion Qiu et al. (2013, 2011). This method will be used to evaluate the reliability and robustness of the proposed memristor-based BSB recall circuit.

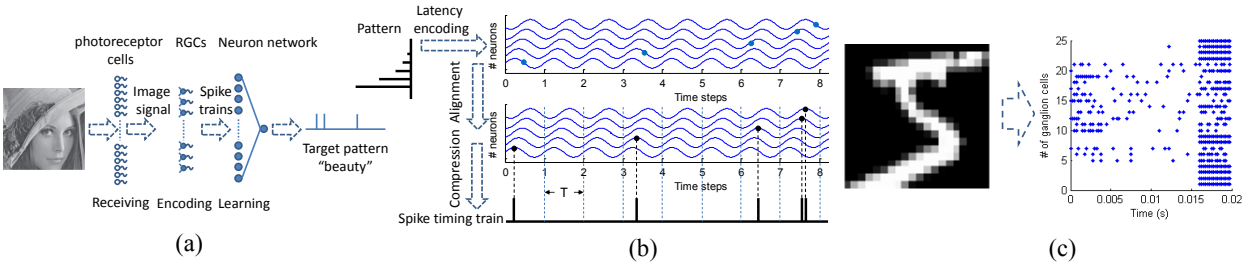

Figure 6: A spike-timing-based model for spatio-temporal pattern recognition. (a) The overview of the spike-timing-based model for vision system; (b) A general LPE flow; (c) An example of spatio-temporal pattern conversion: the image "5" is partitioned and converted to multiple spike-timing trains.

2.3.2 Spiking neural network for temporal learning

Spiking neural networks with temporal codes mimic human vision system for spatio-temporal pattern recognition, demonstrating very high efficiency. Fig. 6(a) illustrates that in human retina, visual signals from 10^8 photoreceptor cells are projected to 10^6 *retinal ganglion cells* (RGCs) in a form of spike trains (temporal codes) through *temporal encoding* Meister and Berry II (1999). These spikes are then processed and learned through *spike-timing-based recall* and *spike-timing-based learning* processes Hu et al. (2013a). We will take such a temporal pattern learning task in Section 7.2 as a case study to examine the performance of MST synapses. The application requires the following three main functions:

Temporal Encoding: We use *latency-phase encoding* (LPE) in Fig. 6(b) to explain how to convert a spatial pattern (*e.g.*, sensory stimuli) into a single spike-timing train Hu et al. (2013a). For a pattern vector, the different locations of variables are encoded into the different phase delays of sinusoid waves. And the amplitude of a variable is represented by the relative timing within the time window of a spike-timing train. Limited by the capacity of spike-timing train, large patterns usually are partitioned and converted into multiple spike-timing trains, as shown in Fig. 6(c). This is so-called *spatio-temporal pattern*.

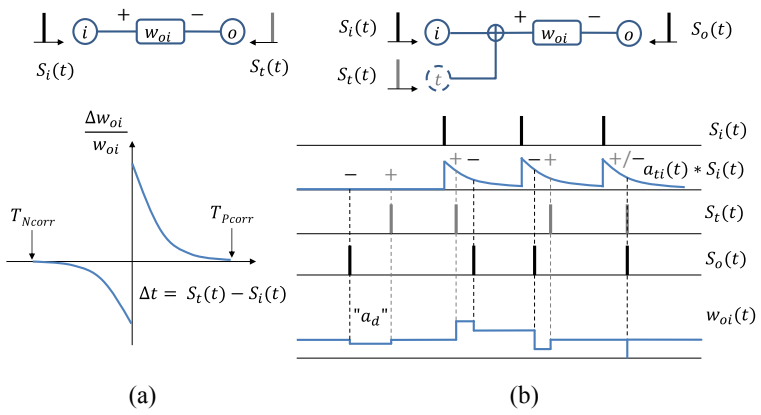

Figure 7: Spike-timing-based learning rules and diagrams of neuron connections. (a) STDP learning rule; (b) ReSuMe learning rule.

Spike-timing-based Recall: Unlike *spike-rate-based recall*, in which the information communication is mainly determined by the rate of input spikes, *spike-timing-based recall* concerns more on their relative timing. In spike-timing-based recall, the relative timing between input and output spikes changes the temporal weight of a synapse (*e.g.*, activated or deactivated) and enables the neural network to produce the desired temporal patterns. The spatial weights of synapses remain unchanged during the entire recall process.

Theoretically, two design approaches can be used to realize the spike-timing function. One attempt is to convert input spike signals into decayed traces to represent the temporal impact. For example, many neuromorphic circuits incorporate an input spike train $S_i(t)$ with a shape modulation term $a_{ti}(t)$ at the input neuron. $S_i(t)$ is then transferred into the convolution form $a_{ti}(t) * S_i(t)$ and passed to the synapse, as illustrated in Fig. 7(b). The synapse performs only spatial weighting function and its weight remains constant in recall. However, complex neuron designs, such as PWM and TDM components, usually are required to provide precise timing controls Snider (2008); Ambrogio et al. (2013). Another approach is adopting temporal synapse which changes synaptic weights over time Ponulak and Kasinski (2010). It can greatly simplify the neuron design. However, synapse's spatial weight is unclear and the synapse design itself becomes very complex.

Spike-timing-based Learning: Most of the spike-timing-based (temporal) learning algorithms can be generalized from the fundamental STDP and ReSuMe learning rules. Let's take a simple structure of one synapse connecting two neurons as an example.

In STDP learning process, the post-neuron is forced to fire the target spike train $S_t(t)$. If an input spike at the pre-neuron $S_i(t)$ occurs within a predefined *positive correlation time window* (T_{Pcorr}) before $S_t(t)$, such as $0 \leq \Delta t = S_t(t) - S_i(t) < T_{Pcorr}$, implying the two neurons have a positive correlation, the synaptic weight in between increases. In contrast, a connection of two negative correlated neurons within a *negative correlation time window* (T_{Ncorr}) shall be weakened. Fig. 7(a) shows a typical dependence of the synaptic weight change rate on Δt.

The learning efficiency of STDP severely relies on the correlation of the input and target spikes. When a target spike is uncorrelated to any of the input spikes, the STDP learning process fails. Thus, more input neurons firing at various patterns are needed to increase the learning capacity of neural networks.

Unlike STDP, the ReSuMe learning rule requires output spikes to participate in the learning process and introduces a cost function to minimize the error between the target and the output spikes. As illustrated in Fig. 7(b), input spikes are first converted to decayed traces and then combined with the target spikes. The produced output spike will be fed back to synapses to assist weight updating. As such, the change of synaptic weight w_{oi} from pre-neuron i to post-neuron o can be described as follows:

$$\frac{d}{dt}w_{oi}(t) = [S_t(t) - S_o(t)][a_t + \int_0^\infty a_{ti}(s)S_i(t-s)ds], \tag{2.11}$$

where a_t is a constant that helps speed up the learning process, and $a_{ti}(t)$ is a shape modulation term which converts $S_i(t)$ to decayed traces. Eq. (2.11) implies that the synaptic weight w_{oi} is updated when $S_t(t) \neq S_o(t)$. The direction and magnitude of the weight tuning are determined by the sign of $S_t(t) - S_o(t)$ and the convolution term $a_{ti}(t) * S_i(t)$, respectively.

3.0 MEMRISTOR MODELS FOR CIRCUIT SIMULATION

3.1 MEMRISTOR VARIATION MODEL

3.1.1 Mathematical analysis

The actual length (L) and width (z) of a memristor is affected by LER. The variation of thickness (h) of a thin film structure is usually described by TF. As a matter of convenience, we define that, the impact of process variations on any given variable can be expressed as a factor $\theta = \frac{\omega'}{\omega}$, where ω is its ideal value, and ω' is the actual value under process variations.

The ideal geometry dimensions of the TiO_2 thin-film memristor and spintronic memristor used in this work are summarized in TABLE 1.

3.1.1.1 TiO_2 thin-film memristor In TiO_2 thin-film memristors, the current passes through the device along the thickness (h) direction. Ideally the doping front has an area $S = L \cdot z$. To simulate the impact of LER on the electrical properties, the memristor device is divided into many small filaments between the two electrodes. Each filament i has a cross-section area ds and a thickness h. Fig. 8 demonstrates a non-ideal 3D structure of a TiO_2 memristor (i.e., with geometry variations in consideration), which is partitioned into many filaments in statistical analysis.

Table 1: The Device Dimensions of Memristors.

	Length(L)	Width(z)	Thickness(h)
Thin-film	50 nm	50 nm	10 nm
Spintronic	200 nm	10 nm	7 nm

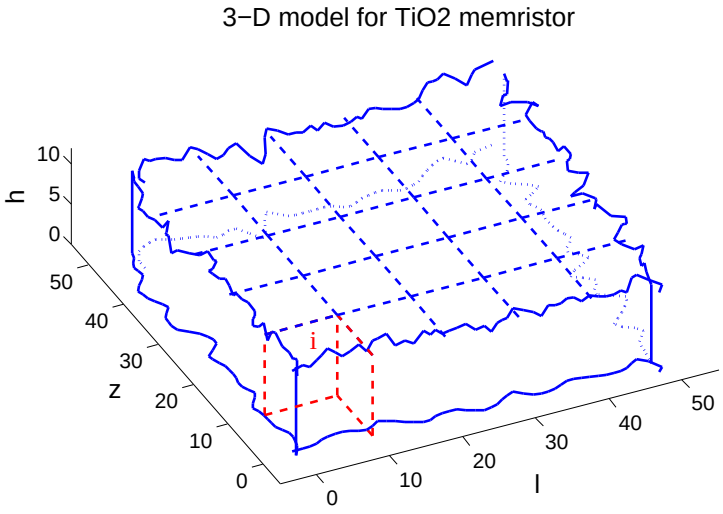

Figure 8: An example of 3D TiO_2 memristor structure, which is partitioned into many filaments in statistical analysis.

As shown in Fig. 8, ideally, the cross-section area of a filament is ds/s of the entire device area and its thickness is h. Thus, for filament i, the ideal upper bound and lower bound of the memristance can be expressed as

$$R_{i,H} = R_H \cdot \frac{S}{ds}, \text{and } R_{i,L} = R_L \cdot \frac{S}{ds}. \tag{3.1}$$

Here, $\theta_{i,s}$ represents the variation ratio on the cross-section area ds, which is caused by 2-D LER. Similarly, $\theta_{i,h}$ is the variation ratio on thickness h due to TF. The resistance of a filament is determined by its section area and thickness, *i.e.*, $R = \rho \cdot \frac{h}{s}$, where ρ is the resistance density. Therefore, the actual upper and the lower bound under the process variations can be expressed as

$$R'_{i,H} = R_{i,H} \cdot \frac{\theta_{i,h}}{\theta_{i,s}}, \text{and } R'_{i,L} = R_{i,L} \cdot \frac{\theta_{i,h}}{\theta_{i,s}}. \tag{3.2}$$

If a filament is small enough, we can assume it has a flat doping front. Then, the actual doping front velocity in filament i considering process variations can be calculated by replacing the ideal values with actual values in Eq.(2.4). We have

$$v'_i(t) = \mu_v \cdot \frac{R'_{i,L}}{h'^2} \cdot \frac{V(t)}{M'_i(\omega'_i)}. \tag{3.3}$$

Here h' and M'_i are the actual thickness and memristance of filament i. Then, we can get a set of related equations for filament i, including the doping front position

$$\omega'_i(t) = \int_0^t v'(\tau) \cdot d\tau, \tag{3.4}$$

the corresponding memristance

$$M'_i(\omega'_i) = \omega'_i \cdot R'_{i,L} + (1 - \omega'_i) \cdot R'_{i,H}, \tag{3.5}$$

and the current through the filament i

$$I'_i(t) = \frac{V(t)}{M'_i(\omega'_i)}. \tag{3.6}$$

By combining Eq. (3.3) – (3.6), the doping front position in every filament i under process variations $a'_i(t)$ can be obtained by solving the differential equation

$$\frac{d\omega'_i(t)}{dt} = \mu_v \cdot \frac{R'_{i,L}}{h'^2} \cdot \frac{V(t)}{\omega'_i(t) \cdot R'_{i,L} + (1 - \omega'_i(t)) \cdot R'_{i,H}}. \tag{3.7}$$

Eq. (3.7) indicates that the behavior of the doping front movement is dependent on the specific electrical excitations, e.g., $V(t)$.

For instance, applying a sinusoidal voltage source to the TiO_2 thin-film memristor such as

$$V(t) = V_m \cdot \sin(2\pi f \cdot t), \tag{3.8}$$

the corresponding doping front position of filament i can be expressed as:

$$\omega'_i(t) = \frac{R_{i,H} - \sqrt{R_{i,H}^2 - A \cdot B(t) \cdot \frac{2}{\theta_{i,h}^2} + 2C[1] \cdot A \cdot \frac{\theta_{i,s}}{\theta_{i,h}}}}{A}. \tag{3.9}$$

Where, $A = R_{i,H} - R_{i,L}$, $B(t) = \mu_v \cdot R_{i,L} \cdot V_m \cdot \cos(2\pi f \cdot t)$, and $C[1]$ is an initial state constant.

The term $B(t)$ accounts for the effect of electrical excitation on doping front position. The terms $\theta_{i,s}$ and $\theta_{i,h}$ represent the effect of both LER and TF on memristive behavior. Moreover, the impact of the geometry variations on the electrical properties of memristors could be affected by the electrical excitations. For example, we can set $\omega(0) = 0$ to represent

the case that the TiO_2 memristor starts from $M(0) = R_H$. In such a condition, $C[1]$ becomes 0, and hence, the doping front position $\omega_i'(t)$ can be calculated as:

$$\omega_i'(t) = \frac{R_{i,H} - \sqrt{R_{i,H}^2 - A \cdot B(t) \cdot \frac{2}{\theta_{i,h}^2}}}{A}, \tag{3.10}$$

which is affected only by TF and electrical excitations. LER will not disturb $\omega_i'(t)$ if the TiO_2 thin-film memristor has an initial state $\omega(0) = 0$.

The overall memristance of the memristor can be calculated as the total resistance of all n filaments connected in parallel. When n goes to ∞, we can have

$$R_H' = \frac{1}{\int_0^\infty 1/R_{i,H}' \cdot di} = R_H \cdot \frac{1}{\int_0^\infty \theta_{i,h}/\theta_{i,s} \cdot di}, \tag{3.11}$$

and

$$R_L' = \frac{1}{\int_0^\infty 1/R_{i,L}' \cdot di} = R_L \cdot \frac{1}{\int_0^\infty \theta_{i,h}/\theta_{i,s} \cdot di}. \tag{3.12}$$

The overall current through the memristor is the sum of the current through each filament:

$$I'(t) = \int_0^\infty I_i'(t) \cdot di. \tag{3.13}$$

The instantaneous memristance of the overall memristor can be defined as

$$M'(t) = \frac{V(t)}{I'(t)} = \frac{1}{\int_0^\infty 1/M_i' \cdot di}. \tag{3.14}$$

Since the doping front position movement in each filament will not be the same because h_i' varies due to TF (and/or the roughness of the electrode contact), we define the average doping front position of the whole memristor as:

$$\omega'(t) = \frac{R_H' - M'(t)}{R_H' - R_L'}. \tag{3.15}$$

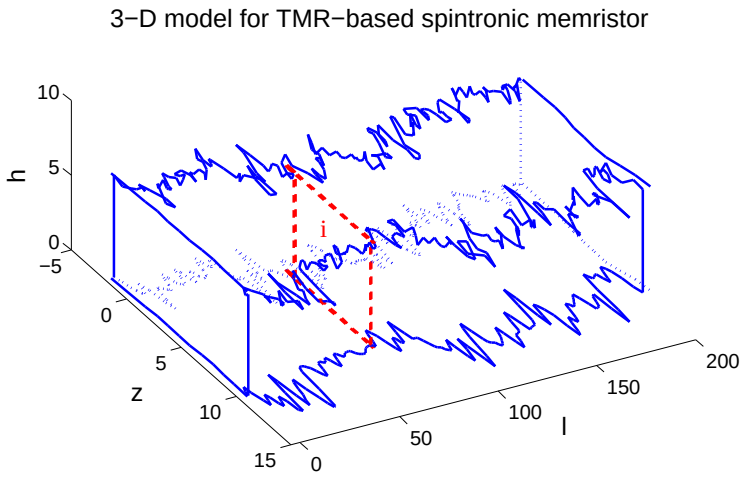

Figure 9: An example of 3D TMR-based spintronic memristor structure, which is partitioned into many filaments in statistical analysis.

3.1.1.2 Spintronic memristor Since the length of a spintronic memristor is usually much longer than the other two dimensions, the impact of the variance in length on the spintronic memristor's electrical properties can be ignored. In our analysis, the device can be chopped into infinite segments along the length direction as shown in Fig. 9. For a segment i, the upper and lower bounds of memristance are:

$$R'_{i,H} = R_{i,H} \cdot \frac{\theta_{i,h}}{\theta_{i,z}}, \text{and } R'_{i,L} = R_{i,L} \cdot \frac{\theta_{i,h}}{\theta_{i,z}}. \tag{3.16}$$

Here we assume the ideal memristance changes linearly within the domain wall, or M_i changes linearly from $R_{j,L}$ to $R_{k,H}$ when $j < i < k$. Here j and k are the two segments at the two boundaries of domain wall and connected to the magnetic domains with either the low or the high resistance states. The memristance of each segment is

$$M'_i = \begin{cases} R'_{i,L}, i < \omega' \\ R'_{i,H}, i \geq \omega' \end{cases} \tag{3.17}$$

So for overall resistance R'_H and R'_L, we have

$$R'_H = \frac{1}{\int_0^\infty 1/R'_{i,H} \cdot di} = R_H \cdot \frac{1}{\int_0^\infty \theta_{i,z}/\theta_{i,h} \cdot di}, \tag{3.18}$$

and

$$R'_L = \frac{1}{\int_0^\infty 1/R'_{i,L} \cdot di} = R_L \cdot \frac{1}{\int_0^\infty \theta_{i,z}/\theta_{i,h} \cdot di}. \tag{3.19}$$

Then the memristance of the whole device is

$$\begin{aligned} M'(\omega') &= \frac{1}{\displaystyle\int_0^{\omega'} \frac{1}{R'_{i,L}} di + \int_{\omega'}^1 \frac{1}{R'_{i,H}} di} \\ &= \frac{1}{\displaystyle\int_0^{\omega'} \frac{1}{R_{i,L}} \cdot \frac{\theta_{i,z}}{\theta_{i,h}} di + \int_{\omega'}^1 \frac{1}{R_{i,H}} \cdot \frac{\theta_{i,z}}{\theta_{i,h}} di} \end{aligned} \tag{3.20}$$

Here the width of each segments z_i varies segment by segment due to the LER effect. The statistical behavior of spintronic memristors can still be evaluated by Monte-Carlo simulation in Section 3.1.3.

We assume the current density applied on the domain wall $J'(t)$ is the one of the segments i where the domain wall located in the middle:

$$J'(t) = J'_i = \frac{V(t)}{M'(\omega') \cdot L \cdot z'_i}. \tag{3.21}$$

Then the domain wall velocity under process variations can be calculated as:

$$\begin{aligned} v'(t) = v'_i &= \frac{d\omega'(t)}{dt} = \frac{\Gamma_v}{L} \cdot J'_{eff}(t), \\ J'_{eff} &= \begin{cases} J', J' \geq J_{cr} \\ 0\ , J' < J_{cr} \end{cases} \end{aligned} \tag{3.22}$$

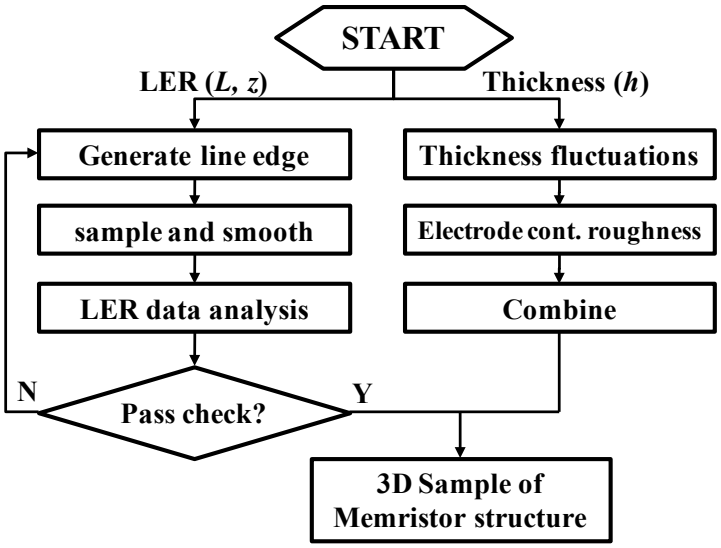

Figure 10: The flow of 3D memristor structure generation including geometry variations.

3.1.2 3D memristor structure modeling

Analytic modeling is a fast way to estimation the impact of process variations on memristors. However, we noticed that in modeling some variations analytically, e.g. simulating the LER may be beyond the capability of analytic model Jiang and et.al (2009). The data on silicon variations, however, is usually very hard to obtain simply due to intellectual property protection. To improve the accuracy of our evaluations, we create a simulation flow to generate 3-D memristor samples with the geometry variations including LER and thickness fluctuation. The correlation between the generated samples and the real silicon data are guaranteed by the sanity check of the LER characterization parameters. The flow is shown in Figure. 10.

Many factors affecting LER show different random effects. Usually statistical parameters such as the auto-correlation function (ACF) and power spectral density (PSD) are used to describe the property of LER. ACF is a basic statistical function of the wavelength of the line profile, representing the correlation of point fluctuations on the line edge at different position. PSD describes the waveform in the frequency domain, reflecting the ratio of signals

with different frequencies to the whole signal. Considering that LER issues are related to fabrication processes, we mainly target the nano-scale pattern fabricated by electron beam lithography (EBL). The measurements show that under such a condition, the line edge profile has two important properties: (1) the line edge profile in ACF figure demonstrates regular oscillations, which are caused by periodic composition in the EBL fabrication system; and (2) the line edge roughness mainly concentrates in a low frequency zone, which is reflected by PSD figure Jiang and et.al (2009).

To generate line edge samples close to the real cases, we can equally divide the entire line edge into many segments, say, n segments. Without losing the LER properties in EBL process, we modified the random LER modeling proposed in Ban et al. (2009) to a simpler form with less parameters. The LER of the i^{th} segment can be modeled by

$$\text{LER}_i = L_{LF} \cdot \sin(f_{max} \cdot x_i) + L_{HF} \cdot p_i. \tag{3.23}$$

The first term on the right side of Eq. (3.23) represents the regular disturbance at the low frequency range, which is modeled as a sinusoid function with amplitude L_{LF}. f_{max} the mean of the low frequency range derived from PSD analysis. Without loss of generality, a uniform distribution $x_i \in U(-1, 1)$ is used to represent an equal distribution of all frequency components in the low frequency range. The high frequency disturbances are also taken into account by the second term on the right side of Eq. (3.23) as a Gaussian white noise with amplitude L_{HF}. Here p_i follows the normal distribution $N(0, 1)$ Jiang and et.al (2009). The actual values of L_{LF}, L_{HF} and f_{max} are determined by ACF and PSD.

To ensure the correlation between the generated line edge samples with the measurement results, we introduce four constraints to conduct a sanity check of the generated samples:

- σ_{LER}: the root mean square (RMS) of LER;
- σ_{LWR}: the RMS of line width roughness (LWR);
- Sk: skewness, used to specify the symmetry of the amplitude of the line edge; and
- Ku: kurtosis, used to describe the steepness of the amplitude distribution curve.

Table 2: The Parameters/Constraints In LER Characterization.

Parameters		Constraints	
L_{LF}	0.8 nm	σ_{LER}	2.5nm $\sim$ 3.5nm
f_{max}	1.8 MHz	σ_{LWR}	4.0nm $\sim$ 5.0nm
L_{HF}	0.4 nm	Sk	0.1nm $\sim$ 0.2nm
/	/	Ku	2.5nm $\sim$ 3.5nm

The above four parameters are widely used in LER characterization and can be obtained from measurement results directly Jiang and et.al (2009). Only the line edge samples that satisfy the constraints will be taken as valid device samples. TABLE 2 summarizes the parameters used in our algorithm, which are correlated with the characterization method and experimental results in Jiang and et.al (2009). And Fig. 11 shows the LER characteristic parameters distribution among 1000 Monte-Carlo simulations.

Even the main function has captured the major features of LER, it is not enough to mimic all the LER characteristics. The difference between LER data and simulation results in the fact that some generated samples are not qualified compared to the characteristic parameters, or the constraints of the real LER profile. Thus, sanity check which screens out the unsuccessful results is necessary. Only those samples in red rectangles shown in Fig. 11 satisfy the constraints and will be used for the device electrical property analysis. The criteria of the sanity check are defined based on the measurement results of LER data.

The thickness fluctuation is caused by the random uncertainties in sputter deposition or atomic layer deposition. It has a relatively smaller impact than the LER and can be modeled as a Gaussian distribution. Since the memristors in this work have relatively bigger dimensions in the horizontal plane than the thickness direction (shown in TABLE 1), we also considered roughness of electrode contact in our simulation: The means of the thickness of each memristor is generated by assuming it follows the Gaussian distribution. Each memristor is then divided into many filaments between the two electrodes. The roughness of electrode contracts is modeled based on the variations of the thickness of each filament. Here, we assume that both thickness fluctuations and electrode contact roughness follow Gaussian distributions with a deviation $\sigma = 2\%$ of thin film thickness.

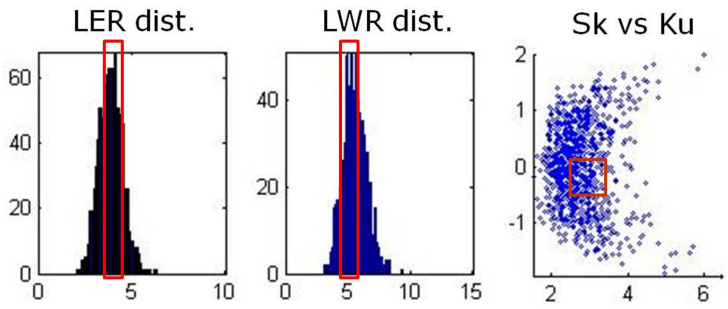

Figure 11: LER characteristic parameters distribution among 1000 Monte-Carlo simulations. Constraints are shown in red rectangles.

Fig. 8 is an example of 3D structure of a TiO_2 thin-film memristor generated by the proposed flow. It illustrates the effects of all the geometry variations on a TiO_2 memristor device structure. According to Section 3.1.1, a 2-D partition is required for the statistical analysis. In the given example, we partition the device into 25 small filaments with the ideal dimensions of $L = 10nm$, $z = 10nm$, and $h = 10nm$. Each filament can be regarded as a small memristor, which is affected by either only TF or both LER and TF. The overall performance of device can be approximated by paralleled connecting all the filaments.

Similarly, Fig. 9 is an example of 3D structure of a TMR-based spintronic memristor. Since the length of a spintronic memristor is much longer than its width and height, only 1-D partition along the length direction is required. In this case, the device is divided into 200 filaments. Ideally, each filament has $L = 1nm$, $z = 10nm$, and $h = 7nm$. Each filament i is either in the low resistance state $R'_{i,L}$ or the high resistance state $R'_{i,H}$, with considering the effects of both LER and TF. The overall performance of device can be approximated by paralleled connecting all the filaments.

3.1.3 Experimental results

3.1.3.1 Simulation setup To evaluate the impact of process variations on the electrical properties of memristors, we conducted Monte-Carlo simulations with 10,000 qualified 3-

Table 3: Memristor Devices and Electrical Parameters

TiO_2 thin-film memristor Strukov et al. (2008a)					
$R_L(\Omega)$	$R_H(\Omega)$	$\mu_v(\mathrm{m}^2 \cdot \mathrm{s}^{-1} \cdot \mathrm{V}^{-1})$	/	V_m (V)	f (Hz)
100	16000	10^{-14}	/	1	0.5
Spintronic memristor Wang and Chen (2010)					
$R_L(\Omega)$	$R_H(\Omega)$	$\Gamma_v(\mathrm{nm}^3 \cdot \mathrm{C}^{-1})$	J_{cr} (A / nm^2)	V_m (V)	f (Hz)
2500	7500	2.01×10^{-14}	2.00×10^{-8}	2	10M

D device samples generated by our proposed flow. A sinusoidal voltage source shown in Eq. (3.8) is applied as the external excitation. The initial state of the memristor is set as $M(\omega = 0) = R_H$. The device and electrical parameters used in our simulations are summarized in TABLE 3. Both separate and combined effects of geometry variations on various properties of memristors are analyzed, including:

- the distribution of R_H and R_L;
- the change of memristance $M(t)$ and $M(\omega)$;
- the velocity of wall movement $v(\omega)$;
- the current through memristor $i(t)$; and
- the I-V characteristics.

3.1.3.2 TiO_2 thin-film memristor The $\pm 3\sigma$ (minimal/maximal) values of the device/electrical parameters as the percentage of the corresponding ideal values are summarized in TABLE 4. For those parameters that vary over time, we consider the variation at each time step of all the devices. The simulation results considering only either LER or TF are also listed. To visually demonstrate the overall impact of process variations on the memristive behavior of TiO_2 memristors, the dynamic responses of 100 Monte Carlo simulations are shown in Fig. 12.

TABLE 4 shows that the static behavior parameters of memristors, i.e., R_H and R_L, are affected in a similar way by both LER and thickness fluctuations. This is consistent to our analytical results in Eq. (3.11) and (3.12), which show that θ_s and θ_h have the similar effects on the variation of R'_H and R'_L.

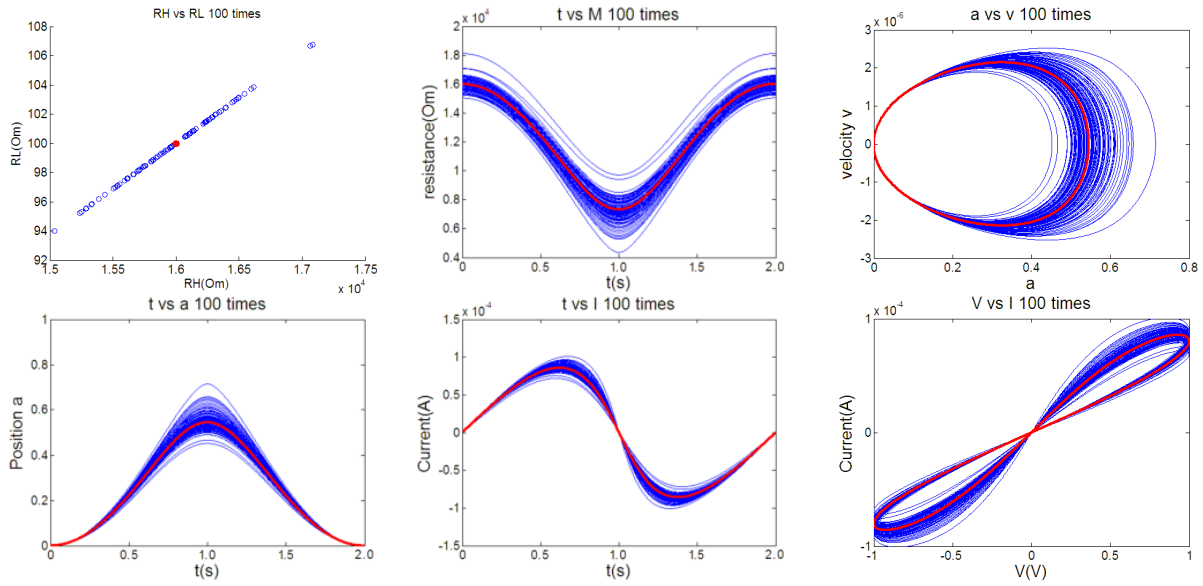

Figure 12: Simulation results for TiO_2 thin-film memristors. The blue curves are from 100 Monte-Carlo simulations, and red lines are the ideal condition. From top left to right bottom, the figures are R_H vs. R_L; $M(t)$ vs. t; v vs. ω; ω vs. t; I vs. t; and $I-V$ characteristics.

However, thickness fluctuation shows a much more significant impact on the memristive behaviors such as $v(t)$, $\omega(t)$ and $M(\omega)$, than LER does. It is because the doping front movement is along the thickness direction: $v(t)$ is inversely proportional to the square of the thickness, and $\omega(t)$ is the integral of $v(t)$ over time as shown in Eq. (3.3) and (3.4). For the same reason, thickness fluctuations significantly affect the instantaneous memristance $M(\omega)$ as well.

Because the thickness of the TiO_2 memristor is relative small compared to other dimensions, we assume the doping front cross-section area is a constant along the thickness direction in our simulation. The impact of LER on $\omega(t)$ or $v(t)$, which is relatively small compared to that of the thickness fluctuations, is ignored in TABLE 4.

An interesting observation in Fig. 12 is that as the doping front ω moves toward 1 (fully doped), the velocity v regularly grows larger and reaches its peak at the half period of the sinusoidal excitation, i.e. t=1s. This can be explained by Eq. (3.5): the memristance is getting smaller as ω moves toward 1 (fully doped). With the same input amplitude, a

smaller resistance will result in a bigger current and therefore a bigger variation on $v(t)$. Similarly, memristance $M(\omega)$ reaches its peak variance when ω is close to 1 (fully doped).

We also conduct 10,000× Monte Carlo simulations on the same samples by applying a square wave voltage excitation. The amplitude of the voltage excitation is ±0.5V. The simulation results are also shown in TABLE 4. The results of the static behavior parameters, i.e., R_H and R_L, are exactly the same as those with sinusoidal voltage inputs because they are independent of the external excitations, The results of the memristive behavior parameters such as $v(t)$, $\omega(t)$ and $M(\omega)$ show similar trends as those with the sinusoidal voltage inputs. Based on Eq. (3.9), $\omega(t)$'s variance is sensitive to the type and amplitude of electrical excitation, because $B(t)$ greatly affects the weight of the thickness fluctuation parameter. That is why the thickness fluctuation has a significantly impact on the electrical properties of memristors under sinusoidal and square voltage excitations.

3.1.3.3 Spintronic memristor The $\pm 3\sigma$ values of the device/electrical parameters based on 10,000 Monte-Carlo simulations are summarized in TABLE 5. The visual demonstration of 100 Monte-Carlo simulations is shown in Fig. 13.

For the spintronic memristor, the impact of LER on the electrical properties of memristors is more than that of thickness fluctuation. This is because the direction of the domain wall movement is perpendicular to the direction of spin-polarized current. The impact of thickness fluctuations on very small segments cancel each other during the integral along the direction of the domain wall movement.

"LER only" simulation results show that the $+3\sigma$ corner of LER has more impact on the electrical properties than that of -3σ corner. This is because the line width variation is the dominant factor on the variation of electrical properties of spintronic memristors, and the line edge profiles used in our LER parameters have a right-biased feature Jiang and et.al (2009). Since normal distribution is assumed for the variations of thickness, σ_h has approximately symmetric impact on $\pm 3\sigma$ corners.

The impact of LER on the memristive parameters $v(t)$, $\omega(t)$ and $M(\omega)$ is also larger than thickness variation. Again, the impact of thickness fluctuations on very small segments cancel each other during the integral along the direction of the domain wall movement.

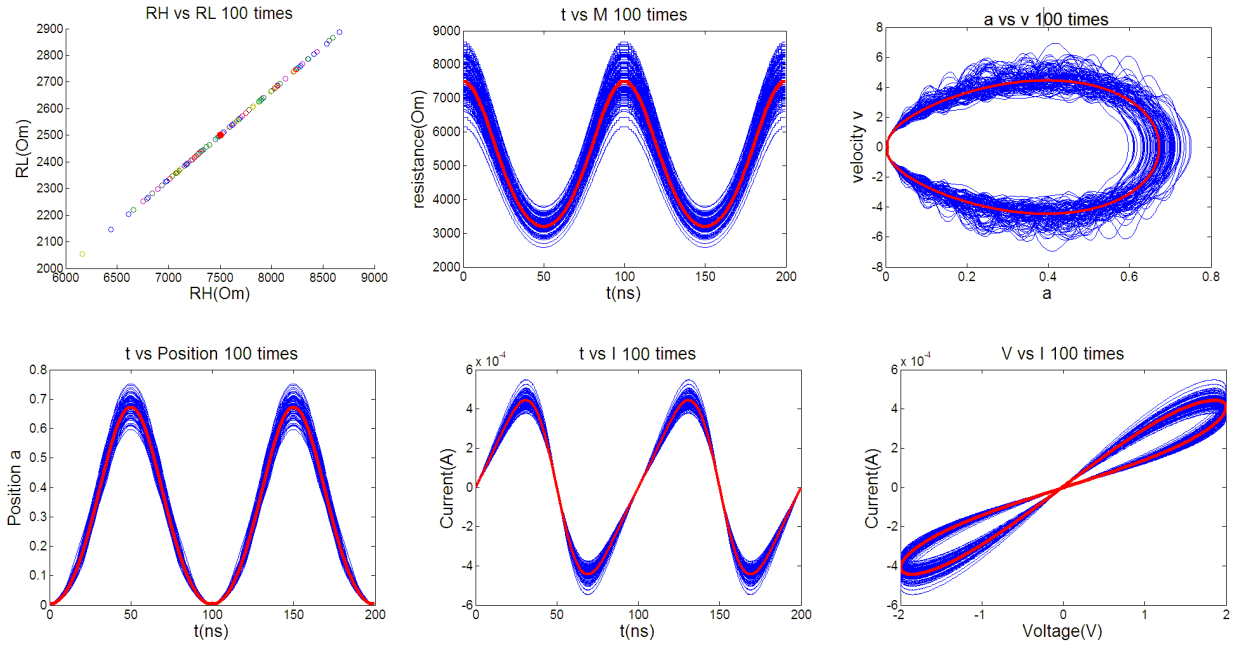

Figure 13: Simulation results for spintronic memristors. The blue curves are from 100 Monte-Carlo simulations, and red lines are the ideal condition. From top left to right bottom, the figures are R_H vs. R_L; $M(t)$ vs. t; v vs. ω; ω vs. t; I vs. t; and $I-V$ characteristics.

Table 4: 3σ min./max. of TiO_2 memristor parameters

Sinusoidal Voltage	LER only		TF only		overall	
	-3σ(%)	$+3\sigma$(%)	-3σ(%)	$+3\sigma$(%)	-3σ(%)	$+3\sigma$(%)
R_H&R_L	-5.4	4.1	-5.5	4.8	-6.4	7.3
$M(\omega)$	-5.4	4.1	-37.1	20.8	-36.5	24.1
$\omega(t)$	0.0	0.0	-13.3	27.5	-14.7	27.4
$v(\omega)$	0.0	0.0	-9.3	15.6	-10.4	16.9
$i(\omega)$	-4.7	5.7	-9.3	15.7	-10.7	17.2
Power	-4.7	5.7	-8.8	14.1	-10.1	15.6

Square wave Voltage	LER only		TF only		overall	
	-3σ(%)	$+3\sigma$(%)	-3σ(%)	$+3\sigma$(%)	-3σ(%)	$+3\sigma$(%)
R_H&R_L	-5.3	3.7	-6.2	5.2	-6.6	6.9
$M(\omega)$	-5.3	3.7	-17.8	13.2	-15.4	14.4
$\omega(t)$	0.0	0.0	-12.1	16.6	-13.0	15.6
$v(\omega)$	0.0	0.0	-11.6	17.7	-12.5	16.7
$i(\omega)$	-4.0	5.2	-11.7	17.7	-12.6	17.6
Power	-4.0	5.2	-7.7	9.8	-8.5	10.1

Similarly, we also conduct Monte Carlo simulations by applying a square wave voltage excitation. The amplitude of the voltage excitation is $\pm$1V. The similar trends as that of sinusoidal excitations are observed.

3.1.4 Similarities and differences

TiO_2 thin-film memristors and TMR-based spintronic memristors have many similarities. For example, the device dimensions are both in nanometer-scale; the overall memristances can be expressed by simple equations with only three parameters R_H, R_L, and ω; the memristive functions are obtained by changing the position of doping front or magnetic domain.

However, according to the different device structures and physical mechanisms, there are some significant distinctions between the characteristics of these two types of memristors. Although the memristive behaviors of these two memristors both come from the movement of the boundary between two segments with different resistance states, the relationship between the total memristance of the memristor M and the boundary position ω are quite different: In a TiO_2-based memristor, $M(\omega)$ is a linear function of ω. In a spintronic memristor, M is inversely proportional to ω because the direction of the domain wall movement is perpendicular to the direction of spin-polarized current. This fundamental difference introduces very different electrical responses of these memristors even under the same process variations.

Table 5: 3 σ min./max. of spintronic memristor parameters

Sinusoidal Voltage	LER only		TF only		overall	
	$-3\sigma(\%)$	$+3\sigma(\%)$	$-3\sigma(\%)$	$+3\sigma(\%)$	$-3\sigma(\%)$	$+3\sigma(\%)$
R_H&R_L	-15.3	22.9	-6.1	5.8	-16.4	20.9
$M(\omega)$	-15.1	23.3	-11.0	11.0	-16.3	21.1
$\omega(t)$	-9.7	8.1	-8.4	9.5	-11.8	8.1
$v(\omega)$	-10.7	22.1	-9.1	9.9	-21.5	22.5
$i(\omega)$	-18.5	18.5	-8.9	10.1	-17.7	17.8
Power	-18.4	18.6	-8.3	9.4	-17.8	17.8
Square wave Voltage	LER only		TF only		overall	
	$-3\sigma(\%)$	$+3\sigma(\%)$	$-3\sigma(\%)$	$+3\sigma(\%)$	$-3\sigma(\%)$	$+3\sigma(\%)$
R_H&R_L	-15.8	22.0	-5.3	5.7	-15.9	24.2
$M(\omega)$	-15.6	21.8	-8.5	9.7	-17.0	25.5
$\omega(t)$	-13.1	13.8	-7.5	7.7	-17.2	16.2
$v(\omega)$	-16.5	20.7	-10.0	8.3	-20.1	25.2
$i(\omega)$	-19.5	17.1	-9.0	9.3	-22.1	20.5
Power	-19.4	17.1	-7.6	7.7	-20.9	19.6

For a TiO_2 memristor, thickness fluctuation is the primary variation source that affects the device electrical properties, while LER demonstrates a more important effect in a TMR-based spintronic memristor. On one hand, the static parameters of TiO_2-based memristors, i.e., R_H and R_L, show less sensitivity to the process variations compared to spintronic memristors. On the other hand, the memristive behavior parameters of TiO_2-based memristors, i.e., $v(t)$, $\omega(t)$ and $M(\omega)$, are more affected from the process variations than spintronic memristors. These similarities and differences will be important references when different types of memristors are chosen for various applications and manufacturing environments.

3.1.5 Conclusion

In this work, we evaluate the impact of geometry variations on the electrical properties of TiO_2-based memristors and spintronic memristors, by conducting analytic modeling analyses and Monte-Carlo simulations. We investigate the different responses of the static and memristive parameters of memristors under various process variations and analyze their implication for the electrical properties of memristors. A simple LER sample generation algorithm is also proposed to speed up the related Monte-Carlo simulations. To author's best knowledge, this is the first work that conducts the quantitative evaluation and comparison on the impact of geometry variations to these two types of memristors.

3.2 STATISTICAL MEMRISTOR MODEL

The filament-based geometric variation model is powerful for device variation analysis, however, it is very time-consuming in large scale Monte-Carlo simulations. An fast statistical memristor variation model is preferred to speed up the simulation within acceptable error.

3.2.1 Statistical analysis of memristor model

When the device variations are within the reasonable range, we can assume the ratio between the actual device parameter and its designed value as a polynomial expression, or $x' = \eta \cdot x$. Here η is a coefficient representing the effects of process variations. Our goal is to find an efficient methodology to compute the variation-aware coefficient η. The total memristance M', which is a time-varying parameter, can be uniquely defined by ${R'}_H$, ${R'}_L$, and $\omega'(t)$. In this section, we will examine the variations of ${R'}_H$, ${R'}_L$, and $\omega'(t)$ first, and then derive the corresponding process-variation aware memristor model.

3.2.1.1 Distribution of ${R'}_H$ and ${R'}_L$ For a given TiO_2 memristor, we use ${R'}_H$ and ${R'}_L$ to denote its actual highest and lowest total memristances, respectively. With considering the impact of Random Doping Defect (RDD), Eq. 3.2 could expand to:

$$ {R'}_{H,i} = \int_0^{{h'}_i} \frac{\rho_{off}}{{s'}_i(\omega'_i)} \cdot \omega'_i, \text{ and } {R'}_{L,i} = \int_0^{{h'}_i} \frac{\rho_{on}}{{s'}_i(\omega'_i)} \cdot \omega'_i \tag{3.24} $$

As Eq. 3.24 shows, the geometry variations(${h'}_i$ and ${s'}_i$) influence both ${R'}_H$ and ${R'}_L$ simultaneously within the filament i. It indicates ${R'}_H$ and ${R'}_L$ are correlated. However, they are not fully correlated because of the randomness in ρ_{on} and ρ_{off} incurred by RDD. We define $\gamma = \rho_{on}/\rho_{off}$, which can be modeled by a normal distribution as $\gamma = \mu_\gamma \cdot (1 + \sigma_\gamma \cdot D)$. Here $\mu_\gamma = R_H/R_L$ and $D \sim N(0, 1)$. The actual value of σ_γ will be determined by the particular device structure, material and fabrication process. In our following simulations, we assume $\sigma_\gamma = 2\%$.

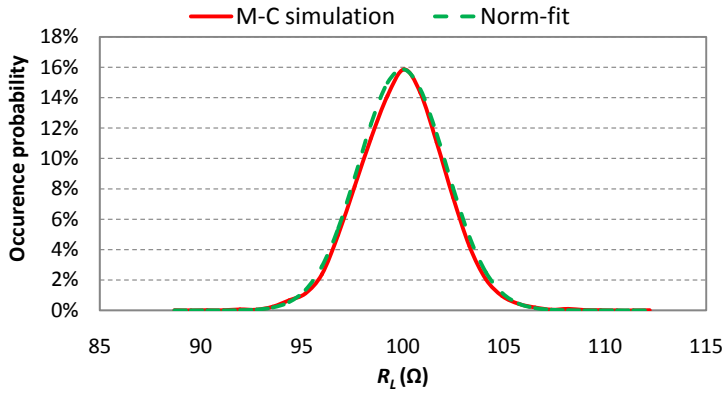

Figure 14: The Distribution of R'_L from 10,000 Monte-Carlo simulations and the fitting curve of Eq. (3.25).

To obtain the distributions of R'_H and R'_L, we conducted Monte-Carlo simulations with 10,000 3D device samples. Our results show that the distributions of both R'_H and R'_L are close to normal distributions and can be approximated by

$$R'_L = R_L \cdot (\mu_{R_L} + \sigma_{R_L} \cdot E), \tag{3.25}$$

and

$$R'_H = R_H \cdot (\mu_{R_H} + \sigma_{R_H} \cdot E) \cdot (1 + \sigma_\gamma \cdot D). \tag{3.26}$$

Here, two independent random numbers $E \sim \mathcal{N}(0,1)$ and $D \sim \mathcal{N}(0,1)$ are introduced. E represents the correlation between R'_H and R'_L due to the same geometry variation sources. D represents the impact of RDD, which affects the ratio between ρ_{off} and ρ_{on}.

Figure 14 compares the approximated normal distribution shown in Eq. (3.25) and the actual distribution of R'_L from Monte-Carlo simulations. The mean square root error incurred by the normal distribution approximation is only 4.4%.

3.2.1.2 Distribution of ω' As shown in Eq. 3.7, the calculation of the average doping front ω' requires time-consuming filament-based simulations. If we can somehow extract the actual ω' directly from the designed value ω with the consideration of process variations, the simulation cost can be improved.

Considering the fact that $R_H \gg R_L$ in a TiO_2 memristor, a simple approximation of Eq. 3.9 can be:

$$\omega(t) = 1 - \sqrt{1 - X}, \tag{3.27}$$

where,

$$X = \frac{2\mu_v}{\gamma \cdot h^2} \cdot (\int_{t_0}^{t} V(t) \cdot dt + \omega_0 - \frac{1}{2} \cdot \omega_0^2). \tag{3.28}$$

Eq. 3.28 shows that the variation of ω can be directly linked to the variation of X, which has three independent contributions: (1) the variation of thin-film thickness h, (2) the impact of RDD that is represented by γ, and (3) the magnetic flux of the input signal $\varphi = \int V dt$. Interestingly, LER does not impact ω', as also proved by the simulations in Hu et al. (2011a).

Impact of flux φ and boundary condition.

If there are no process variations, the average doping front ω will be uniquely determined by the magnetic flux φ, as shown in Eq. 3.28. However, after taking into account the process variations, the historical profile of the electrical excitations (instead of only the absolute value of φ) will introduce the additional variations of φ by interacting with the device process variations such as thin film thickness h' and RDD γ',

To understand how the historical profiles of φ, e.g., the amplitude and the time duration etc., affect the variation of ω, we conducted the Monte-Carlo simulations with 10,000 3D device samples and trace the position of the doping front $\omega'(t)$ for every device. A sinusoid input signal $V = V_{ap} \cdot \sin(2\pi f \cdot t)$ with a fixed frequency $f = 0.5Hz$ is applied. The $+3\sigma$ and -3σ variations of $\omega(t = 1s)$ when varying V_{ap} from $0.1V$ to $1.2V$ are shown by the curves labeled with "3D, +3sigma" and "3D, -3sigma" in Figure 15, respectively. Here the results show that the deviation of the actual value $\omega'(t = 1s)$ from the designed value $\omega(t = 1s)$, normalized against $\omega(t = 1s)$. We note that the absolute value of $\varphi = V_{ap}/(\pi f)$, which is proportional to V_{ap}.

The simulation results show that the 3σ variance of ω' is approximately proportional to φ, except at the boundary when ω' is close to 1 (or the total memristance is close to R'_L): a big overshoot of the $+3\sigma$ corner at $V_{ap} = 1.1V$ is observed. This is because X is proportional to φ and ω changes rapidly when X is approaching 1, as shown in Eq. 3.27–3.28.

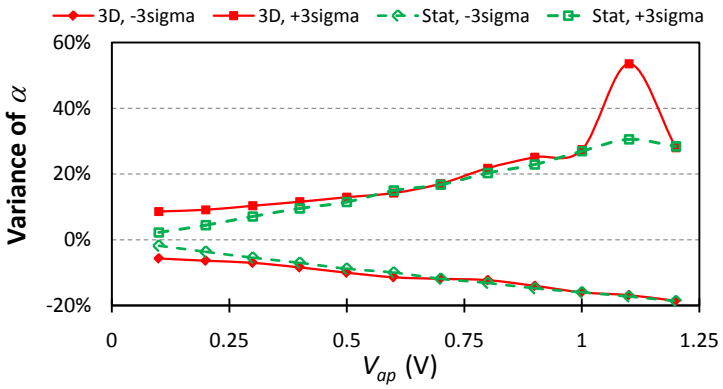

Figure 15: $\pm 3\sigma$ variances of ω vs. V_{ap} at $t = 1s$.

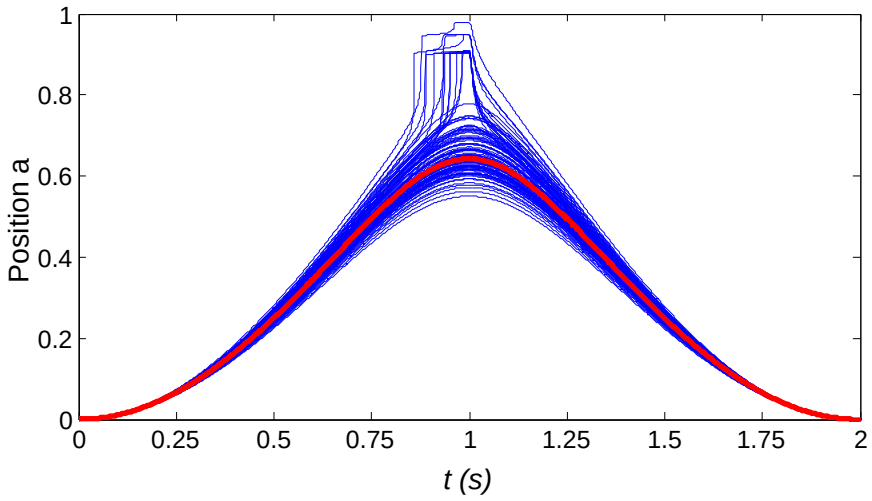

Figure 16: $\omega'(t)$ when $V_{ap} = 1.1V$ based on 100 Monte-Carlo simulations.

A close look at this boundary situation is shown in Figure 16, which records the movement history of the doping fronts of 100 device samples when $V_{ap} = 1.1V$. The RED curve is the theoretical result while the 100 BLUE curves come from Monte-Carlo simulation. The average doping fronts of some devices with large process variations hit the device boundary during the movements and cause the large variance of ω' at the $+3\sigma$ corner. The devices at the -3σ corner do not enter the non-linear region in Eq. 3.27 yet, and correspondingly, "3D, -3sigma" in Figure 15 do not have abnormal disturbance.

Further increasing the flux φ, i.e., increasing V_{ap} from $1.1V$ to $1.2V$ makes the variance of ω' drop, as also shown in Figure 15. Under this scenario, the amplitude of φ is so large that the average doping front of most of the simulated devices have reached the device boundary. These devices with a constant $\omega' = 1$, however, will not contribute to the statistics of the variances of average doping fronts. A new concept named "effective flux", which includes the constraints of device boundary, can be expressed as

$$\varphi_{eff} = \begin{cases} \int_{t_0}^{t} V_{ap} \cdot dt & (\omega' < 1) \\ h^2(R_H + R_L)/(2\mu_v R_L) & (\omega' \geq 1). \end{cases} \tag{3.29}$$

Impact of process variations.

The complexity of Eq. 3.27 and 3.28 make it infeasible to find a simple analytical expression of the variance of the ω' even if we assume both h' and γ' follow normal distributions. However, we are still able to construct a polynomial-based ω' model by taking the means and the variances of h' and γ' as the variables.

By running extensive numerical simulations under various conditions, we found the actual ω' can be modeled as the product of the designed value ω and a coefficient η that describes the influence of process variations as:

$$\omega' = \eta \cdot \omega, \tag{3.30}$$

where, η can be expressed by an heuristic formula (partially from the first order Taylor expansion of Eq. 3.27 and 3.28 as

$$\eta = \frac{1}{(1 + \varphi \cdot \varepsilon_1 + \varphi \cdot \varepsilon_2 \cdot (w_1 \cdot E + w_2 \cdot G)) \cdot (1 + \sigma_\gamma \cdot D)}. \tag{3.31}$$

Table 6: Statistical model of TiO_2 thin-film memristor.

Variation Parameters: μ_{R_H}, σ_{R_H}, μ_{R_L}, σ_{R_L}, σ_γ
Coefficients: w_1, w_2, ε_1, ε_2
Independent Random Numbers: $D \sim \mathcal{N}(0,1)$, $E \sim \mathcal{N}(0,1)$, $G \sim \mathcal{N}(0,1)$.
Memristance boundary:
$R'_H = R_H \cdot (\mu_{R_H} + \sigma_{R_H} \cdot E) \cdot (1 + \sigma_\gamma \cdot D)$
$R'_L = R_L \cdot (\mu_{R_L} + \sigma_{R_L}) \cdot E$
Input flux:
$$\varphi_{eff} = \begin{cases} \int_{t_0}^{t} V_{ap} \cdot dt & (\omega' < 1) \\ h^2(R_H + R_L)/(2\mu_v R_L) & (\omega' \geq 1) \end{cases}$$
Doping front position:
$\omega' = \eta \cdot \omega$
$\eta = \frac{1}{(1+\varphi\cdot\varepsilon_1+\varphi\cdot\varepsilon_2\cdot(w_1\cdot E+w_2\cdot G))\cdot(1+\sigma_\gamma\cdot D)}$
Memristance:
$M'(\omega) = R'_L \cdot \omega' + R'_H \cdot (1 - \omega')$

Similar to the definitions we used in Section 3.2.1.1, D and E are two independent random numbers that represent the impact of RDD and geometry variations, respectively. To avoid overestimating the impacts of geometry variations on the ω', a new random number $G \sim \mathcal{N}(0,1)$ is introduced to remove the impact of LER. ε_1 and ε_2 are two scalars extracted from the actual simulations. The coefficients w_1 and w_2 represent the weights of E and G, where $w_1^2 + w_2^2 = 1$.

3.2.1.3 Distribution of M'

By modeling R'_H, R'_L and ω' with normal distribution approximations and heuristic polynomial approximations, the total memristance M' of a TiO_2 memristor can be simply calculated by $M'(\omega) = R'_L \cdot \omega' + R'_H \cdot (1 - \omega')$. Table 6 summarizes the main steps and equations included in our proposed statistical model of the TiO_2 memristor.

3.2.2 Model generation flow

In this section, we describe the parameter extraction methodology for the generation of the statistical model of TiO_2 memristors. Some critical implementation considerations are also discussed.

Based on the discussion in Section 3.2.1, we propose a four-step extraction methodology to obtain the corresponding statistical model parameters.

Step 1: Model input characterization.

In the electrical testing of a memristor device, only three parameters can be measured directly: R'_H, R'_L and $I(t)$. The measurement of $I(t)$, which is a time-vary variable, requires two doping front ω movements following both $0 \rightarrow 1$ and $1 \rightarrow 0$ directions under certain electrical excitations: as we discussed in Section 3.2.1.2, the electrical excitations must be carefully controlled to avoid the case where the doping front hits the device boundary. The absolute value of the flux φ can be simply calculated as the integral of the applied voltage over time. The actual data can come from either the electrical testing or the Monte-Carlo simulations with detailed device modeling.

Step 2: Distribution generations for R'_H and R'_L.

The PDF of R'_L can be modeled as a normal distribution as shown in Eq. 3.25. The mean μ_{R_L} and variance σ_{R_L} can be easily extracted from the statistical data generated in Step 1. Though R'_H in Eq. 3.26 does not strictly follow a normal distribution, a mean of μ_{R_H} can still be extracted. By being exposed to the same geometry variations, $\sigma_{R_H} \approx \sigma_{R_L}$ in Eq. 3.26. As we shall show in Section 3.2.3, modeling R'_H as a normal distribution approximation introduces very marginal discrepancy and has limited impact on the accuracy of ω modeling. Finally, the variance of σ_γ needs to be provided by material-level characterizations.

Step 3: Distribution generation for ω'.

For every device sample, the resistance of the memristor and doping front location at a given time stamp t_j can be simply calculated as:

$$M'(t_j) = \frac{V(t_j)}{I(t_j)},\ \omega'(t_j) = \frac{R'_H - M'(t_j)}{R'_H - R'_L}. \tag{3.32}$$

Based on the calculated samples, the distribution of $M'(t_j)$ and $\omega'(t_j)$ can be generated with the mean μ_ω and the variance σ_ω of $\omega'(t_j)$. Then two scalars – ε_1 and ε_2 in from Eq. (3.30) and (3.31) can be extracted.

We proposed a heuristic method to generate w_1 and w_2: We start with an assumption that $w_1 = 1$ and $w_2 = 0$. The corresponding $\mu_{\hat{M}}$ and $\sigma_{\hat{M}}$ are calculated from the distribution of M'. At this moment, we have $\sigma_{\hat{M}} > \sigma_M$ and $\mu_{\hat{M}} \neq \mu_M$ by overestimating the impact of geometry variations. The mismatch between measurement data and the simulated parameters, i.e., $\mu_{\hat{M}}$ and $\sigma_{\hat{M}}$ could be large. Continuously decreasing w_1 can reduce the

mismatch between the distributions of the simulated data and the measurement data. Then the optimal approximations of w_1 and w_2 can be obtained under the best fitting condition.

Step 4: Model verification and improvement.

After constructing the statistical models by extracting all necessary parameters, the static and dynamic behaviors of TiO_2 memristors under various electrical excitations can be simply simulated without conducting the time-consuming Monte-Carlo simulations. However, we can still run Monte-Carlo simulations by taking into account the device samples to verify the accuracy of our proposed models.

The accuracy of our proposed model may be improved by optimizing the termination conditions of the parameter optimization iterations. The detailed algorithm written in pseudocode is shown as below:

Algorithm 1: Statistical Model for Computer Simulation

$E = \mathcal{N}(0,1)$, $G = \mathcal{N}(0,1)$, $D = \mathcal{N}(0,1)$ generated at the beginning of simulation;

$\varphi(n+1) = \varphi(n) + V(n+1) \cdot dt$
if $abs(\varphi(n+1)) > \varphi_{eff}$
  $\varphi(n+1) = sign(\varphi(n+1)) \cdot \varphi_{eff}$
end if
$\eta(n+1) = \frac{1}{(1+\varphi(n+1)\cdot\varepsilon_1+\varphi(n+1)\cdot\varepsilon_2\cdot(w_1\cdot E+w_2\cdot G))\cdot(1+\sigma_\gamma\cdot D)}$
$vel(n+1) = \mu_v \cdot \frac{R_L}{h^2} \cdot \frac{V(n+1)}{M(n)}$
$\omega(n+1) = a(n) + vel(n+1) \cdot dt$
$\omega'(n+1) = \eta(n+1) \cdot \omega(n+1)$
Check boundary condition of $\omega'(n+1)$
If $\omega'(n+1)$ reaches boundary
$\varphi(n+1) = 0$
end if
$M'(n+1) = R'_L \cdot \omega'(n+1) + R'_H \cdot (1 - \omega'(n+1))$
return M

3.2.3 Simulation results

Because of the lack of published data of memristor device variations, we use Monte-Carlo simulation results as the baseline to validate our proposed statistical models. Three parameters that are used to generate our statistical models – R'_H, R'_L and $I(t)$, are simulated by using the 3D device structure examples. However, there is nothing prevent us from using electrical testing data of memristor devices as the input of our model generation. The parameters of the generated statistical model are summarized in Table 7.

In this section, we verify the effectiveness of our proposed statistical model by comparing the results of various electrical properties with that of Monte-Carlo simulations based on 3D device structure samples. A total of 10,000 Monte-Carlo simulations are running.

Test 1: Fixed input signal. A sinusoid input signal $V = V_{ap} \cdot \sin(2\pi f \cdot t)$ with $V_{ap} = 1V$ and $f = 0.5Hz$ is applied to the simulated memristor device. Figure 17, Figure 18, and Figure 19 show the simulation results of $\omega'(t)$, $M'(t)$, and $I - V$ characteristics, respectively. The results of our statistical model (labeled "stat") approximate the results of the Monte-Carlo simulations with the 3D device samples (labeled "3D") very well: On average, our statistical model shows only $\sim 2\%$ difference from the one of the 3D Monte-Carlo simulations, for the memristances at the $\pm 3\sigma$ corners. Table 8 summarizes the variances of each electrical property at each corner.

Test 2: Frequency Dependency. The same sinusoid input signal is applied with $V_{ap} = 2V$ while changing f from $2Hz$ to $10Hz$. Again, our statistical model demonstrates a good approximation in the whole range, as shown in Figure 20.

Test 3: Impact of Flux. In this test, the input signal is still a sinusoid signal with a fixed $f = 0.5Hz$. We vary V_{ap} from 0.1V to 1.2V and measure ω and M at $t = 1s$. For an ideal device without any process variations, ω reaches to 1 when $V_{ap} = 1.26V$. Hence, there is no need to raise V_{ap} beyond it. The comparisons of ω and M vs. various V_{ap} are shown in Figure 15 and Figure 21, respectively. Our results show that the memristance M results from our statistical model fits well with that of the Monte-Carlo simulation results in most of the working region. A large discrepancy is observed on ω curves only when V_{ap} is high $\sim 1.1V$.

The difference between the results of 3D Monte-Carlo simulation and our statistical

Table 7: Statistical model parameters

Variation Parameters		Coefficients	
μ_{R_H}	0.994	ε_1	-0.028
μ_{R_L}	0.994	ε_2	0.072
σ_{R_H}	2.16%	w_1	0.98
σ_{R_L}	2.16%	w_2	0.2
σ_γ	2%		

Table 8: Variance between 3D model and statistical model.

	$a - t$	$M - t$	$V - I$
$+3\sigma$	4.97%	2.03%	2.20%
-3σ	5.05%	2.12%	1.99%

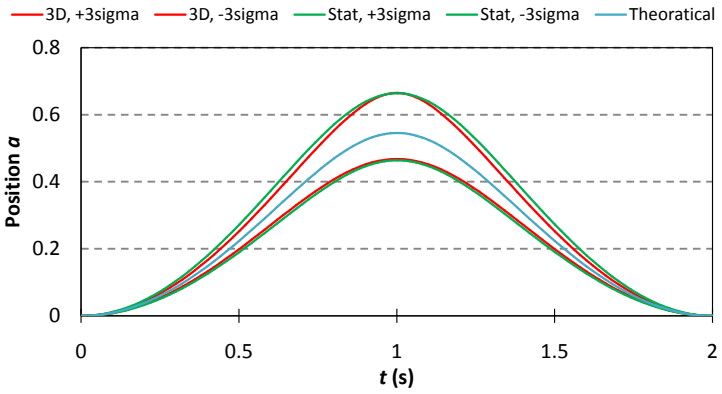

Figure 17: Comparison of $\omega'(t)$ at $\pm 3\sigma$ corners.

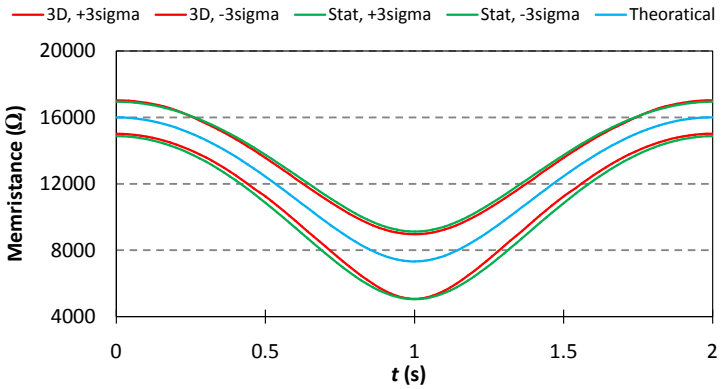

Figure 18: Comparison of $M'(t)$ at $\pm 3\sigma$ corners.

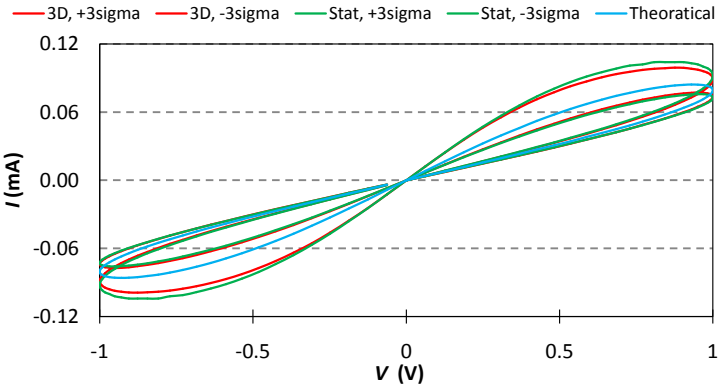

Figure 19: Comparison of $I - V$ at $\pm 3\sigma$ corners.

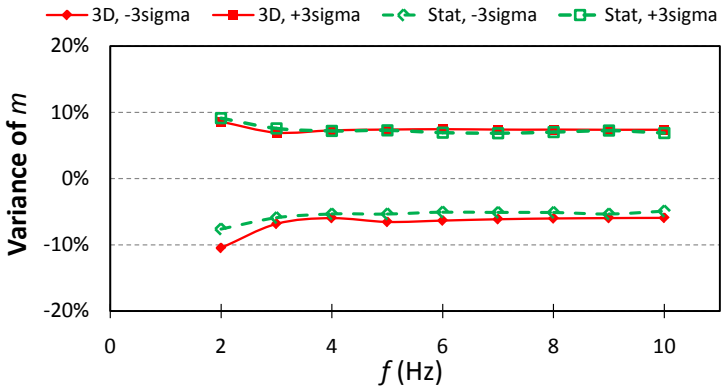

Figure 20: $\pm 3\sigma$ variances of M vs. f at $t = \frac{1}{2f}$.

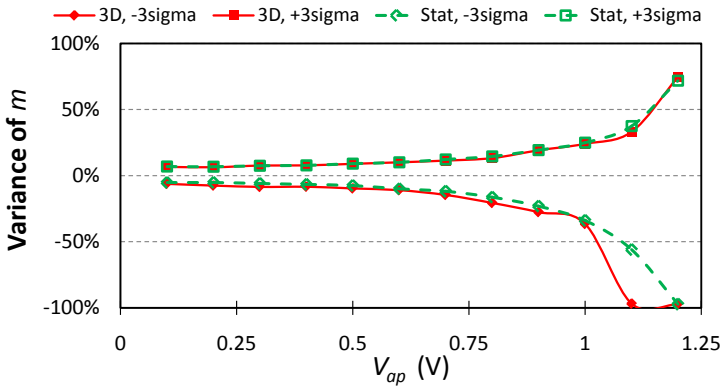

Figure 21: $\pm 3\sigma$ variances of M vs. V_{ap} at $t = 1s$.

model simulations at $V_{ap} = 1.1V$ is due to the nonlinear relationship between φ and ω when ω is close to 1. After memristance enters into the low resistance region, continuing to increase the flux may not necessarily raise the variance of $\omega = 1$ because the doping fronts will hit the device boundary. The two simulations then merge into each other. In real design, we recommend the designer not operate the memristors in the low resistance region when utilizing its analog state.

Runtime Comparison.

The proposed statistical model of the TiO_2 memristor can significantly improve runtime cost. For instance, 10,000 device-based Monte-Carlo simulations Hu et al. (2011a) took ~ 2 hours in a MATLAB environment, while the same number of simulations using the proposed statistical model only spent ~ 2 seconds.

3.3 STOCHASTIC MEMRISTOR MODEL

To better describe the stochastic memristive switching in both static states and dynamic switching process, we proposed a stochastic model for TiO_2 memristive switch based on both the inspection of the physical mechanisms Yang et al. (2009); Pickett et al. (2009) and the statistical analysis of experimental data Yu et al. (2011); Gaba et al. (2013).

3.3.1 On and OFF Static States

The static stochastic behavior can be described by the distributions of R_L and R_H. In TiO_2 memristor, the initial barrier width w follows a normal distribution and the device resistance exponentially depends on w. Therefore the distribution of state resistance follows the lognormal *probability density function* (pdf) function, which is Yang et al. (2009):

$$f_x(x;\mu,\sigma) = \exp\left(-\frac{(\ln x/\mu)^2}{2\sigma^2}\right) / \left(x\sigma\sqrt{2\pi}\right), x > 0. \tag{3.33}$$

Here, μ is the normal mean and σ is the standard deviation. Note that R_L or R_H does not change within a given static state for each switching cycle since we use post-probability

function to predict the static state after each dynamic switching behavior. Therefore, we can use lognormal function (Lognorm) to generate the sampling data, such as

$$R_L = \text{Lognorm}(\mu_{R_L}, \sigma_{R_L}), \text{and} \quad R_H = \text{Lognorm}(\mu_{R_H}, \sigma_{R_H}). \tag{3.34}$$

3.3.2 Dynamic Switching Process

The dynamics in TiO_2 memristor is a complex oxide electroforming process. It can be explained as an electro-reduction and vacancy creation process caused by high electric fields and enhanced by electrical Joule heating. Usually the barrier width w is used to model the vacancy channeling mechanism. Although the vacancy channeling mechanism has been evidenced by experiments Yang et al. (2009), it is difficult to match it to a pure physical model. Instead, our model is based on the analysis of three major behaviors; we start with a mathematical analysis of the analog stochastic switching behavior from the statistical aspect, and then bridge the parameters in mathematical expression with the physical excitation. At last, the impact of over tune is integrated into the stochastic model.

Analog Stochastic Switching Behavior: The stochastic resistance changing has been observed in high frequency measurement at low voltage Pickett et al. (2009). The time dependency of switching probability can be approximated by the *cumulative probability function* (CDF) of lognormal distribution, such as Medeiros-Ribeiro et al. (2011):

$$P(\text{Success switch}) = \frac{1}{2}\text{erfc}\left[-\frac{(\ln t_{switch}/\mu_t)^2}{\sqrt{2}\sigma_t^2}\right]. \tag{3.35}$$

Here, t_{switch} represents the pulse width of activation time. And μ_t and σ_t are related to the external voltage V.

Instead of studying the complicated physical mechanism and its impact, we use mathematical method to analyze the ON-OFF switching probability. According to Eq. 3.35, the ON-OFF switching probability can be approximated by a CDF of lognormal distribution, differentiation of $P(\text{Success switch})$ at t_{switch}, then is a pdf of the lognormal distribution, such as

$$\frac{dP(\text{Success switch})}{dt_{switch}} = f_{t_{switch}}(t_{switch}; \mu_t, \sigma_t). \tag{3.36}$$

Eq. (3.36) describes the distribution of the increment of switching probability $dP(\text{Success switch})$ at time t_{switch} when applying a signal with a short pulse width dt_{switch}.

The switching mechanism of a memristive device is intrinsic. Hence, the characteristic of the stochastic behavior remains unchanged and follows the same probability function during its switching process. From its physical meaning perspective, Eq. (3.36) reflects the increment of switching probability at time t_{switch}, which can be associated to the resistance change ΔR. Physically, a successful switching event with a pulse of t_{switch} indicates that the device resistance changes from R_L to R_H, or vice versa, that is, $\Delta R = |R_H - R_L|$.

Considering that both ON and OFF switching are the cumulative results of the analog resistance changing and the increment of switching probability is directly reflected by the change of resistance, the change of analog resistance at time t_{switch} can be generated by mapping to the distribution of the increment of switching probability, leading to

$$\frac{dR}{dt} = (R_H - R_L) \cdot f_{t_{switch}}(t_{switch}; \mu_t, \sigma_t). \tag{3.37}$$

Time & Voltage Dependency of Switching Probability describes the switching probability of memristive switch under applied voltage V and activation time t_{switch}. The switching process resulted from the cumulative impact of input signals can be modeled with CDF function. The lognormal switching time distribution comes from the nonlinear switching dynamics of the devices. Considering that the median switch time (μ_t) is exponentially dependent on the applied voltage amplitude V, we approximate μ_t as an exponential function, such as:

$$\mu_t = \exp(aV + b), \tag{3.38}$$

where a and b are fitting parameters.

Since σ_t has only a weak dependence on V, we can approximate the relationship between σ_t and V by a hard threshold squashing function, such as

$$\sigma_t = \begin{cases} \sigma_{thres\_H} & (\sigma_t \geq \sigma_{thres\_H}) \\ c \cdot V + d & (\sigma_{thres\_L} < \sigma_t < \sigma_{thres\_H}) \\ \sigma_{thres\_L} & (\sigma_t \leq \sigma_{thres\_L}) \end{cases}. \tag{3.39}$$

Where, c and d are fitting parameters. σ_{thres_H} and σ_{thres_L} are the upper and lower boundaries, respectively. Our model applies two individual sets of fitting parameters to ON and OFF switching processes.

The Resistance Shifting Due To Over Tune: Over tune stands for the behavior when one or more external voltage pulses continue being applied in the switching direction after the state switching of memristor already succeeds. For example, apply an ON switching signal to a device already in ON state. Based on the vacancy channeling mechanism, the over tune in OFF state continues eliminating the oxygen vacancy until all the oxygen vacancies disappear and the device becomes an insulator. In ON state, the over tune creates more oxygen vacancies to form more conducting channels. The device mechanism becomes less appropriate to be modeled with barrier width w since the channel frontier no longer exists. The resistance shifting in real devices is even more complex after including thermal, electron kinetic energy, and other physical issues. During over tune, a memristor device remains in the same static state and the resistance shifting follows the static resistance distribution. However, a systematic impact on μ_{R_L} and μ_{R_H} has been observed Yi et al. (2011).

Here, we use a statistical method to analyze the impact of over tune on the resistance shifting. The charge q flowing through the device is used as the input variable, which has a direct impact on the number of oxygen vacancies and the device resistance. To exhibit the trend of resistance shifting, a linear approximation can be assumed between the passing charge q and the mean shifting μ_{shift} as Strukov et al. (2008a):

$$\mu_{shift} = e \cdot q = e \cdot \frac{V}{M} \cdot t. \tag{3.40}$$

Here, e is the fitting parameter that describes the shift speed of mean, M is the current memristor resistance. The new μ_{R_L} and μ_{R_H} can be calculated from Eq. (3.39):

$$\mu'_{R_L} = \mu_{R_L} - \mu_{on-shift} = \mu_{R_L} - e_{on} \cdot q, \ \mu'_{R_L} \geq 0. \tag{3.41a}$$

$$\mu'_{R_H} = \mu_{R_H} + \mu_{off-shift} = \mu_{R_H} + e_{off} \cdot q. \tag{3.41b}$$

Though more complicated fitting equations can be established, such an approach is impractical and unnecessary at current stage considering of insufficient experimental data available.

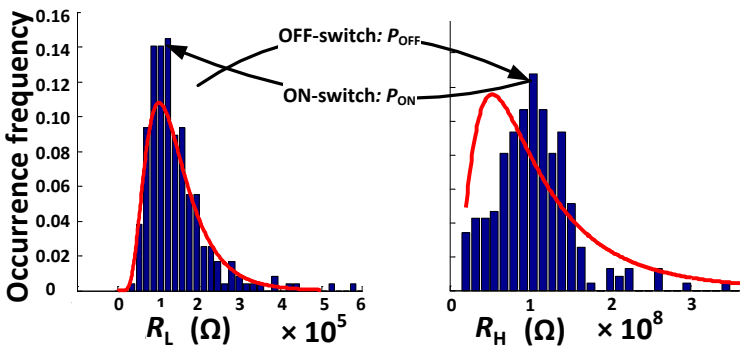

Figure 22: The static state distribution of a TiO_2 memristor device.

The resistance shifting caused by over tune is constrained within the target resistance state, demonstrating less impact on the overall memristor characteristic compared to the ON-OFF switching.

3.3.3 Stochastic Model Verification

We verified the proposed stochastic model from perspectives of static states and dynamic switching process.

Static States: Fig. 22 shows the resistance distributions of a memristive switch in ON and OFF states. The blue bars in the figure are real measurement data of a TiO_2 memristor device Yi et al. (2011). The results show that the lognormal distribution fits well to the real device data in ON state. However, in OFF state, the heavy tail is captured but the median value is slightly skewed. Though the distribution of R_H is not perfectly fitted, the error in distribution fitting of R_H has ignorable impact in the circuit simulation since R_H is more than two orders of magnitude higher than R_L.

Dynamic Switching Process: Fig. 23 shows the time dependencies of ON and OFF switching probability at different applied voltages. The results have high approximation to the experimental results Medeiros-Ribeiro et al. (2011). The error mainly comes from the approximation of the relationship between σ_t and V. As aforementioned, establishing a more reliable estimation of σ_t requires more experimental data. Fig. 24 shows the simulated analog resistance changing process of a TiO_2 memristor to better demonstrate the time and

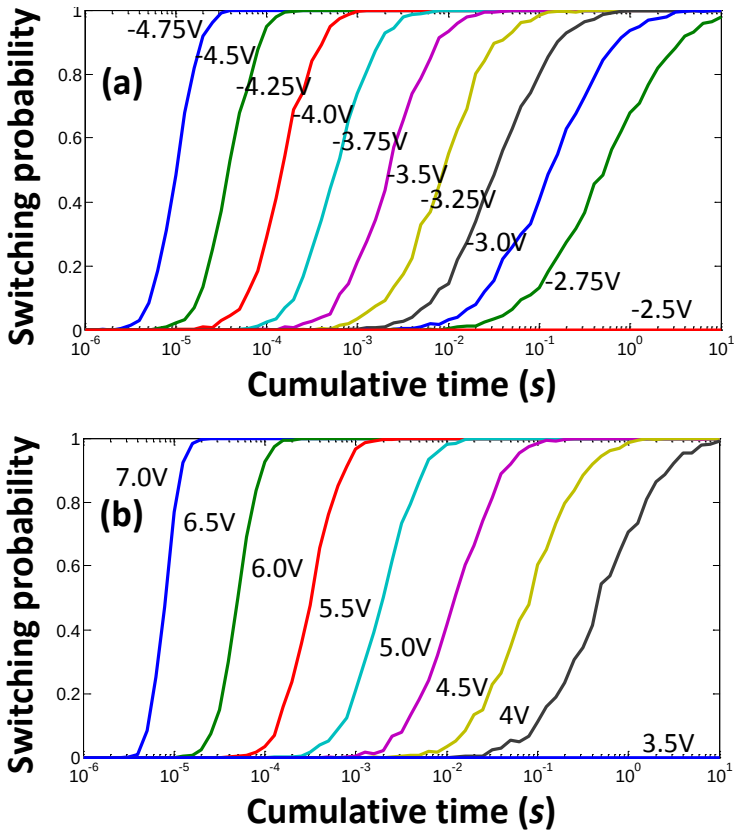

Figure 23: The time dependency of ON (a) and OFF (b) switching at different external voltage V.

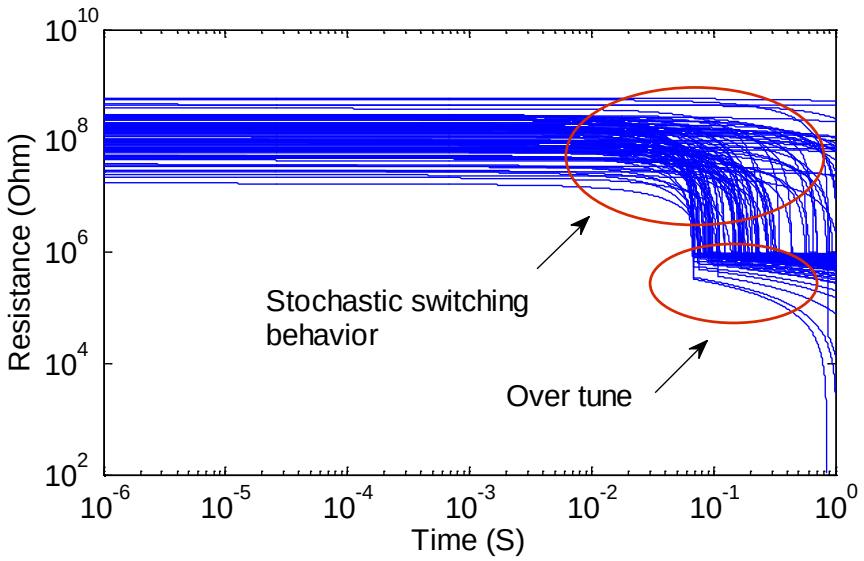

Figure 24: The analog switching process of a TiO_2 memristor.

voltage dependency of switching probability and the resistance shifting due to over tune. The external voltage is set as $3.0V$ to switch the memristor from R_H to R_L. The 100 curves in the figure represent the resistance changings by repeating 100 times of the ON switching procedure for the same device. The distribution of 100 tests agrees well with the switching probability curve at $-3.0V$ in Figure 3(a): about 40% of the curves reach R_L before $0.1S$.

Considering the obvious stochastic behavior of memristive device at nanometer regime, traditional device modeling based on curve fitting is not enough. In this work, we built a stochastic model for TiO_2 memristor by bridging the key physical mechanisms and the experimental data fitting. The model combines the stochastic characteristics in static states and dynamic switching process together, and extends the stochastic study to the analog state while still holding high approximation to the existing data. The accurate and fast estimation on the distribution of device's analog states makes the proposed model more meaningful for higher level circuit and system designs. This model can be generalized to other metal oxide memristors Yu et al. (2011); Gaba et al. (2013) for the same stochastic nature, that is, the percolation property of the thin dielectric soft breakdown. The proposed model can be further enhanced by integrating with reliable physical model that precisely describes

the stochastic switching mechanism. The complex and slow physical model generates the required distribution data to develop the proposed fast stochastic model.

3.4 SUMMARY

In this chapter we propose a complete scheme including three models to simulate the device to device variation and stochastic behavior of memristor devices. These models are independent to device physical models and could be integrated together to study the overall impact on different memristor devices. The result form single device study could be used for higher level simulations.

4.0 EFFICIENT SIMULATOR FOR MEMRISTIVE CROSS-POINT ARRAY SIMULATION

After we have the variation models for single memristor devices, a efficient simulator specially designed for cross-point array simulation is needed because memristor cross-point array is the most common architecture for memristive circuits and also friendly to neuromorphic designs. In neural network algorithms, large matrix is needed to represent the synapse network among neurons, take the brain for example, one neuron is usually connected to more than one thousand of other neurons, thus to directly mimic the neuron network of the brain, at least a 1k-by-1k matrix is needed to represent the synapse network among 1k neurons. This corresponds to the similar size of cross-point arrays in hardware, which means, a simulator needs to at least simulate the dynamic behavior of 1 million memristors for each cross-point array during the simulation. However, current simulators are not fully aware of the importance of memristor cross-point array and no special design is applied to it to accelerate the simulation, and the simulation is very time-consuming. To author's best knowledge, so far no 1k-by-1k cross-point array has been simulated in the published works.

4.1 MATLAB-BASED SIMULATION OVERFLOW

The MATLAB simulation process is shown in Figure. 25. With experiment data and models of memristors and selectors, we want to automate the process of evaluating their performance as memory elements in crossbar arrays. It is preferred that the simulation process could be easily updated with new data and models, then with designed benchmarks the performance could be evaluated.

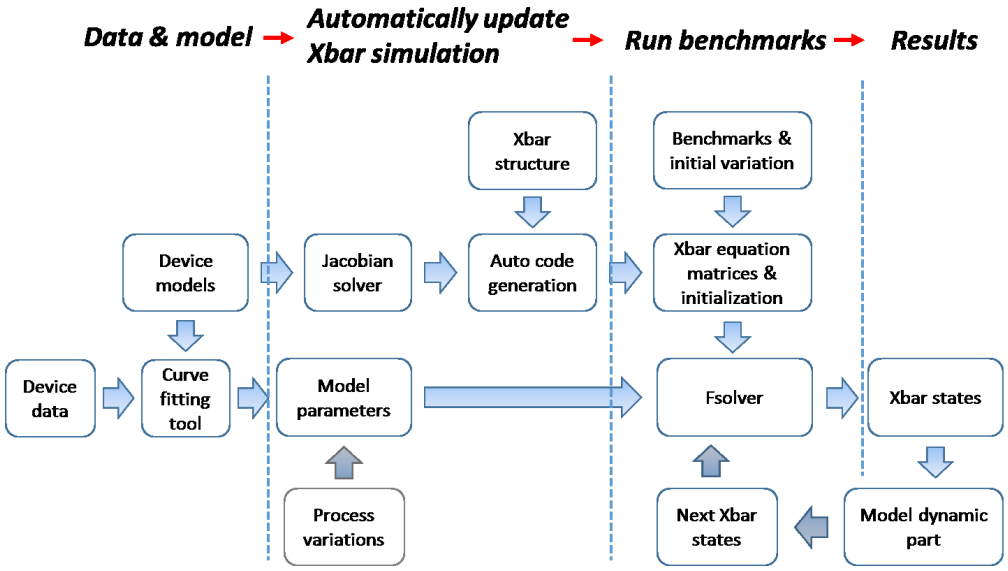

Figure 25: Matlab simulation flow

For a $N - by - N$ crossbar array, we could define the voltage on top of the memristor as $\mathbf{V_{top}}$, the voltage between memristors and selectors as $\mathbf{V_{mid}}$ and the voltage on bottom of selectors as $\mathbf{V_{bot}}$. The entire non-linear differential equations are coded in matrix format and stored as sparse matrix. Later it is solved using the *fsolver* provided by MATLAB, to speed up the simulation process, first, we pre-define the initial value for each voltage point in the crossbar, second, we pre-calculate the Jacobian matrix to get the trajectory for each variable. In this way, the special designed simulation process could achieve 2 3 orders of magnitude faster than the general purpose SPICE simulation process.

4.2 SIMULATION SETUP

In simulations, all devices data and model parameters are taken from macro scale devices to ensure the same scalability except the wire resistance is take from 50nm technology. The initial states of memristors and their variation are modelled from the same device sample.

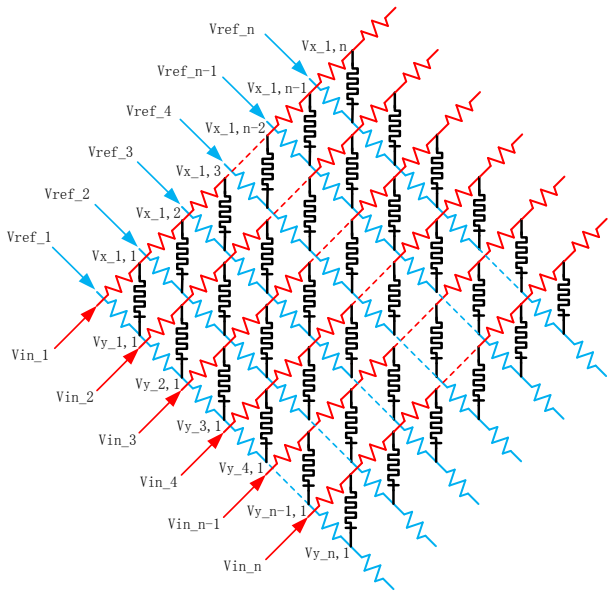

Figure 26: Memristor crossbar model

And the device to device variation is reflected by the noise in static model parameters of memristor and selector generated at the beginning of simulation, which will then held unchanged during the simulation. Last but not least, the stochastic switching behaviour is reflected by the noise in dynamic model parameters of memristors generated at each cycle of the simulations.

4.2.1 Cross-point array model

The memristor cross-point array contains two sets of parallel conductive interconnects crossing perpendicularly, and at every crosspoint there is a memristor, as shown in Figure. 26. Every memristor can be accessed by selecting the corresponding word-line and bit-line. Although passive memristor cross-point array has potentially the highest storage density benefited from memristor's simple structure ($4F^2$ per device), it has substantial sneak path leakage issue caused by half-selected and unselected devices, which will cause misprogramming and misreading. To clearly and accurately analyse the impact of sneak path, we use experiment verified device models rather than simplified switch models in the simulation.

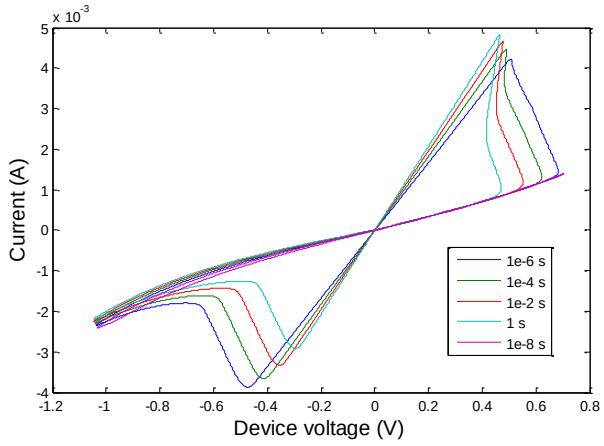

Figure 27: I-V curve of TaOx memristor device compact model

The interconnect resistance R_{wire} between two adjacent junctions is set to 0.65 Ω for a $4F^2$ cross-point structure according to the International Technology Roadmap for Semiconductors for 50 nm technology. The wire and device capacitance is not considered in this simulation, since its impact could be approximated as delay constants in addition to this work.

4.2.2 Memristor model

The memristor model used in this work is the latest TaO_x device model from HP lab Strachan et al. (2013). It accurately describes the static and dynamic behaviour of the memristor, especially reflect the time dependency of the switching behaviour, which is missing in the switch model. Figure. 27 shows the I-V curves of the device with fixed maximum/minimum voltages but different sweeping frequency. It could be observed that longer sweeping time (slower sweeping frequency) results in smaller switching voltage amplitude as well as more conductive ON-state. Thus misprograming may still happen when a device has a long enough signal applying across it, although its voltage amplitude is lower than the designed switching voltage. And this factor could only be reflected by dynamic simulations rather than transient analysis.

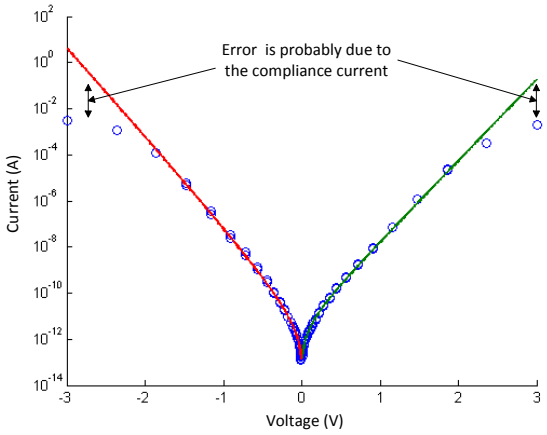

Figure 28: I-V curve of selector model based on TiN tunnelling device, the fitting error at high voltage probably comes from the compliance current limitation.

4.2.3 Selector model

Currently most memristors are linear or slightly non-linear devices, including the TaO_x device showed above. Although crossbar memory with linear devices could be written and read in small size, the severe sneaking path current leakage limits its further development. It is generally believed that non-linearity is necessary for the passive crossbar memory. One of the most important reasons is that introducing the non-linear element could greatly reduce or even eliminate the sneak path current leakage, thus increase the possible number of devices within one crossbar memory and alleviate the requirements on memristor's other properties, such as ON/OFF resistance, ON-OFF ratio and etc. Although memristor with intrinsic non-linearity is more preferred for fabrication, Considering the total device as a serial connection of linear memristor and non-linear selector is more helpful for the quantitative investigation of the non-linearity, and more practical at current stage.

We use tunnelling barrier device as our selector in this simulation. The selector's $I-V$ performance curve is shown in Figure. 28, which is well fit with Eq.(4.1) when $-2\text{V} < V < 2.5\text{V}$. a,b and c are fitting parameters. The curve differs from experimental data at high voltage amplitude ($> 2\text{V}$) because of the current compliance in measurement environment.

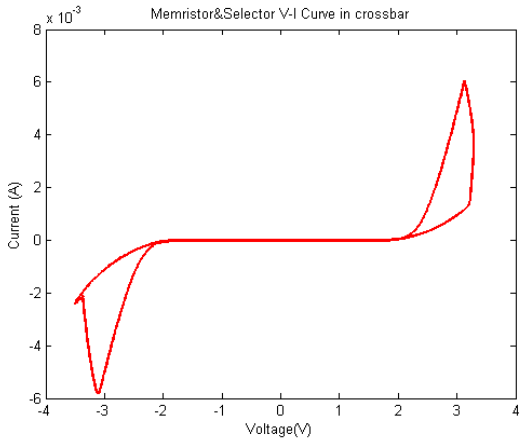

Figure 29: I-V curve of Nonlinear device with TaOx memristor in series connection with the TiN selector.

In this case, we could assume that without current compliance, the data points could follow the fitting curves and in that case the selector device may be eternally shorted.

$$I = a \cdot sinh(\frac{V}{b}) \cdot exp(\frac{|V|}{c}) \tag{4.1}$$

The $I - V$ performance curve of the integrated non-linear memristor device is shown in Figure. 29.

4.2.4 Variation model

The variation model of memristor we proposed in Chapter 3 could be used here. However, since we also have the latest variation data from HP labs for TaO_x memristor devices, we will just apply it here for our study.

4.2.4.1 Cycle-to-cycle variation model Many metal-insulator-metal based memristors have natural stochastic property. This property exhibits as two parts in a single device: first, the dynamic switching behaviour is stochastic, especially in slow-switching process excited by low switching voltage; second, the resulted static resistance states are stochastic,

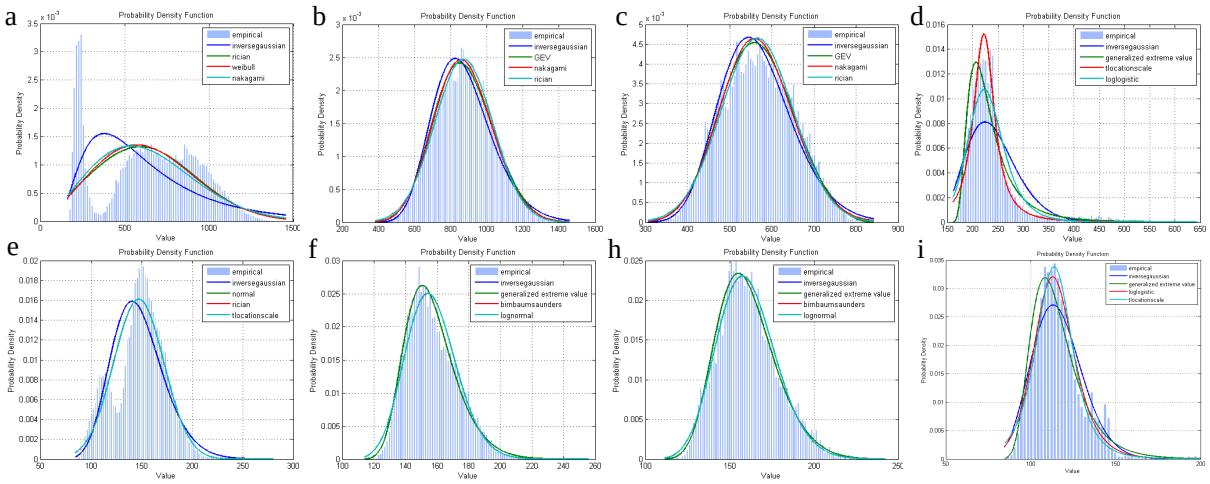

Figure 30: Distribution of HRS/LRS from the 2 billion on/off cycles of single TaOx device. a,e are the overall resistance distribution for HRS and LRS, respectively. b,c,d and f,h,i stand for the first billion cycles, next 0.5 billion cycles and the last 0.5 billion cycles for HRS and LRS, respectively.

appearing as the variations in ON state and OFF state resistance. The impact of stochastic switching could be avoided by using high switching voltage for fast switching or extended switching time to ensure the successful switching. However, the stochasticity in ON/OFF state resistance, or to say, cycle-to-cycle variation could not be ignored.

It is hard to explicitly study the cycle-to-cycle variation because its heavy coupling with device-to-device variations caused by immature fabrication process at current stage. As the fabrication process is also fast developing, it is more preferred to choose the best device now to study its property as our future standard. We select the best endurance data published so far for the same TaOx device, this device has satisfying endurance as well as stable switching behaviour and large ON/OFF ratio.

The distribution of 2 billion on/off cycles of single TaOx device is shown in Figure. 30. At a first glance the it seems hard to find a suitable *probability density function*(**pdf**) to fit the distribution since there exits multiple peaks. However, the peaks could be separated in three sets for both ON and OFF state data: 1 1 billion cycles, 1 1.5 billion cycles and 1.5 2 billions

cycles. By examining the six part of empirical data with 17 existing popular parametric distributions, we found that the *General Extreme Value*(**GEV**) distribution has the best fitness, as shown in Figure. 30. By the extreme value theorem, the GEV distribution is used as an approximation to model the maxima of long(finite) sequences of random variables. If we separate the memristor into many filaments, the overall resistance could be approximated as that of the most conductive filament if and only if the most conductive filament is much more conductive than all of others and nearly all the current travels though it. Thus, the nice fitness of GEV distribution implies that a good device with stable switching behaviour and high endurance should have only one stable functional conductive channel. In simulation we take the first 1 billion endurance data as our sample to extract the ON/OFF state resistance and initial state variation.

4.2.4.2 Device-to-device variation model The device-to-device variation in memristor crossbar could be improved after the fabrication process of memristor device is settled. So far the device-to-device variation in experiment is still too high and not within our interest. To consider the impact of device-to-device variation in future, we would like to assume it follows a Gaussian distribution or at least a lognormal distribution if the heavy tail issue could not be eliminated. Thus in simulation, we add Gaussian noise directly to the model, or on the parameter within the exponential part of the model, which will result in a near-lognormal distribution in current passing devices.

4.3 ACCURACY AND EFFICIENCY EVALUATION

The simulation result is verified with SPICE simulation. Since large scale simulation is too time consuming in LTSPICE, we only verified the transient simulation up to 64-by-64 crossbar size and the dynamic simulation up to 4-by-4 crossbar size. The simulation result are shown in Figure. 31. Both simulation result achieve perfect match between SPICE simulation and MATLAB simulation. To test the efficiency of our simulator, we conduct more simulations. The test platform is Thinkpad W520 8GB with MATLAB 2009b 32-bits.

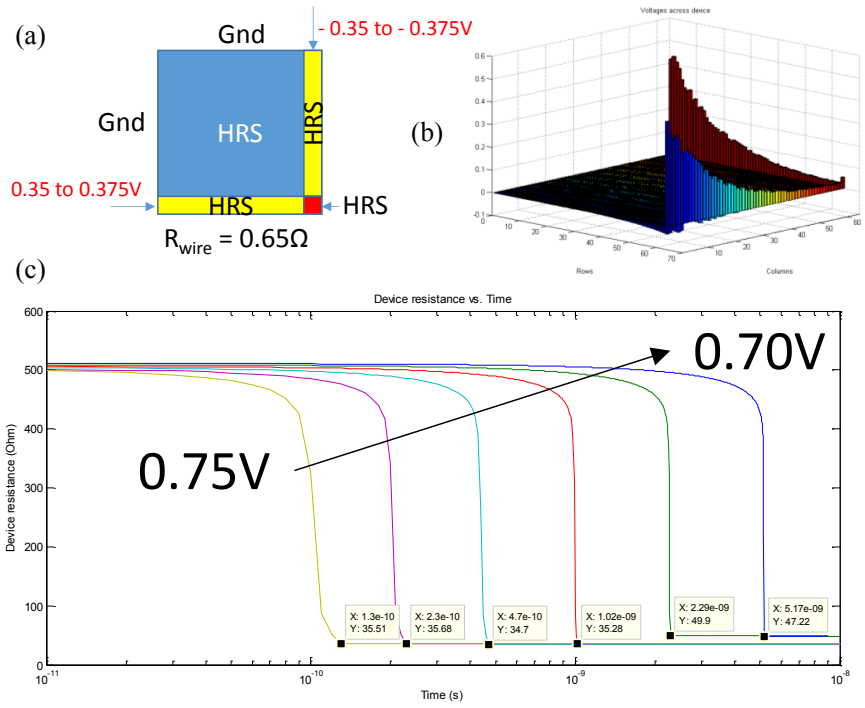

Figure 31: Simulation examples. (a) Simulation setup; (b) Static simulation result of voltage map for 64-by-64 linear cross-point array; (c) Dynamic simulation result of voltage on selected device for 4-by-4 linear cross-point array.

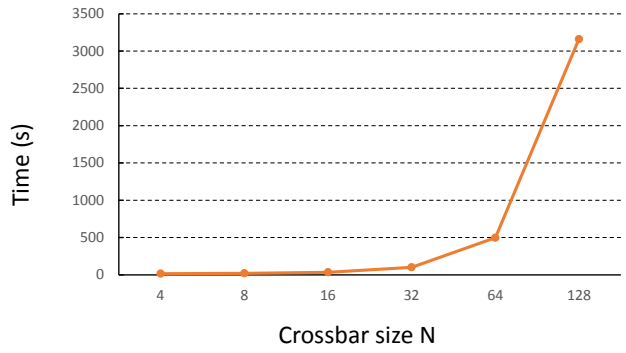

Figure 32: Simulation speed of linear cross-point array with different size.

For transient simulation of pure memristor cross-point array, it takes about 1 minute for 512-by-512 scale; for dynamic simulation of pure memristor cross-point array with 2000 steps and 10ps per step, the performance figure is shown in Figure. 32. We can observe that the simulation time linearly grows with the complexity. For comparison, the 128-by-128 scale simulation takes about 50 minutes and it is about 4 to 5 orders of magnitude faster than the LTSPICE simulator. For memristor+selector simulations, generally it is about 2 3 times slower than the pure memristor simulation because of the complexer model.

4.4 SUMMARY

In summary, linear memristor cross-point arrays are feasible for traditional level-based neurormorphic circuit design. However, the cross-point array size is limited to a small value(below 16-by-16 in current simulation), for neuromorphic design that requires larger neural network connections, selectors are necessary in the cross-point array and spiking neural network should be implemented to realize the neural network application.

5.0 MEMRISTOR-BASED NEUROMORPHIC CIRCUIT COMPONENTS

As we got variation model and circuit simulation tool for memristive circuits, we can begin the design of neuromorphic circuit. Similar to neural network models, neuromorphic circuit is composed of synapses for weight storage and neurons for computation. And in this chapter we will introduce a few memristor-based synapse and neuron designs.

5.1 "MARCO CELL" FOR HIGH-DEFINITION WEIGHT STORAGE

Storing high-precision continuous weight is beyond the capability of a single memristive switch. We proposed a macro cell design composed of a group of parallel connected memristive switches for weight storage.

5.1.1 Characterization of multiple memristive switches

Multiple memristive switches connected in serial or in parallel can provide multi-level conductance(resistance) values by simply combining the ON and OFF states of these devices. Comparing the two connection topologies, the design of parallel connection can be easily adapted on crossbar arrays. Also, it can provide a linear function of the read-out current, mitigating the pressure on sensing circuit. Thus, a group of parallel connected memristive switches is adopted in our design. The programming/detecting on the different ON and OFF combinations is realized through the peripheral circuit.

Here we take 9 parallel connected TiO_2 memristive switches as an example. Figure. 33 shows the distribution of its overall conductance G_{all}. To evaluate the impact of resistance

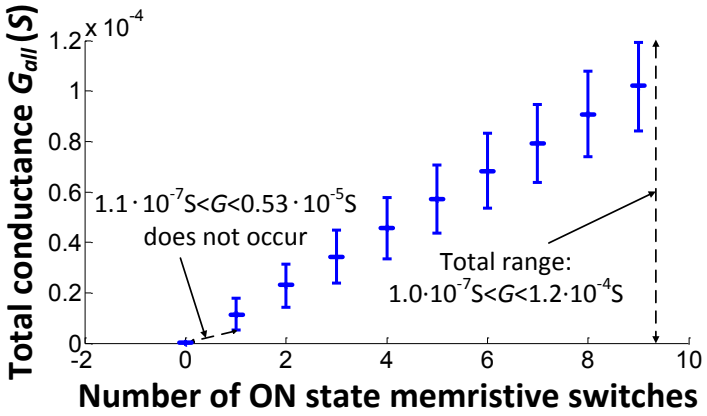

Figure 33: The conductance distribution of parallel connected memristive switches.

in OFF state, we gradually increase the device number in ON state and remain the others in OFF state. The simulation result shows that the mean and deviation of G_{all} linearly grows as the number of ON memristive switches increases. Moreover, when all the memristive switches are in OFF state, the variation is negligible, indicating the variation in OFF State has little impact on the total conductance. In other words, the ON-state variation dominates the distribution of G_{all}. Thus, with more memristors in a macro cell, it can achieve larger conductance range, roughly proportional to G_{ON} times number of memristors.

5.1.2 "Marco cell" design

The parallel connection of memristive switches can be easily adopted in crossbar arrays. Let's use a 3-by-3 memristive switch cross-point array in Figure 34(a) as an example. By combining the three inputs wires together and connecting the three output wires, the 9 memristive switches in this structure are parallel connected. We name such a structure as a **macro cell**.

The given example has 10 possible ON-OFF device combinations, corresponding to 10 different conductance levels. Ideally, the 10 conductance levels can be differentiated by tuning the number of memristive switches in ON state. Unfortunately, as the simulation result

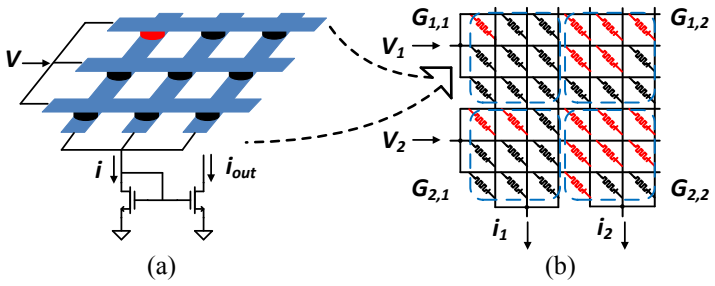

Figure 34: (a) A macro cell containing of 9 memristive switches on a 3x3 crossbar. (b) Partitioning a 6x6 memristive switch crossbar to obtain a 2x2 macro cell crossbar for continuous weight storage.

in Figure. 33 shows that the large resistance variation of ON state causes overlapping of conductance distribution, which is problematic in realizing traditional digitalized data storage for lacking of noise margin between adjacent levels. However, it also indicates continuous analog weight storage since a macro cell can achieve any arbitrary conductance within the overlapping range. For instance, the total conductance G_{all} of the macro cell in Figure 34(a) ranges from $0.53 \cdot 10^{-5} S$ to $1.2 \cdot 10^{-4} S$. The unreachable conductance ranges from $1.1 \cdot 10^{-7} S$ S to $0.53 \cdot 10^{-5} S$, corresponding to the region from the upper bound of 9 switches in OFF state to the lower bound when only one switch is in ON state.

The proposed macro cell design can be easily adopted on larger crossbar structure. Figure 34(b) shows an example in which a 6x6 memristive switch crossbar is partitioned into 4 macro cells to implement a 2x2 weight matrix. The design sacrifices density while offering a practical and reliable way to realize continuous resistance state for analog information storage and computation via binary switching memristors. The largest advantage of such a design is the dramatical decrement of programming complexity: a complex and slow feedback scheme is necessary when tuning a memristor to a specific analog state. In contrast, binary switching of memristors can adopt the existing memory programming scheme that is simple and reliable.

Compared with the crossbar array implementation by using floating gates or capacitors, the two-terminal memristor enables a much simpler structure. Moreover, the charge-based

CMOS devices require certain dimension to guarantee data accuracy, while the memristor technology can easily shrink to nanometer regime. Thus, though a macro cell design employs multiple memristors, it still can provide better area efficiency ($1 \sim 2$ orders of magnitude) over CMOS technologies.

5.1.3 Feedback attempt scheme

A simple feedback attempt scheme can be used to achieve target conductance in a macro cell. Figure. 33 illustrates the conceptual diagram of the programming scheme for demonstration purpose only. First, the number of memristive switches in ON state is determined to tune the overall conductance roughly. The output current is detected to check if the target conductance G has been reached. A feedback control then is given to finely tune the macro cell memristor conductance. If the detected current is not within the absolute error threshold E, an ON state memristor is randomly selected to reset and then set again. Under a given operation condition, the target conductance may not be obtained within a certain number of tryouts, indicating that either one or more memristors are too conductive or too resistive. We need to gradually reduce or increase the memristor number on ON state until the total conductance falls into tunable range. Then, restart the programming through the random attempt scheme.

In weight storage unit design, voltage pulses are used to control the switching of memristive switches. The pulse width t_{switch} is fixed, which is determined by the speed requirement. The amplitude V with $\sim 100\%$ switching probability is required to ensure the deterministic switching.

Figure 35 summarizes the average and the worst-case reconfiguration cycles to approach the different target conductance. The target conductance G_{Tar} can be generated by comparing to a reference current signal. Each data in the figures represents the statistical results of 1 million samples. The simulation results show that reconfiguration cycles increases linearly as the target conductance rises, while dramatically increases as the threshold error E decreases. More importantly, it shows that the proposed feedback attempt scheme can already achieve high precision programming within affordable attempts: any target conductance can be ap-

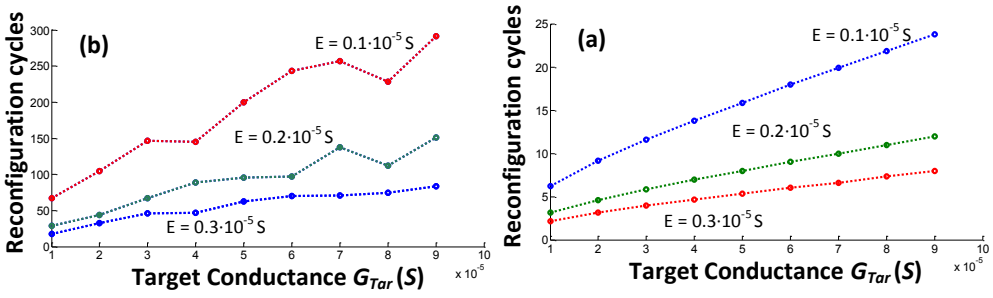

Figure 35: Average (a) and Worst-case (b) reconfiguration cycles to reach target conductance with different absolute error E.

proached within on average 25 attempts with the error of $E = 0.1 \cdot 10^{-5}S$, corresponding to only 1% of the achievable conductance range. In the worst-case study, when $E = 0.3 \cdot 10^{-5}S$, the macro cell reaches the target conductance within 50 attempts in most cases. A rough calculation of $(G_{max} - G_{min}))/2E$ implies that the macro cell can achieve at least 17 non-overlapping conductance levels rather than 10 levels obtained from ON/OFF combinations. If more attempts are affordable, we can increase to 50 non-overlapping conductance levels by reducing E to $0.1 \cdot 10^{-5}S$.

5.2 MEMRISTOR-BASED STOCHASTIC NEURON

The stochastic switching process is a severe issue for non-volatile memories with memristive switches. However, with careful design, it can be leveraged in designing stochastic neurons. Generally, stochastic neurons can be categorized into the binary neuron and the continuous value stochastic neuron.

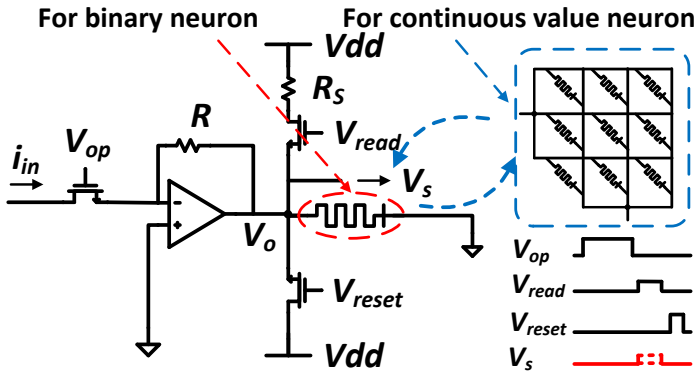

Figure 36: Binary/Continuous value stochastic neuron design

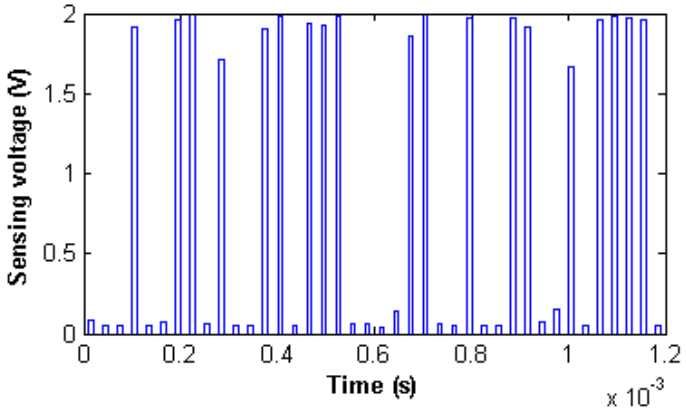

Figure 37: Output example of binary stochastic neuron

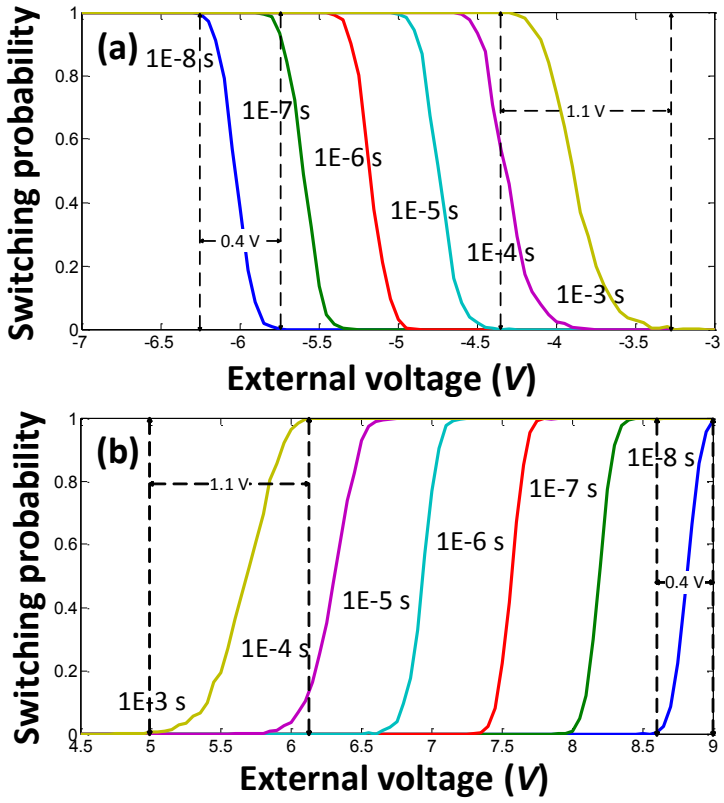

Figure 38: Voltage dependency of ON (a) and OFF (b) switching at different pulse widths t_{switch}

5.2.1 Binary stochastic neuron

Binary stochastic neuron generates random binary pulse signals, which uses external voltage signals to control the probability of 0 (OFF) or 1 (ON) generation. Figure. 36 illustrates the design of a binary stochastic neuron with a memristive switch. The operation timing diagram is given in the inner set of the figure. Figure. 37 shows a output pulse example of binary stochastic neuron. Figure. 38 shows the voltage dependency of ON and OFF switching of a TiO_2 memristive switch. Each curve has a fixed pulse width. The voltage dependency shows a normal dependency between the applied voltage and the switching probability, where t_{switch} has a log impact on the means and deviations of switching distributions.

Accordingly, the binary stochastic neuron can control the probability of random numbers by applying a fixed pulse width t_{switch} and adjusting voltage amplitude V. Figure. 38 also demonstrates the trade-off between t_{switch} and the tunable range of V. The longer pulse width results in the lower applied voltage and the wider tunable range, which alleviates the hardware design complexity but the speed of circuit operation exponentially reduces. The shorter pulse width makes the circuit run much faster, at the cost of smaller tunable range and the increased risk of device damage.

5.2.2 Continuous value stochastic neuron

Continuous value stochastic neuron generates random pulse signal. The voltage amplitude of the pulse signal is an analog value, which falls into a given distribution with controllable mean and noise. As illustrated in Figure. 36, a continuous value stochastic neuron can be constructed by replacing the single memristive switch in a binary stochastic neuron with a macro cell. The noise and mean are controlled by the external voltage signal and the number of memristive switches in a macro cell. An output example of continuous value stochastic neuron is shown in Figure. 39.

Figure. 40 shows the means and standard deviations of the noise generated by the proposed continuous value stochastic neuron. The designs with different macro cells containing $N = 4, 9, 16, 25$ memristive switches are compared. The means and the deviations of total conductance are controllable through the applied voltage. When a zero-mean noise signal is

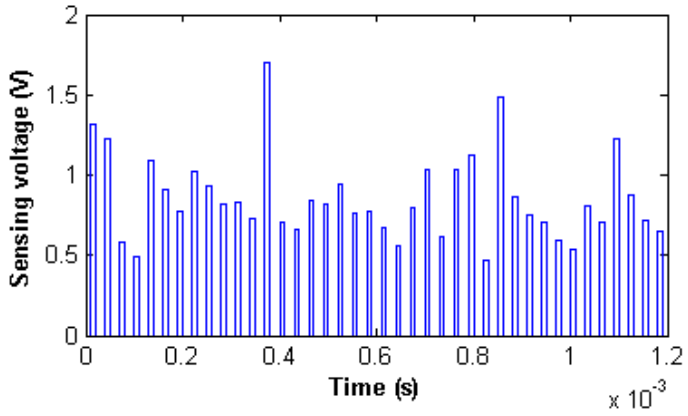

Figure 39: Output example of binary stochastic neuron

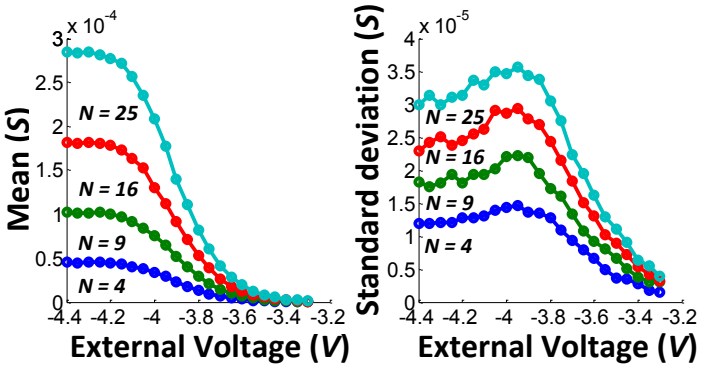

Figure 40: Voltage dependency of ON (a) and OFF (b) switching at different pulse widths t_{switch}

required, an offset current/voltage source can be added at the output V_s to cancel out the mean shifting considering that the voltage amplitude dependency of mean follows a normal CDF.

The variation comes from both the stochastic ON switching process and the randomness of ON state resistance. When $V > -3.95V$, the major contribution to variation comes from stochastic switching. Hence, the standard deviation decreases as the absolute voltage amplitude drops down. When $V < -3.95V$, a memristive switch has $> 80\%$ probability to successfully change to ON state, as shown in Figure. 38. Thus, the randomness of ON state dominates and the deviation is saturated. After all, using memristive switches, it is possible to replace the traditional continuous stochastic neuron with memristive switch based circuit to obtain higher area/power efficiency.

5.3 MEMRISTOR-BASED SPATIO-TEMPORAL(MST) SYNAPSE

In this work, we propose a compact spatio-temporal synapse design based on memristor technology, which supports both spatial and temporal weighting functions at the same time.

5.3.1 Design concept

A spatio-temporal synapse shall support two types of weighting functions:

- *Spatial weighting* that modulates the signal through the synapse. The spatial weight is tunable and adjustable in learning process whereas remains unchanged in recalls.
- *Temporal weighting* which is sensitive to the relative timing of pre-synaptic and post-synaptic signals in both recall and learning processes. The temporal weight reflects synapse's status of ON or OFF (that is, activated or deactivated), which is determined by the correlation strength of the two connected neurons.

Based on the concept, we propose a *memristor-based spatio-temporal* (MST) synapse that is depicted in Fig. 41(a). M_1 realizes the temporal weighting and the conductance of M_2 (*i.e.*, G_2) represents the spatial weight. For ease of explanation, we assume M_1 and M_2

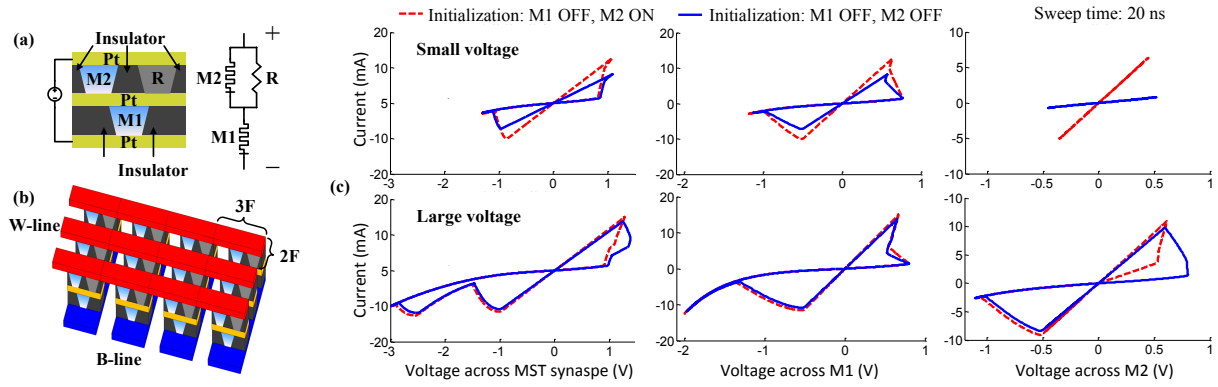

Figure 41: MST synapse design and I-V characteristics. (a) MST structure and equivalent circuit; (b) Conceptual layout in cross-point array; (c) DC-sweep I-V curves of the MST synapse, M_1, and M_2 when applying a small voltage (top figures) or a large voltage (bottom figure).

utilize the same device structure and therefore have the same ON and OFF resistance states (R_{ON} and R_{OFF}) and the identical switching dynamics.

To protect the spatial weight G_2 in recall, we connect M_2 with resistor R in parallel so as that the majority of the voltage across the synapse can apply to M_1. The following inequality corresponding to the worst-case situation when M_1 and M_2 are respectively at ON and OFF states shall be satisfied:

$$R_{\text{ON}} > R_{\text{OFF}} \cdot R/(R_{\text{OFF}} + R). \tag{5.1}$$

Assume $R_{\text{OFF}} = k \cdot R_{\text{ON}}$ and $R = x \cdot R_{\text{ON}}$, Eq. (5.1) turns to $k + x > kx$, which indicates that $x \leq 1$ shall be satisfied because $k > 1$. Here, we set $x = 1$ and make $R = R_{\text{ON}}$. By adjusting the voltage across the synapse, M_1 can be switched alone without impacting M_2. The state change of M_2, however, is always associated with M_1's switching.

The MST synapse's structure is also illustrated in Fig. 41(a). M_2 and R isolated by a thin insulator can be stacked above M_1. The overall structure is quite similar to a double layer *resistive random access memory* (RRAM) device. The layout in Fig. 41(b) shows that

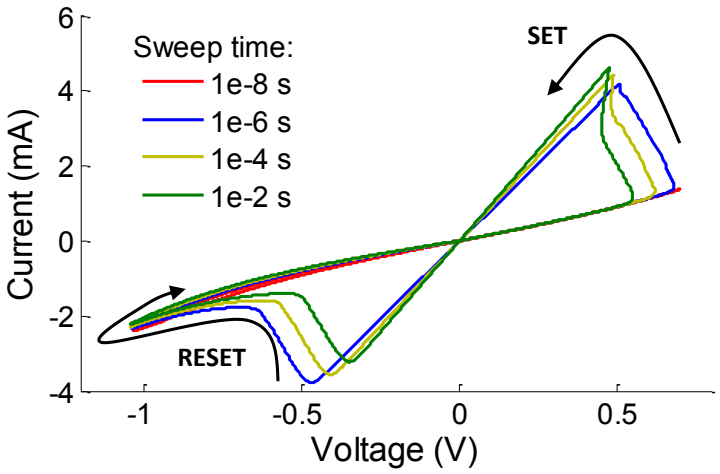

Figure 42: The I-V curve of a TaO_x device. Reproduced based on Strachan et al. (2013).

MST synapse has an area of $6F^2$, which is slightly larger than RRAM cell size of $4F^2$. Here, F represents the technology feature size. More important, the two-terminal structure can be easily integrated into cross-point array, which has been widely investigated and explored in high-density synapse design.

5.3.2 MST synapse characterization

To be more realistic and specific, we adopt the latest TaO_x device model that was strongly grounded in experimental data Strachan et al. (2013). TaO_x has been identified by HP Labs as one of the most promising memristor material for high density, low power, high endurance, and fast programming Jo et al. (2010); Yang et al. (2010). The *current-voltage* (I-V) characteristic based on TaO_x device model is shown in Fig. 42. A sufficient positive SET pulse (negative RESET pulse) makes the TaO_x device switch toward ON (OFF) state. The change of memristance not only depends on the voltage amplitude but also relates to the sweep speed. Note that the proposed MST synapse can be easily extended to other types of memristor technologies.

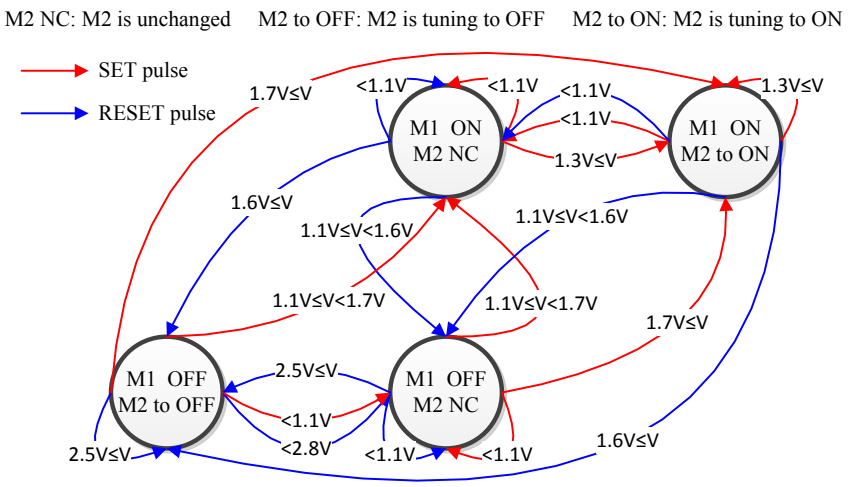

Figure 43: The state transition diagram of MST synapse under spiking excitations.

5.3.2.1 DC Response We first analyze the I-V characteristic of MST synapse and show the simulation results in Fig. 41(c). The corresponding I-V curves of M_1 and M_2 are also included for reference. Obviously, when the voltage on synapse is too small to meet TaO_x device's SET/RESET threshold under a given sweep time, neither M_1 nor M_2 can change. As the synapse voltage increases gradually, $M1$ will first reach switching condition. The top three I-V curves in Fig. 41(c) demonstrate that the temporal component $M1$ can be freely switched ON/OFF while the spatial component M_2 remains unchanged. The bottom three I-V curves in Fig. 41(c) implies that an even larger voltage amplitude is required to trigger the two-stage switching behavior including both M_1 switching and M_2 weight tuning. In summary, a MST synapse exhibits the weighted switching feature under a small excitation and its weight could be programmed if the applied voltage is sufficiently large.

5.3.2.2 Response to Spiking Excitation Spiking neural network is more favorable in hardware implementations for high power efficiency Seo et al. (2011). Instead of the complex synaptic signals in modulated format in Fig. 7, simple square pulses can be used for MST synapses as the input (output) spikes at the pre-neuron (post-neuron). To be consistent with TaO_x operation, we use SET/RESET pulses to represent the positive/negative voltage across a MST synapse. Here, we set the pulse width as $500ps$.

Fig. 43 summarizes the state transition diagram of MST synapse by varying SET/RESET pulses at different voltage amplitudes. Considering that the temporal component $M1$ is either `ON` or `OFF`[1] while the spatial component M_2 could be `unchanged`, `tuning to ON`, or `tuning to OFF`, a synapse could have six possible transition states. However, based on the analysis in Fig. 41(c), the situations of "M_1 `ON` & M_2 is `tuning to OFF`" and "M_1 `OFF` & M_2 is `tuning to ON`" can never occur in MST synapse operation and hence are excluded. Notably, M_2's tuning rate is not a constant value but determined by the present temporal state and spatial weight of synapse as well as the applied voltage.

5.3.3 Synaptic properties of MST synapse

In this section, the synaptic properties of MST synapse including the spike-timing-based recall ability and synaptic weight tunability will be examined and verified. Furthermore, STDP as the fundamental spike-timing-based learning rule will be applied to study MST synapse's learning characteristics.

5.3.3.1 Spike-timing-based recall In a spike-timing-based rcall, M_1 could switch to perform the temporal feature. For instance, the example in Fig. 44(a) shows that a pre-spike initializes a positive SET pulse through the synapse, making it activated. A post-spike results in a negative RESET pulse, which eventually deactivates the synapse. The spatial component M_2, however, remains at its initial value during the entire procedure. The spatial weighting function is reflected by the overall volume of charge through the synapse. To monitor the change of synaptic weight in recall, we apply a low DC signal of $0.2V$ in the simulation.

There are three types of typical timing situations in spike-timing-based recall. We examine and study the MST synapse behavior for each case separately.

Case 1: *The pre-neuron fires much faster than the post-neuron, forming a multi-spike train with many SET pulses and much fewer RESET spikes.* The MST synapse behavior under such excitations is shown in Fig. 44. M_1 turns ON at the first SET pulse and remains ON until a RESET pulse comes. Because of the over-tune induced memristance shift Hu et al.

[1]For simplicity, we refer M_1's `ON/OFF` state as the MST synapse state in the rest of this paper.

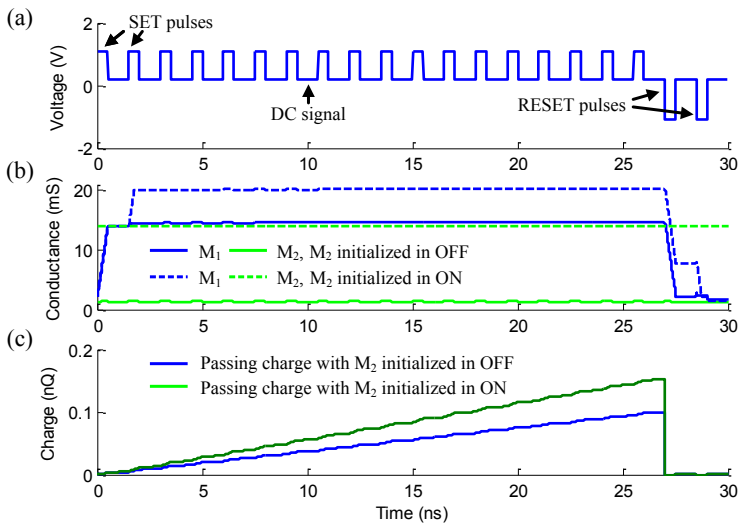

Figure 44: MST synapse in spike-timing-based recall – *Case 1*.

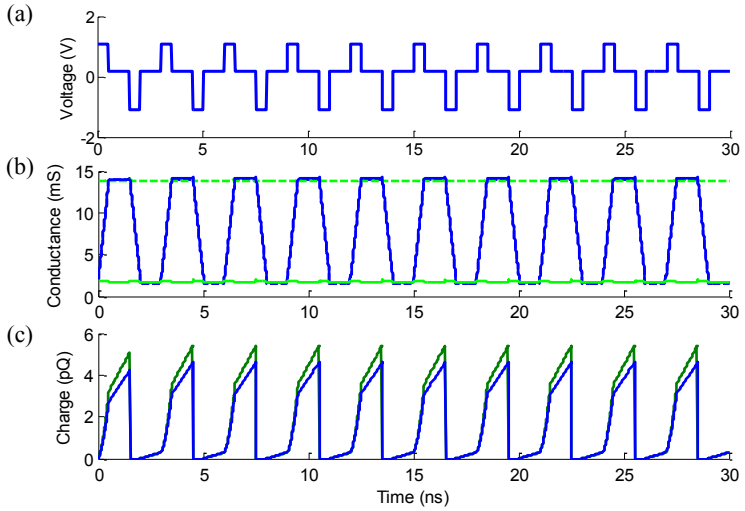

Figure 45: MST synapse in spike-timing-based recall – *Case 2*.

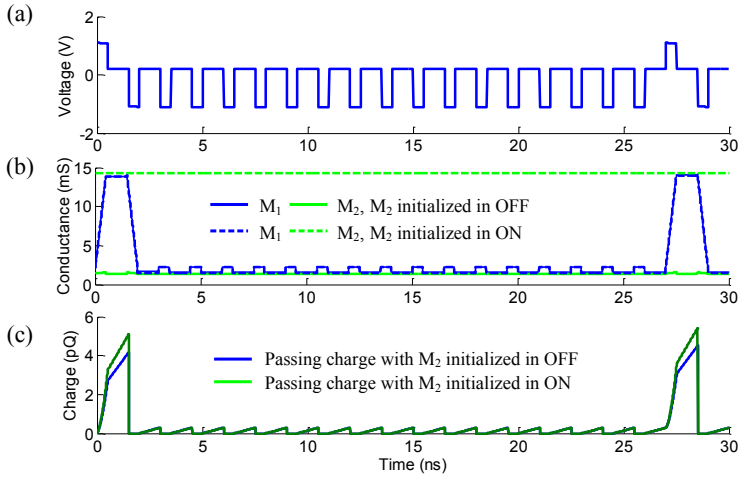

Figure 46: MST synapse in spike-timing-based recall – *Case 3*.

(2014), the conductance of M_1 could increase further if two SET pulses fire in consequence. Even though, M_1 can still be switched OFF by extending the duration or number of the RESET pulses. Notably, during the whole recall process, the value of M_2 remains constant and determines the total charge through the synapse. More specific, the charge accumulation is faster when M_2 is R_{ON} but much slower if M_2 is R_{OFF}.

Case 2: *If the pre- and post-neurons are strongly correlated, their firing events occur alternatively and appear as a sequence of SET/RESET pulses through the synapse.* Under this circumstance, M_1 (and the synapse) switches between ON and OFF states but M_2 is not affected, as shown in Fig. 45. The charge accumulation through the synapse is at a low rate when the synapse is OFF but increases significantly once the synapse turns on. This process is robust and repeatable.

Case 3: *A backward multi-spike train is generated when the burst of the post-neuron is much faster and more frequent than that of the prost-neuron.* As the opposite situation of *Case 1*, the spike-train through the synapse is in a form of many RESET pulses and a few SET pulses. During the sequence of RESET pulses, the synapse is OFF and thus not much charge can pass through it. Certainly, M_1 can be re-activated by any SET pulse. And M_2 keeps at the initial value without any change.

5.3.3.2 Weight tunability As the basis of learning process, the weight tunability of MST synapse including both LTP and LTD will be examined. Note that the tuning targets at only the spatial weight of M_2, not for M_1 that represent the activation status of the synapse. As shown in Fig. 43, spikes with large amplitude at the pre-/post-neurons (or *strong* SET/RESET pulses) enable the tuning process.

For ease of explanation, we give an example shown in Fig. 47. Assume a strong correlation exists between pre- and post-neurons. They fire alternatively, appearing as a sequence of SET/RESET pulses through the synapse. In the first 10 cycles, the spikes generated at the pre-neuron is stronger than at the post-neuron. As a result, M_2 gradually shifts toward ON state with better conductivity, successfully demonstrating the LTP feature. The scenario in the following 10 cycles is opposite: the stronger spikes at the post-neuron makes the RESET procedure more efficient. Thus, the effective conductance of M_2 when the synapse turns

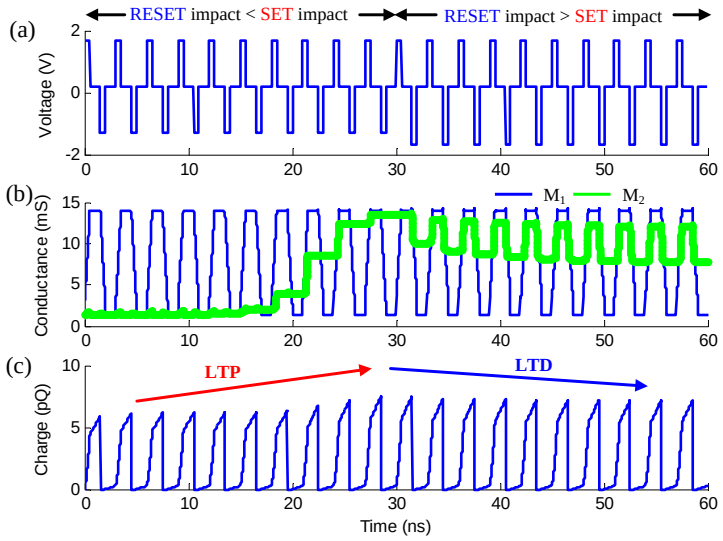

Figure 47: The tunability of MST synapse.

ON gradually reduces, implying a LTD behavior. The change of MST synaptic strength (conductance) is reflected by the charge passed through the synapse, as shown in Fig. 47(c).

In summary, the positive stimuli corresponding to *stronger*-SET:*weaker*-RESET combination enables LTP feature. And the LTD can be realized under the circumstance of *weaker*-SET:*stronger*-RESET pulses, implying negative stimuli.

Note that M_1 takes majority of the pulse voltage and hence can reach ON or OFF state all the times. On the contrary, the conductance of M_2 slowly changes because only a small amount of voltage applies on it. Moreover, a very asymmetric tuning curve of M_2 can be observed in Fig. 47(b). This is because TaO_x devices intrinsically have asymetricity in switching mechanism Strachan et al. (2013). Thus, the selection of memristor device has a critical impact on the tunability of MST synapse.

5.3.4 MST synapse with different memristors

Since the MST synapse design in Section III.A employees two identical memristors, a resistor R is needed to protect G_2 in recall process. However, the resistor R could be absorbed into the M_2 model, then the MST synapse design could be generalized to two memristors M_1 and M_2 with different low resistance states R_{ON}, high resistance states R_{OFF}, tuning-ON voltage

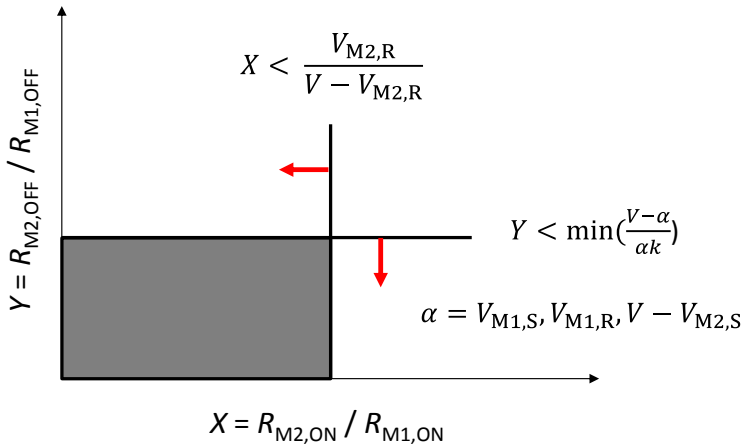

Figure 48: Resistance ratio between M_1 and M_2. Gray area is the working region that satisfies Eq. (5.2).

amplitudes V_S and tuning-OFF voltage amplitudes V_R, the following constrains need to be satisfied in recall process with pulse amplitude V for the worst-case scenarios:

$$\begin{cases} \frac{V_{M1,S}}{V} \leq \frac{R_{M1,ON}}{R_{M1,ON}+R_{M2,OFF}} & \frac{V_{M1,R}}{V} \leq \frac{R_{M1,ON}}{R_{M1,ON}+R_{M2,OFF}} \\ \frac{V_{M2,S}}{V} > \frac{R_{M2,OFF}}{R_{M1,ON}+R_{M2,OFF}} & \frac{V_{M2,R}}{V} > \frac{R_{M2,ON}}{R_{M1,ON}+R_{M2,ON}} \end{cases} . \tag{5.2}$$

To choose the combination of memristors, M_1 is firstly chosen as the reference, then the range of M_2 could be decided by Eq. (5.2). Fig. 48 gives a graphic view of the constrains in Eq. (5.2), $k = R_{M1,OFF}/R_{M1,ON}$, gray area is the working region that satisfies all the constrains. Without lose of generality, we assume $V_{M1,R} > V_{M1,S}$, solving the constrains in Fig. 48 generates the optimal V for maximum ON-OFF ratio of M_2. The result is listed in Eq. (5.3),

$$\begin{cases} V = V_{M1,R} + V_{M2,S} \\ Y = \frac{R_{M2,OFF}}{R_{M1,OFF}} = \frac{V_{M2,S}}{V_{M1,R} \cdot k} \\ X = \frac{R_{M2,ON}}{R_{M1,ON}} = \frac{V_{M2,R}}{V_{M1,R}+V_{M2,S}-V_{M2,R}} \\ \frac{R_{M2,OFF}}{R_{M2,ON}} = \frac{V_{M2,S} \cdot (V_{M1,R}+V_{M2,S}-V_{M2,R})}{V_{M1,R} \cdot V_{M2,R}} \end{cases} . \tag{5.3}$$

From Eq. (5.3), it shows that larger $V_{M2,S}$ helps increase the ON-OFF ratio of M_2 and k plays an important role in the maximum high resistance state of M_2.

5.4 SUMMARY

In this section we introduce the memristor-based neuromorphic circuit components including: "Marco cell", which enables multiple binary and unstable memristive switches to construct a multi-level and stable memristor device for high-definition weight storage; Memristor-based stochastic neuron, on the other side, take advantage of the stochastic behavior of memristor to construct compact true random number generator which is widely used in neuromorphic designs; Last but not least, a Memristor-based spatio-temporal(MST) synapse is proposed to enrich the functionality of synapse, expends its function from spatial analog value storage to spatio temporal weight storage. This is important for the efficient realization of spiking neural network on hardware. For these neuromorphic circuit components, "Marco cell" and stochastic neuron are based on TiO_2 memristor, while MST synapse is based on TaO_x memristor.

6.0 MEMRISTOR-BASED ANALOG NEUROMOPHIC CIRCUITS

In this chapter we introduce the hardware realization of level-based neural network. The Brain-State-in-a-Box neural network model is used as an example to study the performance of memristor-based neuromorphic circuit for spatial pattern recognition.

6.1 DESIGN METHODOLOGY

In this section, we will conceptually explain how to program a memristor cross-point array to store the information of connection matrix. One straightforward way, the *mapping* method, is to map the connection matrix to a memristor cross-point and then directly program the memristors to specified resistance values. A natural way, the *training* method, mimics the software training algorithm and adjusts the memristors iteratively to reach the required input/output function. Since there are many limitations in hardware implementation, a set of modifications to the algorithm are required.

6.1.1 Mapping a connection matrix to a cross-point array

6.1.1.1 Cross-point array vs. matrix Let's use the N-by-N memristor crossbar array illustrated in Figure 49 to demonstrate its the matrix computation functionality. Here, we apply a set of input voltages $\mathbf{V_I^T} = \{V_{I,1}, V_{I,2}, \ldots, V_{I,N}\}$ on the *word-lines* (WL) of the array, and collect the current through each *bit-line* (BL) by measuring the voltage across a sensing resistor. The same sensing resistors are used on all the BLs with resistance r_s, or conductance $g_s = 1/r_s$. The output voltage vector is $\mathbf{V_O^T} = \{V_{O,1}, V_{O,2}, \ldots, V_{O,N}\}$. Assume

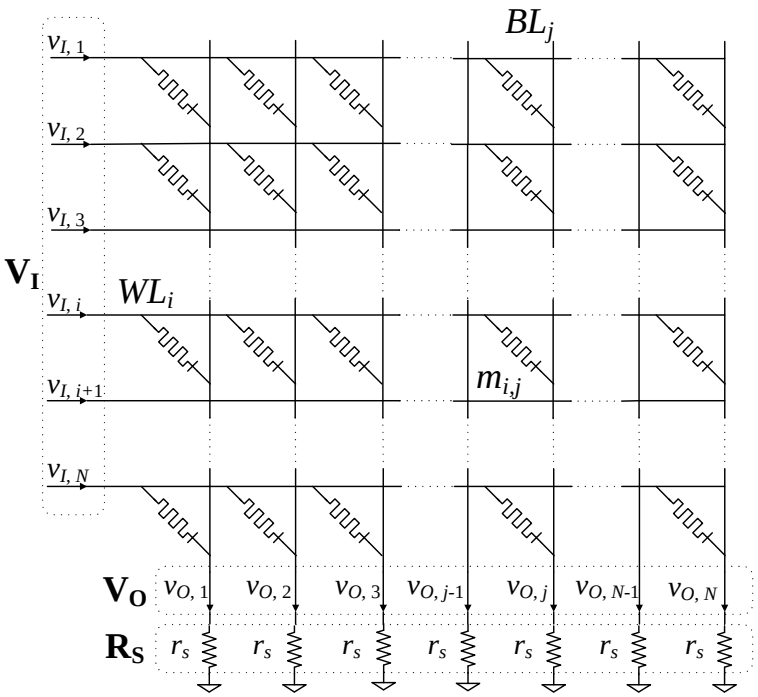

Figure 49: A memristor cross-point array

the memristor sitting on the connection between WL_i and BL_j has a memristance of $m_{i,j}$. The corresponding conductance is $g_{i,j} = 1/m_{i,j}$. Then the relation between the input and output voltages can be represented by

$$\mathbf{V_O} = \mathbf{C}\mathbf{V_I}. \tag{6.1}$$

Here, $\mathbf{C}$ can be represented by the memristances and the load resistors as

$$\mathbf{C} = \mathbf{D}\mathbf{G}^{\mathrm{T}} = diag(d_1, \ldots, d_N) \times \begin{bmatrix} g_{1,1} & \cdots & g_{1,N} \\ g_{2,1} & \cdots & g_{2,N} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,N} \end{bmatrix}^{\mathrm{T}}, \tag{6.2}$$

where, $d_i = 1/(g_s + \sum_{i=1}^{N} g_{i,j})$. To differentiate the mathematical connection matrix $\mathbf{A}$ in neural network, we use $\mathbf{C}$ to describe the physical relation between $\mathbf{V_I}$ and $\mathbf{V_O}$. Thus, all the terms in $\mathbf{C}$ must be positive values.

Please note that some non-iterative neuromorphic hardware uses the output currents $\mathbf{I_O}$ as output signals. Since the BSB algorithm discussed in this work is an iterative network, we take $\mathbf{V_O}$ as output signals, which can be directly fed back to inputs for the next iteration without extra design cost.

6.1.1.2 A fast approximation mapping function

Eq. 6.1 indicates that a trained memristor cross-point array can be used to construct the positive matrix $\mathbf{C}$, and transfer the input vector $\mathbf{V_I}$ to the output vector $\mathbf{V_O}$. However, $\mathbf{C}$ is not a direct one-to-one mapping of conductance matrix $\mathbf{G}$ as indicated in Eq. 6.2. Though a numerical iteration method can be used to obtain the exact mathematical solution of $\mathbf{G}$, it is too complex and hence impractical when frequent updates are needed.

For simplification, $g_{i,j} \in \mathbf{G}$ satisfies $g_{min} \leq g_{i,j} \leq g_{max}$, where g_{min} and g_{max} respectively represent the minimum and the maximum conductances of all memristors in the cross-point array. Thus, a simpler and faster approximation solution to the mapping problem is defined as:

$$g_{j,i} = c_{i,j} \cdot (g_{max} - g_{min}) + g_{min}. \tag{6.3}$$

A decayed version of $\mathbf{C}$, which is $\hat{\mathbf{C}}$ can be approximately mapped to the conductance matrix $\mathbf{G}$ of the memristive array. Plugging Eq. 6.3 in Eq. 6.2, we have

$$\hat{c_{i,j}} = \frac{c_{i,j} \cdot (g_{max} - g_{min}) + g_{min}}{g_s + (g_{max} - g_{min}) \cdot \sum_{j=1}^{N} c_{i,j} + N \cdot g_{min}}. \tag{6.4}$$

Note that many memristor materials, such as TiO_2 memristor, demonstrate a large g_{max}/g_{min} ratio Strukov et al. (2008a). Thus, a memristor at the high resistance state under a low voltage excitation can be regarded as an insulator, that is, $g_{min} \simeq 0$. Moreover, the BSB recall matrix $\mathbf{A}$ is a special matrix with a small $\sum_{j=1}^{N} c_{i,j}$. For example, all the BSB models used for character recognition in our experiments have $\sum_{j=1}^{N} c_{i,j} < 5$ when $N = 256$. And $\sum_{j=1}^{N} c_{i,j}$ can be further reduced by increasing the ratio of g_s/g_{max}. As a result, the impact of $\sum_{j=1}^{N} c_{i,j}$ can be ignored. These two facts indicate that Eq. (6.4) can be further simplified as

$$\hat{c}_{i,j} \approx c_{i,j} \cdot \frac{g_{max}}{g_s}. \tag{6.5}$$

In a summary, with the proposed fast approximation function, the memristor cross-point array performs as a decayed connection matrix $\hat{\mathbf{C}}$ between the input and output voltage signals.

6.1.1.3 Transformation of BSB Recall Matrix

To construct a memristor-based BSB recall circuit, our first task is to transfer the matrix $\mathbf{A}$ in the mathematical BSB recall model to a memristor array with physical meaning. A memristor is a physical device with conductance $g > 0$. Therefore, all elements in matrix $\mathbf{C}$ must be positive as shown in Eq. (6.2). However, in the original BSB recall model, $a_{i,j} \in \mathbf{A}$ could be positive or negative. We propose to split the positive and negative terms of $\mathbf{A}$ into two matrixes $\mathbf{A}^+$ and $\mathbf{A}^-$ as

$$a_{i,j}^{+} = \begin{cases} a_{i,j}, & \text{if} \quad a_{i,j} > 0 \\ 0, & \text{if} \quad a_{i,j} \leq 0 \end{cases}, \quad \text{and} \tag{6.6a}$$

$$a_{i,j}^{-} = \begin{cases} 0, & \text{if} \quad a_{i,j} > 0 \\ -a_{i,j}, & \text{if} \quad a_{i,j} \leq 0 \end{cases}. \tag{6.6b}$$

As such, Eq. 2.9 becomes

$$\mathbf{x}(t+1) = S\left(\mathbf{A}^{+}\mathbf{x}(t) - \mathbf{A}^{-}\mathbf{x}(t) + \mathbf{x}(t)\right), \tag{6.7}$$

Here for the default case we set $\alpha = \beta = 1$. The two connection matrices $\mathbf{A}^{+}$ and $\mathbf{A}^{-}$ can be mapped to two memristor crossbar arrays $\mathbf{M_1}$ and $\mathbf{M_2}$ in a decayed version $\hat{\mathbf{A}}^{+}$ and $\hat{\mathbf{A}}^{-}$, respectively, by following the mapping method in Eq. (6.3).

6.1.2 Training memristor cross-point arrays in BSB model

A software generated connection matrix can be mapped to the memristor crossbar arrays based on the assumption that every memristor in the crossbar could be perfectly programmed to the required resistance value. However, the traditional crossbar programming method faces accuracy and efficiency limitations due to the existence of the sneak paths Heittmann and Noll (2012). Although some recent works were presented to improve the write/read ability of memristor crossbars by leveraging the device nonlinearity Yang et al. (2012), the controllability of analog state programming is still limited. In spite of preparing the memristor crossbars with open-loop writing operations, we propose a *close-loop training method* which iteratively tunes the entire memristor crossbar to the target state. This technique is based on a modification of the software training algorithm.

Let's use the Delta rule in Figure. 5 as an example. A weight $a_{i,j}$ corresponds to the analog state of the memristor at the cross-point of the ith row and the jth column in a cross-point array. A weight updating $\Delta a_{i,j}$ involves multiplying three analog variables: α, $t_j - y_j$, and x_i. Though these variables are available in training scheme design, the hardware implementation to obtain their multiplication demands unaffordable high computation resources. Thus, we simplify the weight updating function by trading off the convergence speed as:

$$\Delta a_{i,j} = \alpha \cdot sign(t_j - y_j) \cdot sign(x_i). \tag{6.8}$$

Here, $sign(t_j - y_j)$ and $sign(x_i)$ are the polarities of $t_j - y_j$ and x_i, respectively. They could be either $+1$ or -1. $sign(t_j - y_j) \cdot sign(x_i)$ represents the direction of the weight change. At each training iteration, the rate of the weight changes remains constant which is controlled by the learning rate *alpha*.

The simplification minimizes the circuit design complexity meanwhile ensuring the weight change in the same direction as that of the Delta rule. As a tradeoff, more iteration may be required to reach a robust weight set since the weight change is not the steepest descent as that in Figure. 5. However, the modification will not affect the training quality much since the decision criterion of the algorithm remains as the error between output patterns and prototype patterns.

6.2 HARDWARE DESIGN AND ROBUSTNESS ANALYSIS

In this section, we will provide the design details of the recall and training circuits for BSB model. Then the types of noise affecting the quality of the BSB circuit will be classified and analyzed.

6.2.1 BSB recall circuit

To realize the BSB recall function at circuit level, we first convert the normalized input vector $\mathbf{x}(t)$ to a set of input voltage signals $\mathbf{V}(t)$. The corresponding functional description of the voltage feedback system can be expressed as:

$$\begin{aligned}\mathbf{V}(t+1) &= S'\left(G_1\mathbf{A}^{+}\mathbf{V}(t) - G_1\mathbf{A}^{-}\mathbf{V}(t) + G_2\mathbf{V}(t)\right) \\ &= S'\left(G_1\mathbf{V}_{\mathbf{A}+}(t) - G_1\mathbf{V}_{\mathbf{A}-}(t) + G_2\mathbf{V}(t)\right)\end{aligned}. \tag{6.9}$$

Here, G_1 and G_2 are the signal gain amplitudes resulted by peripheral circuitry, corresponding to α and β in Eq. 2.9. We use V_{bn} to represent the input voltage boundary, that is, $-V_{bn} \leq V_i(t) \leq V_{bn}$ for any $v_i(t) \in \mathbf{V}(t)$. The new saturation boundary function $S'()$ need to be modified accordingly. In implementation, V_{bn} can be adjusted based on requirements of convergence speed and accuracy. Meanwhile, V_{bn} must be smaller than V_{th}, the programming voltage threshold of memristor, so that the memristance values will not change during the recall process. Practically speaking, V_{bn} can be adjusted based on the requirement of convergence speed and accuracy.

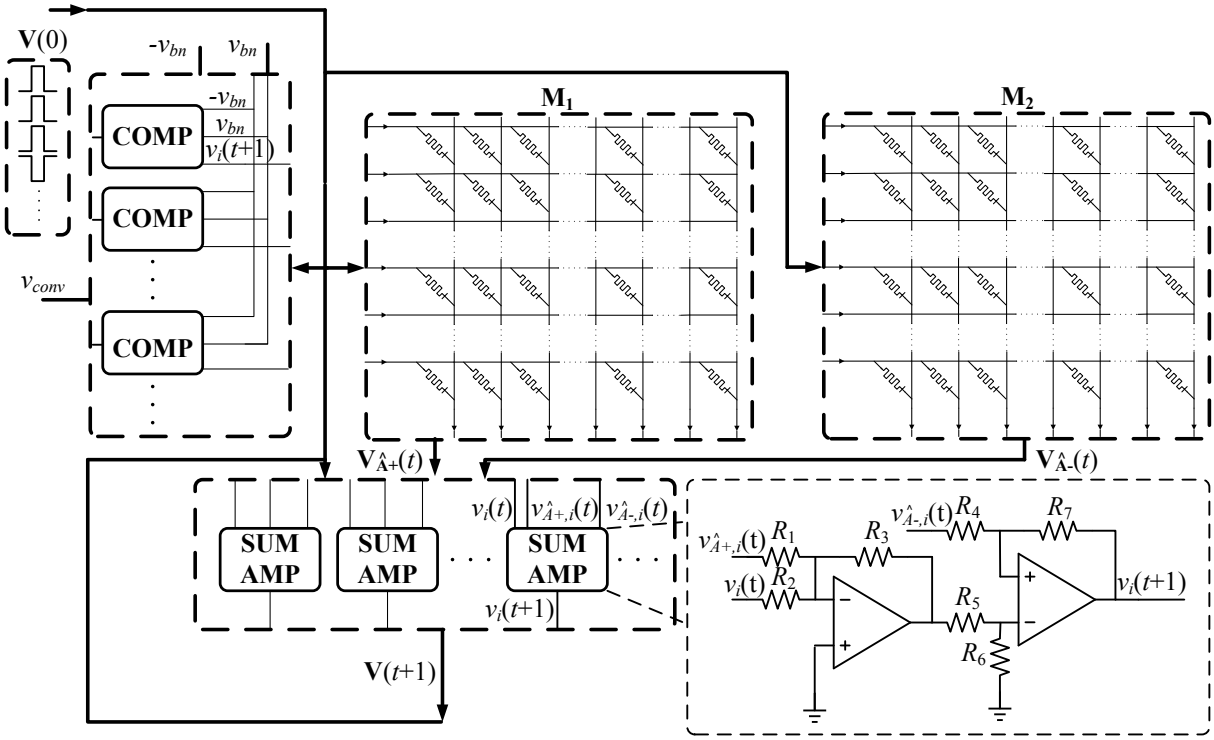

Figure 50: The conceptual diagram of the BSB recall circuit.

Figure. 50 illustrates the diagram of the BSB recall circuit built based on Eq. 6.9. The design is an analog system consisting of three major components. ***Memristor cross-point arrays***: As the key component of the overall design, the memristor cross-point arrays are used to realize the matrix-vector multiplication function in the BSB recall operation. To obtain both positive and negative weights in the original BSB algorithm in Eq. (2.9), two memristor cross-point arrays $\mathbf{M_1}$ and $\mathbf{M_2}$ are required in the design to represent the connection matrices $\hat{\mathbf{A}}^+$ and $\hat{\mathbf{A}}^-$, respectively. The memristor cross-point array has the same dimension as the BSB weight matrix $\mathbf{A}$.

Summing amplifier(SUM-AMP): In our design, the input signal $V_i(t)$ along with $V_{\hat{A}+,i}(t)$ and $v_{\hat{A}-,i}(t)$, the corresponding voltage outputs of two memristor crossbar arrays, are fed into a summing amplifier. The conceptual structure of the summing amplifier can be found in the inner set of Figure 50.

Resulted by the decayed mapping method proposed in Section 6.1.1.2, the required $V_{A+,i}(t)$ is g_s/g_{max} times of the generated $V_{\hat{A}+,i}(t)$. $V_{A-,i}(t)$ has the same requirement too. In our design, we set $R_1 = R_4 = R_6 = 1/g_s$ and $R_2 = R_3 = R_5 = R_7 = 1/g_{max}$. The resulting output of the summing amplifier

$$\begin{aligned} V_i(t+1) &= \frac{g_s}{g_{max}} \cdot V_{\hat{A}+,i}(t) - \frac{g_s}{g_{max}} \cdot V_{\hat{A}-,i}(t) + V_i(t) \\ &= V_{A+,i}(t) - V_{A-,i}(t) + V_i(t) \end{aligned}, \tag{6.10}$$

which indicates that the decayed effect has been canceled out.

The summing amplifier naturally conducts $S'()$ function by setting its output voltage boundary to $\pm V_{bn}$. Moreover, the resistance values $R1 \sim R7$ can be adjusted to match the required α and β in Eq. 2.9, if they are not the default value 1. For an N dimensional BSB model, N summing amplifiers are required.

Comparator: Once a new set of voltage signals $\mathbf{V}(t+1)$ is generated from the summing amplifiers, we send them back as the input of the next iteration. Meanwhile, every $V_i \in \mathbf{V}$ is compared to V_{bn} and $-V_{bn}$ to determine if path i has "converged". The recall operation stops when all the N paths reach convergence. Totally N comparators are needed to cover all the paths.

6.2.2 BSB training circuit

Figure. 51(a) summarizes the operational flow of the BSB training circuit. And the corresponding circuit diagram is illustrated in Figure. 51(b). Our goal is to develop a method to train the memristor cross-point arrays as auto-associative memories for prototype patterns. The training scheme leverages the recall circuit to verify the training result and generate the control signals.

***Step 1*: Initializing the crossbar arrays**. At the beginning of a training procedure, all memristance values in $\mathbf{M_1}$ and $\mathbf{M_2}$ are initialized to an intermediate value. The initialization does not have to be precisely accurate. Indeed, even when all of the memristors are all at either LRS or HRS, the crossbar arrays can still be successfully trained but it requires more time to reach convergence. For instance, training from HRS takes about 2,500 iterations under the setup of Experiment 1 in Section XXX, while initializing the memristors to intermediate states within their memristance range can reduce the iteration number to about 1,000.

***Step 2*: Selecting a prototype pattern** $\gamma^{(k)} \in B^n (k = 1, \ldots, m)$. Here, B^n is the n-dimension binary space $(-1, 1)$. Assume a training set includes m prototype patterns and each pattern $\gamma^{(k)}$ has the same probability to be chosen every time. The counter ST is used to record in sequence the number of patterns that have been successfully trained. When ST¿0, those patterns that have been trained are excluded from the selection.

***Step 3*: Sending** $\gamma^{(k)}$ **to the BSB recall circuit**. We convert $\gamma^{(k)}$ in binary space $(-1, 1)$ to a set of input voltages within the boundary $(-0.1V, 0.1V)$. These input signals are supplied to the two memristor cross-point arrays simultaneously. The resulting signals $\mathbf{V_O}$ can be obtained at the output of the BSB recall circuit.

***Step 4*: Error detection**. An error is defined as the difference between the prototype pattern and the recall result; that is, the difference between the input and output signals of the recall circuit. A piece of error detection circuitry for bit i is shown in Figure. 51(c), which generates only the direction of the weight change based on the simplified algorithm in Section 6.1.2. In total, N pieces of error detection blocks are needed for an N-by-N cross-point array.

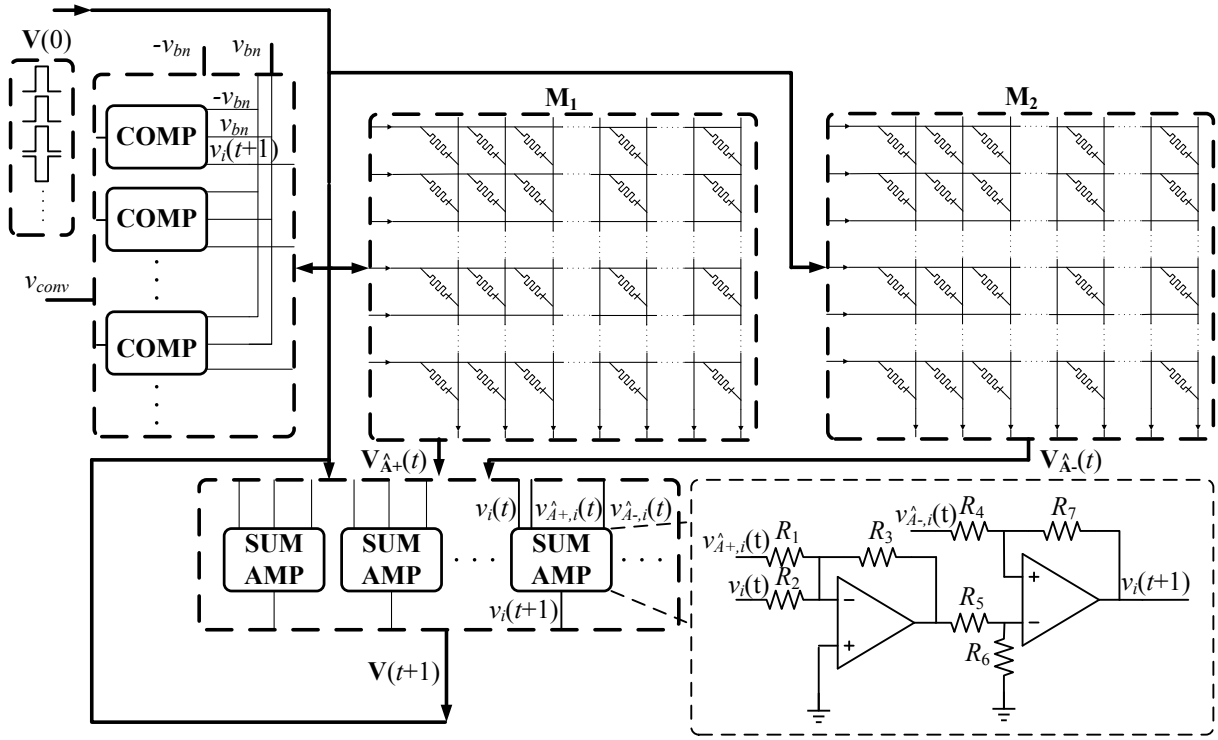

Figure 51: Embedded BSB training circuit: (a) training flow; (b) conceptual circuit diagram; (c) error detection circuit.

Considering that the range of $V_{out}(i)$ could be different from that of $V_{in}(i)$, we apply a scalar λ to the input vector and take $\lambda \cdot V_{in}(i)$ as the target output signal. Rather than generating $\lambda \cdot V_{in}(i)$ in every training, we use the preset threshold voltages for error detection. Since $V_{in}(i)$ is either $0.1V$ or $-0.1V$, four thresholds are needed, including

$$\begin{aligned} {V^+}_{th\_h} &= 0.1\lambda + \theta, {V^+}_{th\_l} = 0.1\lambda - \theta \\ {V^-}_{th\_h} &= 0.1\lambda - \theta, {V^-}_{th\_l} = 0.1\lambda + \theta \end{aligned}. \tag{6.11}$$

Here, θ represents the tolerable difference.

The error detection output $Diff(i)$ could be -1, 0, or 1. When $|V_{out}(i)\lambda \cdot V_{in}(i)| < \theta$, $Diff(i) = 0$, meaning the difference between the normalized $V_{in}(i)$ and $V_{out}(i)$ are so small that we consider them logically identical. Otherwise, $Diff(i) = +1$ or -1, indicating the normalized $|V_{out}(i)|$ is greater or less than the normalized $|V_{in}(i)|$, respectively.

***Step 5*: Training memristor crossbar arrays**. If **Diff** is not a zero vector, which means some error has been detected, the crossbar arrays need to be further tuned. The training signal generation is based on the training rule in Eq. 6.8. In order to control the training step with a finer granularity, we modify only one memristor crossbar each time. For example, one could train $\mathbf{M_1}$ or $\mathbf{M_2}$ when the iteration number is odd or even, respectively.

The weight updating of a memristor crossbar array is conducted by columns, during which constant voltage pulse signals are applied to $\mathbf{M_1}$ or $\mathbf{M_2}$. Note that the real resistance change of a memristor is also determined by its array location and device characteristics. In the design, such difference can be compensated by properly controlling the amplitude/width/shape of training pulses and paying more training iterations.

The polarity of the training pulse for the jth column are determined by $Diff(j)$. The design supplies the training pulses on all the rows of a memristor crossbar. The jth column is connected to ground and all the others are supplied with half of the training voltage. For $\mathbf{M_1}$, the training pattern is either the current selected prototype pattern $\gamma^{(k)}$ (if $Diff(j) = 1$) or its element-wise negated version (if $Diff(j) = -1$). The training signals to $\mathbf{M_1}$ and $\mathbf{M_2}$ have opposite polarities. That is, the training pattern of $\mathbf{M_2}$ uses the current prototype pattern when $Diff(j) = -1$ or its element-wise negated version when $Diff(j) = 1$.

Note that the mapping method uses $\mathbf{M_1}$ and $\mathbf{M_2}$ to represent the positive and negative terms of the BSB connection matrix, respectively. However, the proposed training scheme operated in real design circumstance cannot and does not have to guarantee an identical mapping to software generated matrix. In fact, what matters most is the overall effect of $\mathbf{M_1}$ and $\mathbf{M_2}$, not exact memristance values in each individual crossbar array.

***Step 6*: If training is completed**? The counter ST increases by 1 if a prototype pattern goes through Steps $2 \sim 5$ and reports no error without further tuning $\mathbf{M_1}$ and $\mathbf{M_2}$. Otherwise, ST is reset to 0 whenever an error is detected and all of the patterns in B^n are available in Step 2. ST$= m$ means the entire training set has been successfully "learned" and hence the training stops.

6.3 ROBUSTNESS OF BSB RECALL CIRCUIT

Running the BSB recall circuit constructed in Section 3.2.2 under ideal conditions should lead to the exact same results as the BSB mathematical algorithm. Unfortunately, the noise induced by process variations and signal fluctuations in implementation can significantly affect circuit performance. In this section, we will address the modeling of this noise at the component level. The impact of physical design constrains will also be discussed.

6.3.1 Process variations

6.3.1.1 Memristor cross-point arrays Due to process variations, the real memristance matrix $\mathbf{M}'$ of a memristor cross-point array could be quite different from the theoretical $\mathbf{M}$. Their difference can be represented by a *noise matrix* $\mathbf{N_M}$, which includes two contributors – the systematic noise $\mathrm{N_{M,sys}}$ and the random noise $\mathbf{N_{M,rdm}}$. Consequently, $\mathbf{M}'$ can be expressed by

$$\mathbf{M}' = \mathbf{M} \circ \mathbf{N_M} = \mathbf{M} \circ (1 + \mathrm{N_{M,sys}} + \mathbf{N_{M,rdm}}). \tag{6.12}$$

The impact of $\mathbf{N_M}$ on the connection matrix $\mathbf{C}$ is too complex to be expressed by a mathematical closed-form solution. But numerical analysis shows that it can be approximated by

$$\mathbf{C'_M} = \mathbf{C} \circ \mathbf{N_{CM}} = \mathbf{C} \circ \frac{1}{\mathbf{N_M}} \circ \frac{1}{\mathbf{N_M}}. \tag{6.13}$$

Here, $\mathbf{C'_M}$ is the connection matrix after including memristance process variations. $\mathbf{N_{CM}}$ is the corresponding noise matrix.

In the following analysis, we assume $\mathrm{N_{M,sys}}$ follows a normal distribution. To fully demonstrate the impact of the random process variations, the lognormal distribution is used to generate $\mathbf{N_{M,rdm}}$. Coefficient $\mathrm{Corr_M}$ is used to represent the correlation degree between the two memristor crossbar arrays in the same BSB circuit. When $\mathrm{Corr_M} = 1$, the two arrays have the same systematic noise.

6.3.1.2 Sensing Resistance

Similar to the analysis of memristance variation, we classify the noise induced by $\mathbf{R_S}$ variations into the systematic noise $\mathrm{N_{R,sys}}$ and the random noise $\mathbf{N_{R,rdm}}$. The actual sensing resistance vector becomes

$$\mathbf{R'_S} = \mathbf{R_S} \circ \mathbf{N_{Rs}} = \mathbf{N_S} \circ (1 + \mathrm{N_{R,sys}} + \mathbf{N_{R,rdm}}). \tag{6.14}$$

$\mathbf{C'_R}$, the connection matrix after including $\mathbf{N_{Rs}}$, is

$$\mathbf{C'_R} = \mathbf{C} \circ \mathbf{N_{CR}} = \mathbf{C} \circ [\mathbf{N_{Rs}}\ \mathbf{N_{Rs}}\ \ldots\ \mathbf{N_{Rs}}]\,. \tag{6.15}$$

Here, $\mathbf{N_{CR}}$ is the noise matrix of $\mathbf{C}$ after including the process variation of the sensing resistors. The mean value of r_s distribution, which reflects the impact of systematic noise, can be obtained during the post-fabrication testing. When training the memristances in BSB circuit, $\mathrm{N_{R,sys}}$ should have been included. Hence, in the following analysis, we only consider the random noise $\mathbf{N_{R,rdm}}$, which follows a normal distribution.

6.3.2 Signal fluctuations

The electrical noise from the power supplies and the neighboring wires can significantly degrade the quality of analog signals. Different from the process variations that remain unchanged after the circuit is fabricated, these signal fluctuations vary during circuit operation. Without loss of generality, we assume the run-time noise of the summing amplifier's output signals follows a normal distribution, same as that of the output of the comparators.

6.3.3 Physical challenges

There are three major physical constrains in the circuit implementation: (1) For any $V_i(0) \in \mathbf{V}(0)$, The voltage amplitude of initial input signal $V_i(0)$ is limited by the input circuit, ; (2) boundary voltage V_{bn} must be smaller than V_{th} of memristors; and (3) the summing amplifier has finite resolution.

In the BSB recall function, the ratio between boundaries of $S(y)$ and the initial amplitude of $x_i(0)$,$x_i(0) \in \mathbf{x}(0)$, determines the learning space of the recall function. If the ratio is greater than the normalized value, the recall operation takes more iterations to converge with a higher accuracy. Otherwise, the procedure converges faster by lowering stability. Thus, minimizing the ratio of $|V_i(0)|$ and V_{bn} can help obtain the best performance. However, the real amplifier has a finite resolution and V_{bn} is limited within V_{th} of the memristor. Continuously reducing $|V_i(0)|$ eventually will lose enough information in the recall circuit. So the resolution of the summing amplifier is a key parameter to determine the optimal ratio of $|V_i(0)|$ and V_{bn} in circuit implementation. Certainly it also affects the design cost of amplifier and the overall design.

6.3.4 Impact of sneak paths

When utilizing crossbars as memories, only one WL is raised up and one or a few BLs are accessible at a time (See Figure. 49). The other WLs and BLs remain floating. Such a *single-input-single-output* (SISO) access inevitably results in currents through unintended paths, called the sneak paths . The existence of sneak paths in the passive resistive network is a well-known issue, which greatly limits the size of crossbar arrays and their utilization in memory design. During a recall process, our proposed design accesses the crossbar in *multi-input-multi-output* (MIMO) mode therefore the sneak path is not as critical as in SISO mode.

Training process is a little bit tricky. We set the training voltage slightly higher than V_{th} but much smaller than the switching voltage, because only a small change of memristance is needed in each training step. Hence, the voltage drop on the other memristors is smaller than V_{th}, therefore will not result in the unexpected memristance changes.

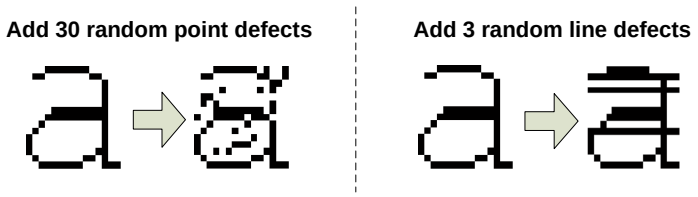

Figure 52: (a) Random line defects; (b) Random point defects.

More importantly, the sneak paths have to be well controlled in memory design because the current through the target device is a critical parameter. In contrast, the major concern in neuromorphic design is that the association between input and output signals can be properly captured (in training) and reflected (in recall) by the memristor crossbar array. The current distribution within the crossbar is not in the area of interest.

6.4 SIMULATION RESULTS I - RECALL

The robustness of the BSB recall circuit was analyzed based on Monte-Carlo simulations at the component level. The experimental setup is summarized in Figure. 53. Memristor device parameters are taken from Strukov et al. (2008a).

We tested 26 BSB circuits corresponding to the 26 lower case letters from "a" to "z". Each character image consists of 16×16 points and can be converted to a 256-entry vector. Accordingly, the BSB recall matrix has a dimension of 256×256. In each test, we created 500 design samples for each BSB circuit and ran 13,000 Monte-Carlo simulations. Two types of input pattern defects, random point defects and random line defects (see Figure 52), have been evaluated.

6.4.1 BSB circuit under ideal condition

For an input pattern, the different BSB circuits have different convergence speeds. Figure 54 shows an example when processing a perfect "a" image through the BSB circuits trained for all 26 lower case letters. The BSB circuits for "a", "l", and "s" reach convergence with

SIMULATION SETUP

Recall Circuit	Memristor Parameters				
	R_H (Ω)	R_L (Ω)	h (nm)	μ_v ($m^2 \cdot s^{-1} \cdot V^{-1}$)	V_{th} (V)
	10000	1000	10	$1.00E^{-14}$	1.1
	Summing Amplifier Parameters				
	G_1	G_2 (Recall/Train)		V_{op+} (V)	V_{op-} (V)
	30	0.6/0		1.05	-1.05
	Comparator		Sensing Res.	Recall Voltage V_R (V)	
	V_{ref_h} (V)	V_{ref_l} (V)	R_S (Ω)	For "1"	For "-1"
	1.0	-1.0	1000	0.1	-0.1
Training Circuit	Memristance movement ΔM				
	Linear model ±3Ω each step		Nonlinear model $\frac{(R_H - R_L) \cdot \mu_v \cdot R_L \cdot V_T \cdot t}{h^2 \cdot M}$		
	Comparator		Training time	Training voltage V_T (V)	
	V_{th_h} (V)	V_{th_l} (V)	t (μs)	For "1"	For "-1"
	±0.125	±0.115	10	1.5	-1.5

Figure 53: Simulation setup of BSB recall circuit.

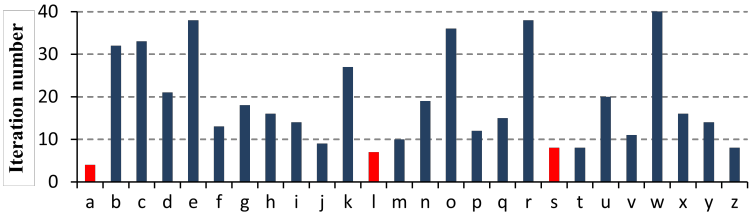

Figure 54: Iterations of 26 BSB circuits for a perfect "a" image.

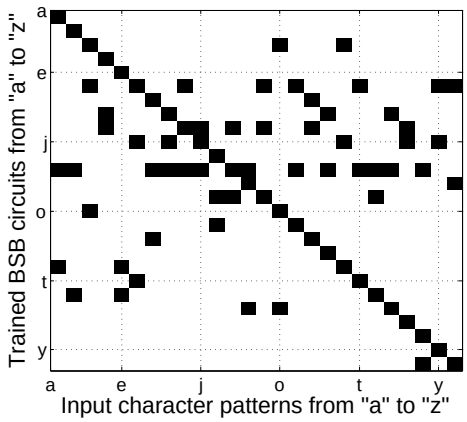

Figure 55: The ideal performance of 26 BSB circuits

the least iteration numbers. The multi-answer character recognition method consider these three letters as winners and send them to word-level language model Wu et al. (2011).

Figure 55 summarizes the performance of the BSB circuit design under ideal condition without input defects, process variations, or signal fluctuations. The x-axis and y-axis represent input images and the BSB circuits, respectively. All the winners are highlighted by the black blocks. Figure 55 shows that a BSB circuit corresponding to its trained input pattern always wins under the ideal condition. However, after injecting noise to input pattern or circuit design, some BSB circuits might *fail* to recognize its trained input pattern. In this work, we use the probability of failed recognitions P_F to measure the performance of BSB circuits.

6.4.2 Process variations and signal fluctuations

Impact of random noise: The random noise in the BSB circuit could come from process variations as well as electrical signal fluctuations. We summarize the impact of every single random noise component in Table 9, based on Monte-Carlo simulation results. Here, we assume two memristor crossbar arrays are fully correlated, *i.e.*, $\text{Corr}_{\text{M}} = 1$. The simulation results show that BSB circuit design has a high tolerance on the random noise: compared to the ideal condition without any fluctuation ("IDEAL"), these random noise of circuits

Table 9: $P_F(\%)$ of 26 BSB circuits for 26 input patterns.

random point numbers	0	10	20	30	40	50
IDEAL	0	2.1	4.2	5.3	10.0	20.8
$M(\sigma_{sys} = 0.1\&\sigma_{rdm} = 0.1)$	0	1.9	4.6	6.5	14.2	24.7
$R_S(\sigma = 0.1)$	0	1.8	4.3	6.2	13.7	24.1
SUM-AMP($\sigma = 0.1$)	0	1.9	4.4	7.7	13.5	23.1
COMPARATOR($\sigma = 0.1$)	0	2.3	5.5	5.4	11.1	22.0
$Corr_M = 0.6$	5.6	10.2	17.2	22.7	30.8	38.6
OVERALL ($Corr_M = 0.6$)	4.6	8.2	15.2	20.7	32.8	36.6
random line numbers	0	1	2	3	4	5
IDEAL	0	7.3	13.8	21.5	35.8	50.2
$M(\sigma_{sys} = 0.1\&\sigma_{rdm} = 0.1)$	0	7.4	14.8	25.5	38.8	53.6
$R_S(\sigma = 0.1)$	0	7.4	14.8	23.3	35.1	51.8
SUM-AMP($\sigma = 0.1$)	0	7.7	15.3	23.4	34.7	52.6
COMPARATOR($\sigma = 0.1$)	0	6.9	14.5	23.3	33.7	53.2
$Corr_M = 0.6$	5.1	14.4	24.7	34.6	44.2	55.1
OVERALL ($Corr_M = 0.6$)	6.3	15.4	24.2	34.1	44.0	58.2

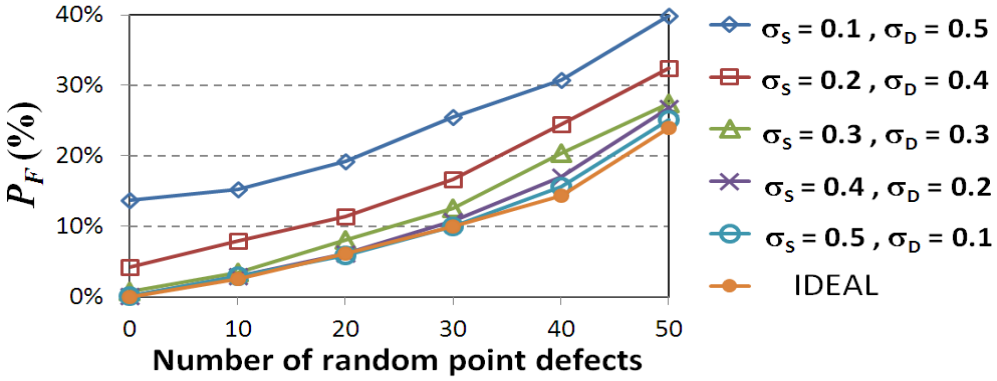

Figure 56: Static noise vs. dynamic noise.

cause slight performance degradation. This is because resilience to random noise is one of the most important inherent features for the BSB model as well as other neural networks.

Static Noise vs. Dynamic Noise: The noise matrices $\mathbf{N_M}$ and $\mathbf{N_{Rs}}$ mainly affect the mapping between connection matrix and memristor crossbar array. Physically noise components come from process variations and remain unchanged during the recall operation. Thus, they can be regarded as ***static noise*** N_S. On the contrary, the noise from the summing amplifiers and comparators are induced by electric fluctuations, which demonstrates a dynamic behaviour during the iteration process. We classify them as ***dynamic noise*** N_D.

We can adjust N_S and N_D and observe the combined impact on BSB circuit performance. For simplicity, we set $\sigma_{rdm}(M) = \sigma(R_S) = \sigma_S$ and $\sigma(AMP) = \sigma(COMP) = \sigma_D$. And $\mathrm{Corr_M} = 1$ to exclude the impact of correlations between the two memristor arrays. The result in Figure 56 shows that the dynamic noise dominates P_F. For example, when $\sigma_D = 0.5$ and $\sigma_S = 0.1$, P_F is high even with a clean input image. Decreasing σ_D but increasing σ_S results in P_F reduction in all regions.

Impact of $\mathrm{Corr_M}$: The BSB circuit implementation uses two memristor crossbar arrays to split the positive and negative elements of $\mathbf{A}$. Reducing $\mathrm{Corr_M}$ and hence increasing the difference in the systematic noise of two memristor arrays can be regarded as $\mathbf{A}^+$ and $\mathbf{A}^-$ having different overall shifts. This is ***directional noise*** in the recall function. As a consequence, $\mathrm{Corr_M}$ demonstrates a higher impact. As shown in Figure 57, when decreasing $\mathrm{Corr_M}$ from 1 to 0, the average P_F dramatically increases.

6.4.3 Impact of summing amplifier resolution

To achieve the same learning space as the normalized BSB model, we set $v_{bn} = 1.6V$ and all elements of $\mathbf{V(0)}$ to be $\pm 0.1V$. Then we vary the summing amplifiers' resolution under different static and dynamic noise configurations. $\mathrm{Corr_M}$ was fixed at 0.6. The simulation results are shown in Figure 58.

Again the simulation results demonstrate the BSB circuit's high tolerance for random noise: when $\sigma_S = \sigma_D \leq 0.4$, P_F is close to the ideal condition of $\sigma_S = \sigma_D = 0$. A 200mV resolution for the summing amplifier is too coarse to be acceptable: the BSB circuit cannot

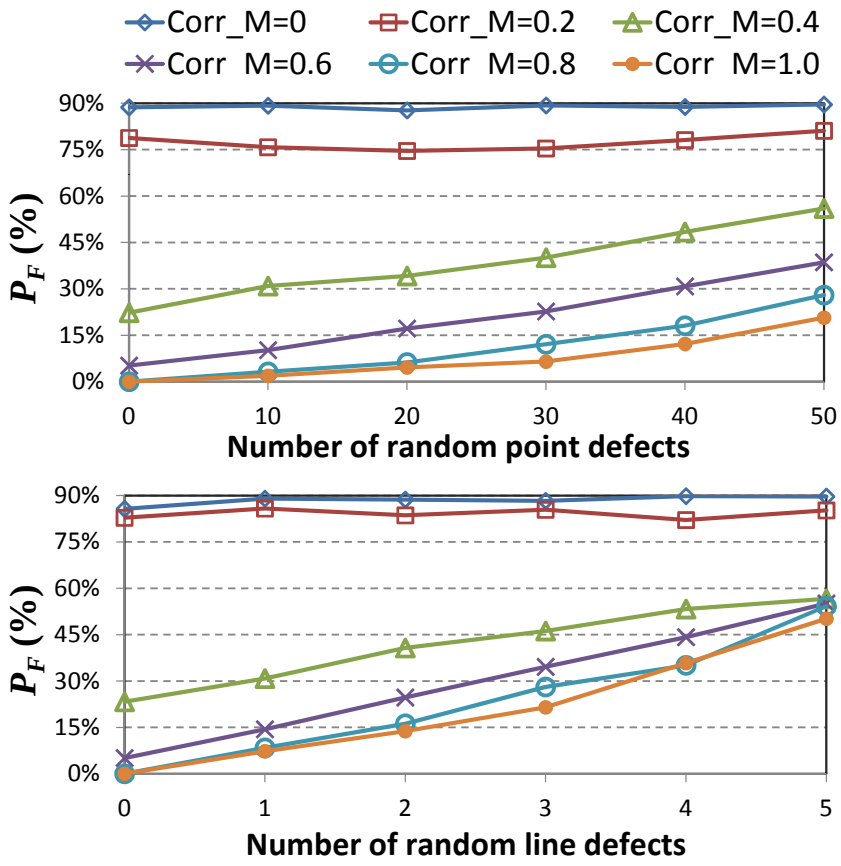

Figure 57: The impact of $\mathrm{Corr_M}$.

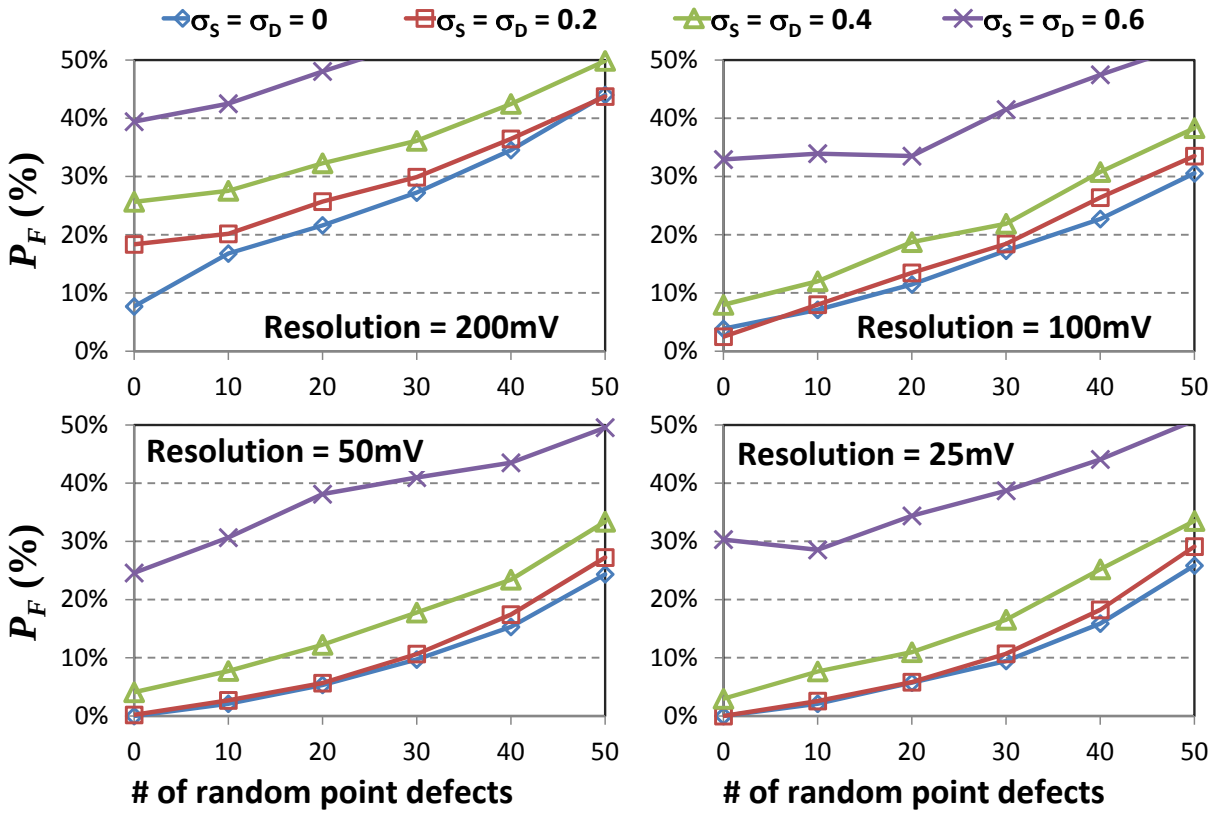

Figure 58: Impact of resolutions of summing amplifiers.

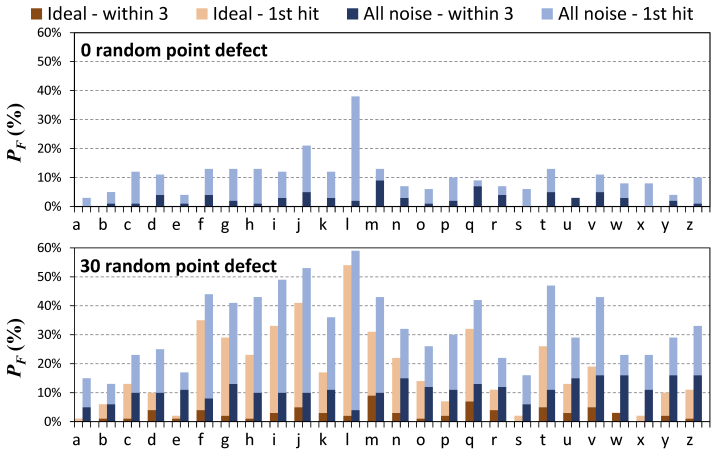

Figure 59: P_F for each character pattern

have zero P_F even under the ideal condition when neither input defects nor random noise are included. The resolution of 100mV is acceptable when the noise is not significant (*e.g.*, $\sigma_S = \sigma_D \leq 0.2$) and the input pattern defect number is small (*e.g.*, less than 20 random point defects). For the given physical constraint configuration, the 50mV and 25mV resolutions show similar results when $\sigma_S = \sigma_D \leq 0.2$.

6.4.4 Overall performance

In the previous analysis, we use the averaged P_F of all 26 BSB circuits for performance evaluation. One thing of particular interest is whether all BSB circuits degrade in the same way as we inject defects and noise into the system, or perhaps certain BSB circuits perform much better or worse than the others. In this test, we set $\text{Corr}_\text{M} = 0.8$ and inject 0 or 30 random points defects for each input image. Figure 59 shows the comparison of P_F of each input character pattern under ideal condition (noise free) and under the scenario including all process variations and signal fluctuations ($\sigma_S = \sigma_D = 0.1$).

The simulation shows that the performance degradation induced by process variations and signal fluctuations have a constant impact on all BSB circuits. When processing a perfect image under the ideal condition, no BSB circuits fails and hence $P_F = 0$. After including static and dynamic noise, P_F ranges from 1% to 7% for different input characters.

When increasing the random point defects to 30 for input images, the range of P_F increases from 0∼10% under ideal condition to 4∼16% after including all the noise sources.

In conclusion, the BSB recall circuit has high resilience to process variations and signal fluctuations since the BSB model has high tolerance to random noise. However, the correlation between two memristor cross-point arrays in the circuit will introduce directional noise into the system. As the correlation rate between two memristor cross-point arrays decreases, the performance of the BSB recall circuit becomes significantly worse. This all implies that using traditional writing operations to program memristor cross-point arrays is not a practical solution, and a training method that enables the circuit to adjust itself to adapt to the noisy environment is necessary.

6.5 SIMULATION RESULT II - TRAINING

The training method iteratively programs the memristor circuit until the required input-output function is achieved, therefore it can overcome most of the impact of process variations and signal fluctuations. In this section, simulation results are presented to demonstrate the training results for the proposed hardware training method and to compare with existing software synthesis methods. First, we compare the convergence speeds between prototype patterns and untrained patterns, essential for the realization of the "racing" BSB recall function Schultz (1993); Wu et al. (2011). Second, the performance of a memristor crossbar as an auto-associative memory is analyzed. In convergence speed analysis, we start with the simple linear memristor model and then employ the nonlinear TiO_2 memristor model based on the real device measurement Strukov et al. (2008a) to demonstrate the training effect. Finally, fabrication defects are considered by assuming that defected cells exist and are randomly distributed in the crossbar arrays. All input patterns are $(-1, +1)$ binary patterns with a length of n. The experimental setup is summarized in Figure. 53.

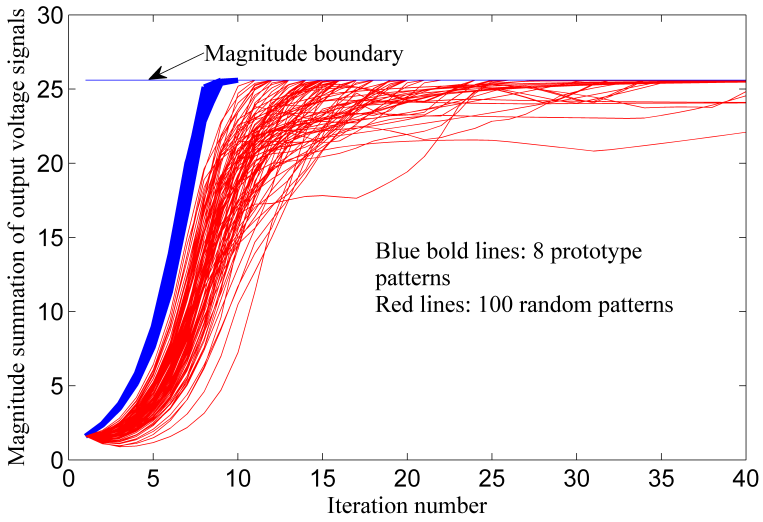

Figure 60: Experiment 1: Iteration number vs. magnitude summation of output voltage signals.

6.5.1 Convergence speed analysis

In BSB recall process, the learned prototype patterns should converge much faster than the unlearned patterns. If this phenomenon appears, then the circuit has "remembered" the prototype patterns and has the ability to classify whether an input pattern is in the set of prototype patterns or not. We conduct the following two experiments to analyze BSB circuit performance based on the convergence speeds.

Experiment 1: There are 8 different randomly generated prototype patterns, $N = 16$. The BSB recall circuit is trained to "remember" these patterns and all 8 learned prototype patterns and 100 unlearned random patterns are then recalled. The results in Figure. 60 clearly show that there is a convergence speed gap between the prototype patterns and the unlearned patterns. Specifically, the 8 prototype patterns all converge to the magnitude boundary before the 9th iteration, while the fastest convergence speed for unlearned patterns is the 12th iteration. The larger the hamming distance between the input pattern and the prototype patterns, the more iteration are required to converge, if convergence is even possible.

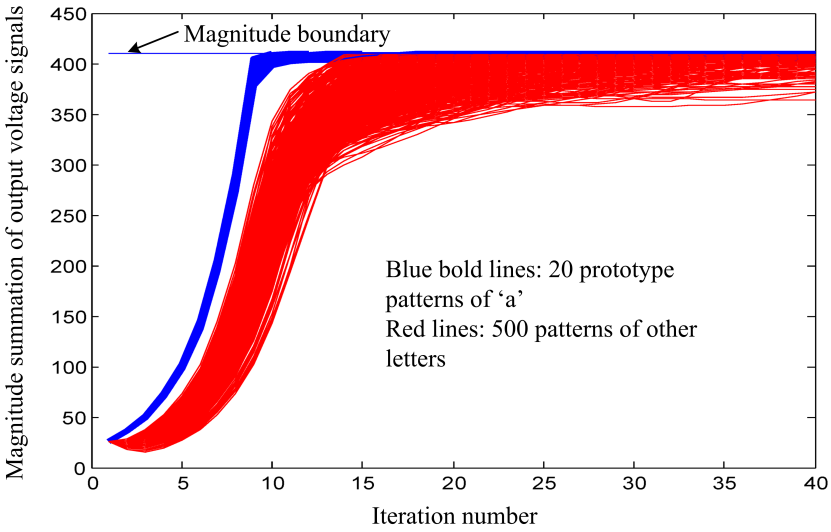

Figure 61: Experiment 2: Iteration number vs. magnitude summation of output voltage signals.

Experiment 2: The 26 lower-case characters from "a" to "z" described in Section 6.4 are used as input patterns. We use 20 patterns of lower-case character "a" representing different size-font-style combinations as the prototype patterns for training. To compare their convergence speeds, 500 patterns representing the other 25 lower-case characters with different sizes/fonts/styles are also recalled. Figure. 61 shows the result. It can be clearly observed that the 20 prototype patterns of the lower-case character "a" converge much faster than the other character patterns. Compare with result of experiment 1 in Figure. 60, as the size of BSB memory N increases (from 16 to 256), the convergence speed gap between prototype patterns and the unlearned patterns becomes more obvious.

In conclusion, a simple but effective training method is realized by circuits and it can be used to construct the hardware architecture for the "racing" BSB algorithm proposed in Wu et al. (2011).

6.5.2 BSB circuit as auto-associative memory

In this subsection we analyze the training effect from the view of an auto-associative memory. The performance criteria of auto-associative memory include the error correction rate, the uniform size of the domain of attraction and the quality of the domain of attraction Park (2010). These criteria are defined as:

Error correction rate: Error correction rate reflects the robustness margin of a perturbed BSB neural network. Given a prototype pattern $\gamma^{(k)} \in B^n (k = 1, \ldots, m)$ and an integer $l \in 0, 1, \ldots, p (\leq n)$, the probability that a $(-1, 1)$ binary input pattern at the Hamming distance of l away from $\gamma^{(k)}$ is attracted to $\gamma^{(k)}$ is denoted by $Pr(\gamma^{(k)}, l)$. The error correction rate, denoted by $ErrCorr(l)$, is the percentage for the average of the $Pr(\gamma^{(k)}, l)$ over all prototype patterns $\gamma^{(1)}, \ldots, \gamma^{(m)}$ and is defined as Park (2010):

$$Err - Corr(l) = \{\frac{\sum_{k=1}^{m} Pr(\gamma^{(k)}, l)}{m}\} \times 100. \tag{6.16}$$

Uniform size of domain of attraction: An associative memory prefers a large overall domain of attraction, indicating that every input pattern eventually converges to a prototype pattern. When optimizing the training algorithm, it requires to uniformly increasing the domain of attraction for every prototype pattern rather than focusing only on a few of them. Thus, "uniform size of domain of attraction" is a useful measurement standard for the performance of associative memory.

The number of (−1,+1) binary input patterns that are at the Hamming distance of l away from $\gamma^{(k)}$ and whose final states are $\gamma^{(k)}$ is defined as its *domain of attraction*, denoted by $Doa(\gamma^{(k)}, l)$. The uniform size of domain of attraction, denoted by $Uni - Doa(k)$, means the percentage for the $\sum_{l=0}^{p} Doa(\gamma^{(k)}, l)$ over the maximum of the $\sum_{l=0}^{p} Doa(\gamma^{(k)}, l)$ for all prototype patterns $\gamma^{(1)}, \ldots, \gamma^{(m)}$, which is defined as Park (2010):

$$Uni - Doa(l) = \{\frac{\sum_{l=0}^{p} Doa(\gamma^{(k)}, l)}{max_{1 \leq k \leq m} \sum_{l=0}^{p} Doa(\gamma^{(k)}, l)}\} \times 100. \tag{6.17}$$

Quality of domain of attraction: From testing all the possible input patterns, we can use the corresponding output patterns to evaluate the quality of domain of attraction, which reflects the overall performance of the BSB associative memory (not only for different pro-

totype patterns). As we generate random binary patterns to test, we can calculate their Hamming distance with all prototype patterns. The prototype pattern with the least hamming distance to the input pattern is regarded as the "most-likely" prototype pattern in the sense of Hamming distance. Then we can divide the $(-1, +1)$ binary input patterns into four classes based on their final state:

- Best: among the nearest prototype patterns in the sense of Hamming distance;
- Good: a prototype pattern that is not one of the nearest prototype patterns in the sense of Hamming distance, meaning it is not the "most-likely" prototype pattern.
- Negative: a spurious state (final state is none of the prototype patterns); and
- Bad: a state that is not convergent but trapped in a limit cycle.

The quality of the domain of attraction is represented by the number of $(-1, +1)$ binary input patterns in each class.

Experiment 3: We compare the training effect of our em-bedded hardware circuit with the classic BSB training algorithms - Lillo et al. (1994) and Perfetti (1995), and a more recent BSB training algorithm, Park (2010). The test case is taken from Park (2010). In this experiment, we consider the following five prototype patterns with $n = 10$:

$$\begin{aligned}
\gamma^{(1)} &= [-1, +1, -1, +1, +1, +1, -1, +1, +1, +1]^T \\
\gamma^{(2)} &= [+1, +1, -1, -1, +1, -1, +1, -1, +1, +1]^T \\
\gamma^{(3)} &= [-1, +1, +1, +1, -1, -1, +1, -1, +1, -1]^T. \\
\gamma^{(4)} &= [+1, +1, -1, +1, -1, +1, -1, +1, +1, +1]^T \\
\gamma^{(5)} &= [+1, -1, -1, -1, +1, +1, +1, -1, -1, -1]^T
\end{aligned} \tag{6.18}$$

Figure. 62,63 and 64 summarize the simulation results of the error correction rate, the uniform size of domain of attraction, and the quality of domain of attraction, respectively. The results obtained from our proposed hardware design are labeled as "Hardware".

The simulation results show that our hardware circuit per-forms better than Lillo (1994) in error correction rate and the uniform size of domain of attraction. It also achieves a much higher score in the class of "Best" in the quality of domains of attraction test. Compared to

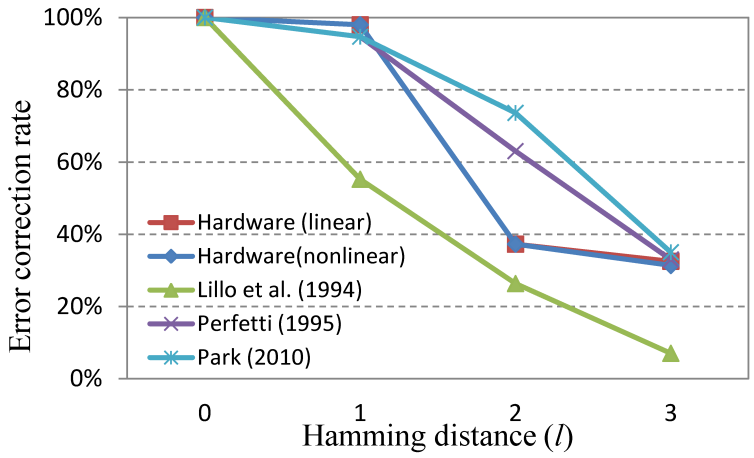

Figure 62: Experiment 3: Error correction rate.

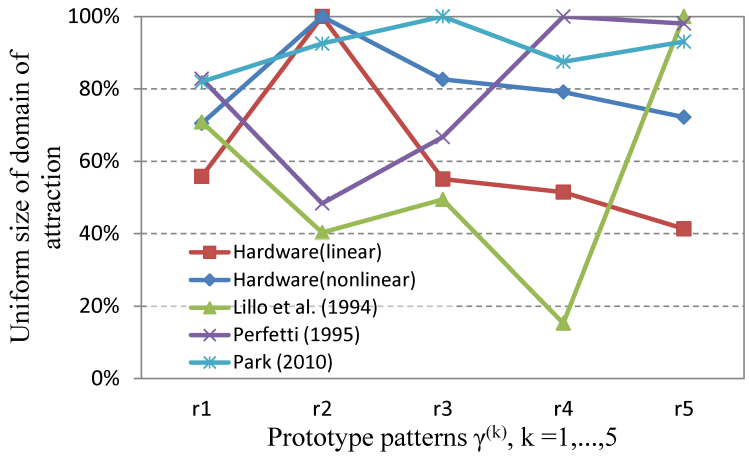

Figure 63: Experiment 3: Uniform size of domain of attraction.

	Best	Good	Negative	Bad
Hardware (linear)	419	6	465	134
Hardware (nonlinear)	465	0	473	86
Lillo (1994)	164	1	859	0
Perfetti (1995)	478	34	512	0
Park (2010)	502	10	512	0

Figure 64: Experiment 3: Quality of Domain of attraction.

Perfetti (1995), our scheme is competitive for the similar performance in uniform size and the quality of domains of attractions. However, it has a large drop in error correction rate when Hamming distance $l \geq 2$. Since our hardware circuit was built based on the fundamental training algorithm, it cannot be as good as Park (2010), the "state of the art" training algorithm developed based on constrained optimization. However, our scheme advances for its simple structure and low computation requirement. Moreover, it provides much faster training speed than the traditional software solutions since we utilize memristor crossbars embedded on-chip.

A drawback of our training circuit is the existence of "bad" states representing some input patterns that cannot be classified by the circuit. To solve this issue and further improve the training quality, we should further improve the embedded training circuit, or draw support from the external training machine with the sophisticated software training algorithm.

6.5.3 Nonlinear memristance change

To evaluate the impact of the nonlinear device characteristics, we tested the memristor-based circuit by assuming the memristance changes linearly and then applying the non-linear model in Figure. 53. The corresponding simulation results are labeled as Hardware (linear) and Hardware (nonlinear), respectively in Figure. 62, 63 and 64.

In the setup of Hardware(linear), we assume that every memristor in the crossbar has the same magnitude for the memristance changing in every training step. In other words, every memristor has the same constant learning rate. Hardware(nonlinear) is much closer to reality by including the nonlinear TiO_2 thin-film memristor model. The learning rate then depends on the current state of the memristor. Therefore, different memristors have different learning rates since they are in different states, and the learning rates of a memristor at the different training steps are different.

The simulation result is very interesting: experiments using the nonlinear memristor model demonstrate negligible performance degradation in the error correction rate, while surpassing the experiments using the linear device model in the other two criteria. This is due to the nonlinearity of memristor results in an unbalanced weight change in the training

process. Memristance movement ΔM relies on the current memristance state. The larger the memristance is, the smaller the rate of memristance change in a training step will be. Therefore, the average memristance in Hardware (nonlinear) becomes smaller compared to that of Hardware (linear). The smaller average memristance actually helps the convergence of some input patterns that cannot converge ("Bad") when using the linear device model. Therefore the scores of the last two criteria increase under the nonlinear model. In conclusion, by carefully setting the memristor training period, the nonlinear memristor model will not degrade the training quality of the proposed hardware circuit.

6.5.4 Defected cells

In nanotechnology, the existence of fabrication defects is a common and important issue. As a result, we cannot guarantee every memristor cell works properly. A small portion of the cells may not be programmable but stuck at either HRS or LRS.

We continue the case study from Sections VI.B and VI.C in this test. In this experiment, the nonlinear memristor model is used. The impact of defected cells is investigated by varying the percentage of such cells from 1% to 5%. For simplicity, the corner conditions are considered in which all the defected cells in a memristor crossbar are all at HRS or LRS. In total, there are four possible combinations of the dead cell states in $\mathbf{M_1}/\mathbf{M_2}$: LRS/LRS, LRS/HRS, HRS/LRS, and HRS/HRS. Since $\mathbf{M_1}$ and $\mathbf{M_2}$ have equivalent weights in the BSB circuit, the impact of LRS/HRS is the same as that of HRS/LRS. Hence, only three combinations need to be discussed.

In a neural network, the weights along the diagonal represent the self-connections of neurons and affect system performance more significantly M. H. Hassoun (1993). Thus, we distinguish the defected cells along the diagonal and those at the other locations in the connection matrix and analyze the impacts separately. The simulation results of the error correction rate, the uniform size of domain of attraction and the quality of domain of attraction are compared in Figure. 65, 66, and 67, respectively. Here, to better qualify and compare the quality of domain of attraction, we assign a weight to each class: "Best" = 5, "Good" = 3, "Negative" = 1, and "Bad" = 0. Figure. 67 shows the weighted summation of

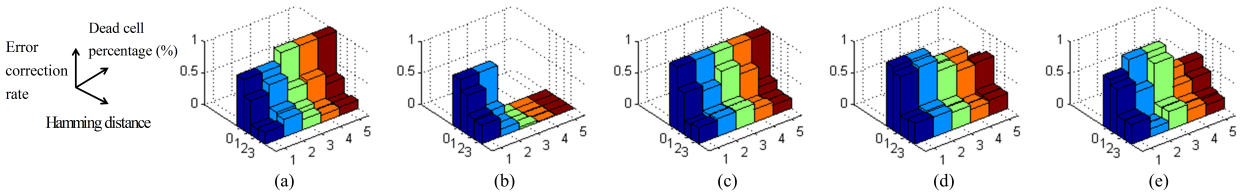

Figure 65: Error correction rate. (a) LRS/LRS, G_1=30; (b) LRS/HRS, G_1=30; (c) HRS/HRS, G_1=30; (d) HRS/HRS, G_1=50; (e) HRS/HRS, G_1=50, defected cells are only in diagonal direction.

the numbers of input patterns in the four classes as the defected cell number varies from 1% to 5% of totally 10000 memrisotrs in the crossbar array.

In common, the training speed degrades after introducing defected cells. Other memristors need to be programmed further to compensate for the errors induced by the defected cells. Hence, more training iterations have been observed.

First, we look at the defected cells not along the diagonal of the crossbars. Three corner cases LRS/LRS, LRS/HRS, and HRS/HRS are considered. The gain of the summing amplifier $G_1 = 30$. It can be observed that LRS defected cells have a more severe impact on the performance and robustness of the BSB circuit. Comparably, the combination of

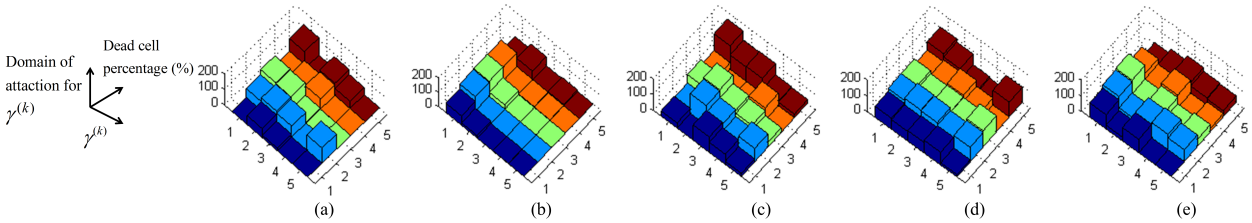

Figure 66: Uniform size of domain of attraction. (a) LRS/LRS, G_1=30; (b) LRS/HRS, G_1=30; (c) HRS/HRS, G_1=30; (d) HRS/HRS, G_1=50; (e) HRS/HRS, G_1=50, defected cells are only in diagonal direction.

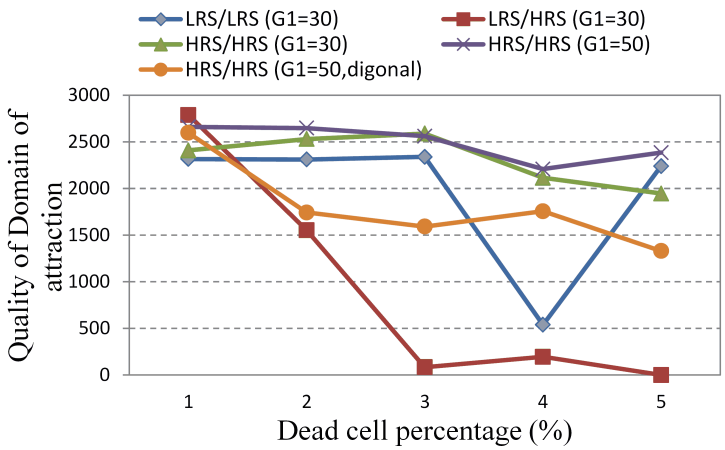

Figure 67: Quality of domain of attraction.

LRS/LRS has worse performance as compared to HRS/HRS, because the over-weight due to LRS in the connection matrix can cause more damage than the zero-weight induced by HRS. The LRS/HRS corner has the worst performance for the unbalanced combination results in the largest error in the connection matrix.

Increasing G_1 from 30 to 50 can greatly improve the performance and robustness of the memristor-based hardware design. Combined with neural network theory, this observation can be explained as follows: when defected cells exist so that the connection matrix is damaged to a certain extent, increasing the learning rate helps reduce the chaos in the training steps and decreases the possibility that the system sticks in a loop. The famous gambler tells us the less chance we have to win the game, the more stakes should be put in each play in order to increase the opportunity of winning the desired money Epstein (2012). Here is a similar situation: when some information in the con-nection matrix is lost or damaged, a larger training step should be taken to reach convergence faster.

Finally, the impact of defected cells along the diagonal of the matrix is investigated. As expected, the weights on the diagonals are very sensitive to damage and dramatically degrade system performance. For example, when a memristor on the diagonal is stuck at HRS or LRS, its generated output signal will be fed back as input of crossbar and hit the same defected

cell. Hence, only the devices on the same column provide limited compensation on the error. In contrast, the error induced by a non-diagonal defect can be compensated by memristors on other columns and hence has better chance to recover. Fortunately, post-fabrication testing can help diagnose those cells and some design techniques, such as redundancy, can be utilized to remove them from the diagonal.

6.6 SUMMARY

In this work, we first introduce a framework of hardware realization of neural network algorithms with memristor crossbar arrays. More specifically, we transform the mathematical expression of Brain-State-in-a-Box (BSB) training and recall model to pure physical device relation and design the corresponding circuit architecture. The multi-answer character recognition algorithm is used to in the experiments for robustness analysis of the proposed design. We thoroughly study the impacts of various noises induced by process variations and electrical fluctuations and discuss the physical constrains in circuit implementation. Interestingly, the correlation between the two memristor crossbar arrays within a BSB recall circuit and the resolution of summing amplifier have the most impact on the performance of the circuit, while the random noises do not have obvious correlation with the character pattern which is "trained" and stored in the BSB circuit.

Constrained by memristor array size, we implemented a small-scale BSB circuit to verify and evaluate the proposed design concept. However, the memristor-based vector matrix computing unit is not limited to simple BSB algorithm. It can also be used in realizing advanced applications supporting large dimensional and high accurate data by partitioning a complex design into small computing components. Another major concern is the unbalanced design complexity of the crossbar arrays and periphery circuitry. A possible solution is to share the periphery circuitry by several columns within a memristor crossbar array or among a few crossbars. Besides the functionality validation and design robustness in this work, we plan to evaluate the design in terms of performance, power consumption, and design cost in future work.

7.0 MEMRISTOR-BASED SPIKING NEUROMORPHIC CIRCUITS FOR TEMPORAL LEARNING

7.1 MST SYNAPSE FOR TEMPORAL LEARNING

7.1.1 STDP Learning Rule

As an improved version of Hebbian learning rule at temporal space, STDP learning rule can be taken as a causality detector. Correlation time window of a spike is used to evaluate its causality with other spikes: If a pre-spike fires before (after) a post-spike within the correlation time window, the synaptic strength of the synapse in between shall be potentiated (depressed), corresponding to a LTP (LTD) behavior.

Fig. 68 shows the realization of STDP learning by employing MST synapse under these two conditions. Here, a spike pulse is followed by a small DC signal of $0.4V$ lasting for T_{Pcorr}, representing the positive correlation time window of the spike. The negative correlation time window T_{Ncorr} can be formed by setting a normal RESET pulse $T_{Pcorr} - T_{Ncorr}$ ahead of the target spike. In this way, synapses for uncorrelated input spikes are pre-deactivated and will not be affected. Since a natural T_{Ncorr} has been naturally defined by the time between previous target spike and the current input spike in the design, a separate setting of T_{Ncorr} can be saved.

Fig. 68(a) illustrates the scenario when the input pulse injects first and the corresponding output pulse falls within T_{Pcorr} of the input pulse. Though the *strong* SET pulse remain unchanged, the small DC signal associated with the input pulse degrades the *strong* RESET pulse to a *normal* RESET. Such a condition makes the M_2 conductance increase, resulting in a LTP process that synapse conductance increases. The simulation results match well with

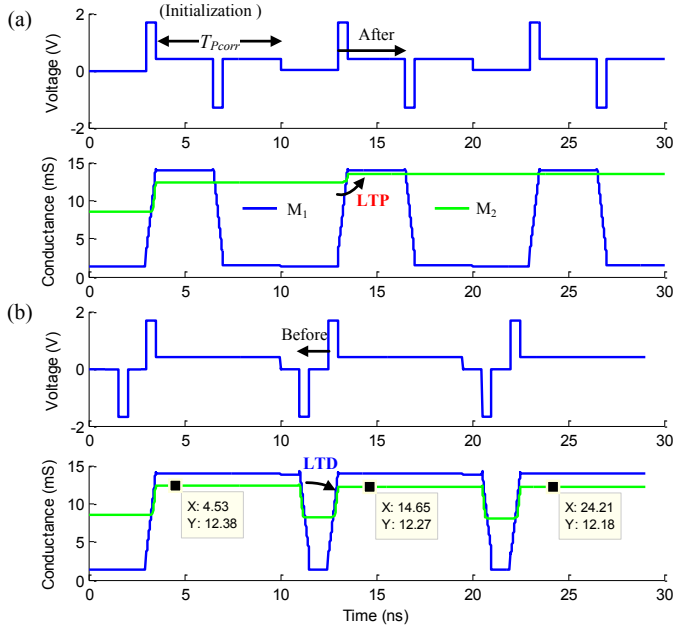

Figure 68: Employ MST synapse to realize STDP learning. (a) LTP; (b) LTD.

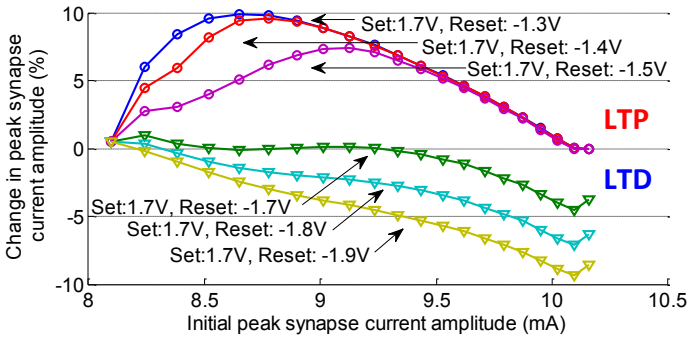

Figure 69: Characteristics of STDP learning ability of MST synapse.

our previous analysis in Section 5.3.2. The realization of LTD process is shown in Fig. 68(b), where RESET pulses remain *strong* and has larger impact than the SET pulses. And we can obviously observe the decrement of synapse conductance.

Causality decision is not the only important criteria of a synapse's STDP learning ability, which is also determined by the amplitude, time and shape of spike signals as well as the initial synaptic weight. We characterize the learning efficiency of a MST synapse and show the results in Fig. 69. The characterization is conducted by applying two recall cycles respectively before and after a STDP learning cycle (either LTP or LTD) at different SET/RESET pulse configurations. The learning efficiency is measured by the change of peak currents in the two recalls, which corresponds to the *excitatory post-synaptic current* (EPSC) used for synaptic behavior characterization in biological synapse. The results shows that in LTD the change rate of peak current slowly increases as the initial peak current amplitude grows, implying TaO_x device moves towards ON state. The convex shape of LTP curves corresponds to the slow change rate of TaO_x device near OFF state.

7.1.2 ReSuMe Learning Rule

ReSuMe learning rule is the temporal version of Delta learning rule. It has better learning quality than STDP since it considers output signal as feedback signal and minimizes the error between target signal and output signal.

We use the example in Fig. 70 to illustrate the ReSuMe learning ability of MST synapse. Note that we need to bound M_1's ON state here for the given memristor model, otherwise M_1 may enter a "more ON" state and becomes "too excited" that the weighting effect of M_2 can not be differentiated. Similar as STDP learning, a small DC signal with a period of T_corr followed a spike is used to represent its correlation time window. However, based on ReSuMe, the DC signal shall terminate when its correlated counterpart fires a spike. Thus, the convolution term $a_{di}(t) * S_i(t)$ of the original ReSuMe algorithm in Fig. 7 is represented the synaptic current excited by input pulses and the following DC signals.

The given example presents four typical situations. (1) There is no input pulse. Since target pulse is lower than SET threshold, MST synapse remains at OFF state. Neither the

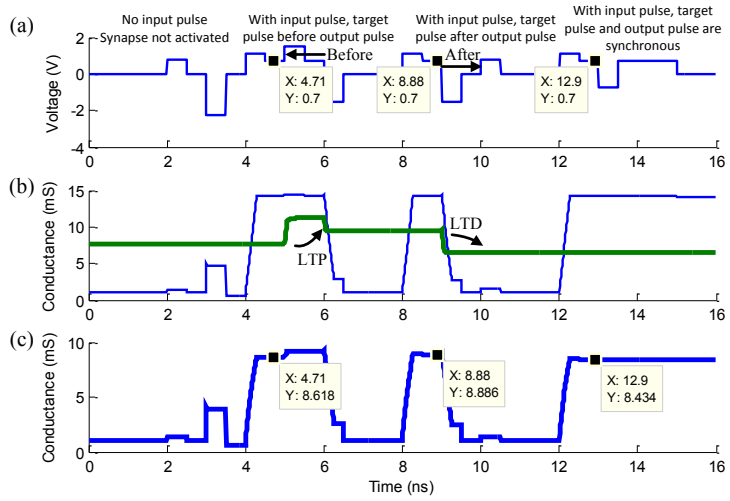

Figure 70: Employ MST synapse to realize ReSuMe learning.

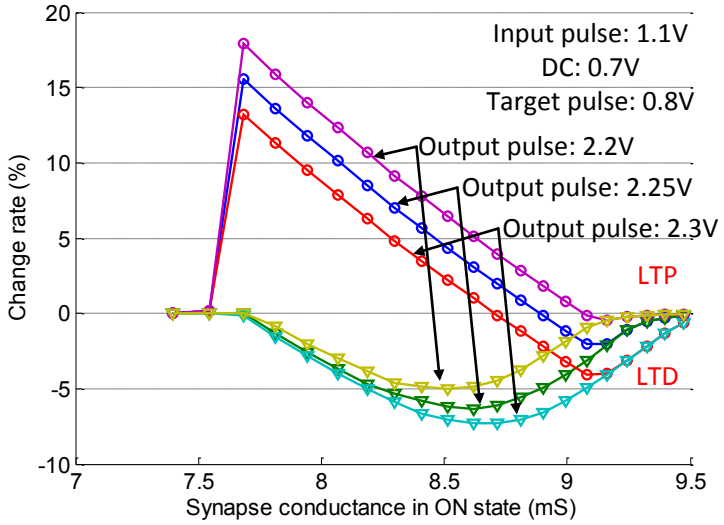

Figure 71: The MST synapse conductance change rate vs. its initial state.

target pulse nor the output pulse can change the synaptic conductance. (2) The target pulse happens before the output pulse, and both of them fall into the correlation window of the input pulse. As such, the output pulse performs as a normal RESET and cause LTP process. (3) Similar to (2) however the output pulse happens before the target pulse. The output pulse makes M_2 shift toward OFF state, implying a LTD process. Under this situation, the target pattern does not contribute to the learning process because it cannot SET the device alone while the DC signal has already been terminated. (4) The target and output pulses are approximately synchronized. There is no update on M_2, even the synapse keeps at ON and the DC signal applies. Note that the design even do not require a perfect matching of the target and output pulses, because memristor state change requires SET/RESET pulse last for sufficient time. Fig. 70 misses two situations when the target pulse or the output pulse is uncorrelated to the input pulse. Therefore, one of them does not meet the time correlation window. In this way, target pulse has no contribution or output pulse becomes a strong RESET, leads to no change or a LTD process, respectively.

Fig. 71 summarizes the ReSuMe learning characteristics of MST synapse with settings. The result is obtained by applying the similar characterization flow in Section 7.1.1 but between target pulses and output pulses. Different from STDP's characterization in Fig. 69(b), here we examine the synaptic conductance in ON state to benefit circuit design. With larger output pulses as RESET, the LTP and LTD curves shift down linearly. As a result, ReSuMe exhibits better learning characteristics than STDP – better linearity of peak current change rate and more flexible state modification.

Interestingly, both STDP's and ReSuMe's learning characteristics obtain the similar trend to the observation in biological synapses: LTP has linear decreasing trend with synaptic strength's increment, while LTD is not strongly relevant with initial synaptic strength Bi and Poo (1998). For TaO_x MST synapse, ReSuMe's LTP behavior is more linear then STDP's, while the LTD behavior is more nonlinear. This is again back to its intrinsic switching mechanisms Strachan et al. (2013).

7.1.3 "The 1^{st} Spike Dominates" property

"The 1^{st} spike dominates" is an important feature that have been observed in biological vision systems , where the first or the first few spikes carry(ies) the most information in a spike-timing train.

Let's consider a spike train from the pre-neuron to the post-neuron through a MST synapse, starting at $t = 0$. It contains N spikes that happen at t_i, $i = 1, 2, \cdots, N$. Assume a positive low DC voltage is always applied and the initial state of the synapse is set as OFF. The post-neuron has zero charge at $t = 0$ and starts accumulation afterwards such as:

$$Q(t) = \begin{cases} \int_0^t (I_{in}(t) - I_{leak}) \cdot dt & \text{before neuron fires} \\ 0 & \text{when neuron fires} \end{cases} . \tag{7.1}$$

where,

$$I_{in}(t) = \begin{cases} V_{dc} \cdot G_{\text{OFF}}(t) & \text{if } t < t_1 \\ V_{spike} \cdot G_{\text{ON}}(t) & \text{if } t = t_i,\ i = 1, 2, ..., N \\ V_{dc} \cdot G_{\text{ON}}(t) & \text{if } t > t_1 \ \&\ t \neq t_i \end{cases} . \tag{7.2}$$

Here, $I_{in}(t)$ denotes the input current at time t, V_{dc} and V_{spike} are the voltage amplitudes of the DC and the spike signals, respctively. And the ON and OFF conductances of the MST synapse are represented by $G_{\text{ON}}(t)$ and $G_{\text{OFF}}(t)$, respectively.

Eq. (7.2) shows that the first spike triggers the synapse to switch from OFF to ON. Afterwards, the charge accumulation at post-neuron becomes much faster because $G_{\text{ON}} \gg G_{\text{OFF}}$. In other words, the first spike makes a significant change on the synaptic strength, dominating the synapse feature. In addition, in the cases that V_{dc} is much lower than V_{spike} or the spikes fire frequently, the charge accumulation process is determined by the spike number N of the spike train. This also agrees with the experimental results in biological vision systems [14].

For ease of explanation, we omit the width and shape of spikes as well as the dependence of synapse conductance on the applied voltage in the formulation of Eq. (7.1) and Eq. (7.2). Though these detailed design factors do not affect the above conclusion, we have included them in the following implementation of NN applications to promise the design accuracy.

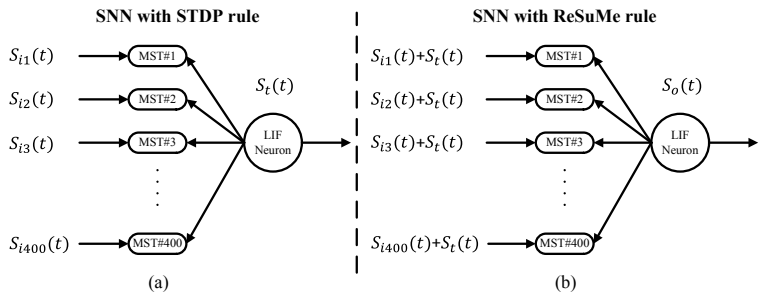

Figure 72: A spiking neural network with 400 synaptic inputs and 1 LIF output neuron and the spike configurations for (a) STDP rule or (b) ReSuMe rule.

7.2 MST SYNAPSE-BASED SPIKING NEUROMORPHIC CIRCUITS FOR TEMPORAL PATTERN RECOGNITION

7.2.1 Application Setup

Here, we demonstrate the use of MST synapses in spiking neural network by adopting a case study from Ponulak and Kasinski (2010). The selected neural network is trained on a random target firing pattern of the length of $400ns$. As shown in Fig. 72, in this neural network, 400 synaptic inputs go through MST synapses and are collected by a *leaky integrate-and-fire* (LIF) neuron. As aforementioned in Section II-B, applying the temporal encoding method can convert any spatial pattern into spatio-temporal patterns as the input spiking trains $S_{i,1\sim400}(t)$ in the given study case. If MST synapses have been properly trained, the fire pattern of the output neuron $S_o(t)$ shall converge to the target firing pattern $S_t(t)$.

To accommodate the easy timing control of MST synapses, the function of the LIF neuron can be simplified as follows: (1) The charge at the LIF neuron is leaking at a constant speed of Q_{leak}. (2) The LIF neuron fires whenever its accumulated charge exceeds the firing threshold, *i.e.*, $Q > Q_{th}$.

For ease of comparison, we adopt the same assumption in Ponulak and Kasinski (2010) that each synaptic input fires only once during the single presentation of the target signal. And the particular input pulses are distributed uniformly throughout the $400ns$ time interval. The conductances of 400 MST synapses are initialized randomly by applying a Gaussian

distribution to M_2 conductance. All the synaptic inputs are limited to excitatory because the MST synapses based on real physical devices does not provide negative weights. Table 10 summarize the signal setup in recall and learning processes.

7.2.2 Learning Temporal Patterns using STDP

First, the STDP learning rule is applied to the given application. As illustrated in Fig. 72(a), during the learning process, the LIF neuron is forced to fire at the target pattern $S_t(t)$.

Fig. 73(a) shows the progress of 60 learning cycles, each of which last $400ns$. Within each learning cycle, the target pattern $S_t(t)$ fires at the designed time illustrated by ∘ in the figure. After each learning cycle, we conduct a recall and record the output pattern $S_o(t)$ as $\times$. Our simulation results show that overall $S_o(t)$ tends to approach $S_t(t)$ but it is not always successful. This is due to the intrinsic drawback of STDP learning rule: the natural output of neuron does not participate in the learning process. So sometimes the difference between $S_t(t)$ and $S_o(t)$ cannot be minimized.

Fig. 73(b) compares the conductance of the 400 MST synapses at the initialization (blue curve) and after 60 cycles of learning (red bars). At the end of each learning process, most synapse are saturated at the tuning range of synapse conductance, implying that the constrains of physical memristor devices can severely affect the earning performance. In other words, a larger memristance range could greatly improve the STDP learning performance.

7.2.3 Learning Temporal Patterns using ReSuMe

Similarly, we examine the effectiveness of temporal pattern learning using ReSuMe learning rule Ponulak and Kasinski (2010). Different from STDP, ReSuMe includes the error minimization between the target pattern $S_t(t)$ and the output pattern $S_o(t)$. As shown in Fig. 72(b), $S_t(t)$ is combined with every synaptic input signal $S_i(t)$. The output pattern $S_o(t)$ is then fed back to every synapse to participate in the learning process. Since the memristor state change relies on both the amplitude and period of the excitation, an error margin can be naturally formed between $S_t(t)$ and $S_o(t)$: whenever the target and the output pulses overlap, the rest of their pulse duration has little impact on the memristor state.

The learning progress of ReSuMe rule of the given example is shown in Fig. 74(a). In most test cases, the output pattern generated by the LIF neuron converges to the target pattern within 25 learning cycles, demonstrating a much successful learning ability. Moreover, the pattern learning tasks can be completed without observing synapse conductance saturation as shown in Fig. 74(b). It delivers an important information: even the MST synapse design based on TaO_x devices has limited synaptic conductance tuning range, it can still provide sufficient learning ability in neural network applications with assist of appropriate learning rules. Further increase of memristance range will alleviate the situation and makes more learning rules practical.

7.2.4 Energy Estimation

To evaluate the energy efficiency of MST synapses, we calculate the energy consumption per spike in recall and learning processes by following

$$\mathrm{E_{total}} = \mathrm{E_{SET}} + \mathrm{E_{DC}} + \mathrm{E_{RESET}}, \tag{7.3}$$

where, $\mathrm{E} = \mathrm{V^2T/R_{MST-synapse}}$. An additional energy consumption related to the target pulse $\mathrm{E_{TARGET}}$ shall be added into Eq. (7.3) in ReSuMe learning. Based on the signal setup in Table 10, the recall consumes only $14.6pJ$ per spike and the energy consumption in learning process is $36.7pJ$ and $64.0pJ$ per learning spike for STDP and ReSuMe, respectively. Note that the micro model of TaO_x devices used in this work has a low resistance range of $70\Omega \sim 670\Omega$ Strachan et al. (2013). Applying nano-scale devices can significantly increase the memristance value to $100\mathrm{K}\Omega \sim 1\mathrm{M}\Omega$ Jo et al. (2010) and further reduce energy per spike to the sub-pJ region.

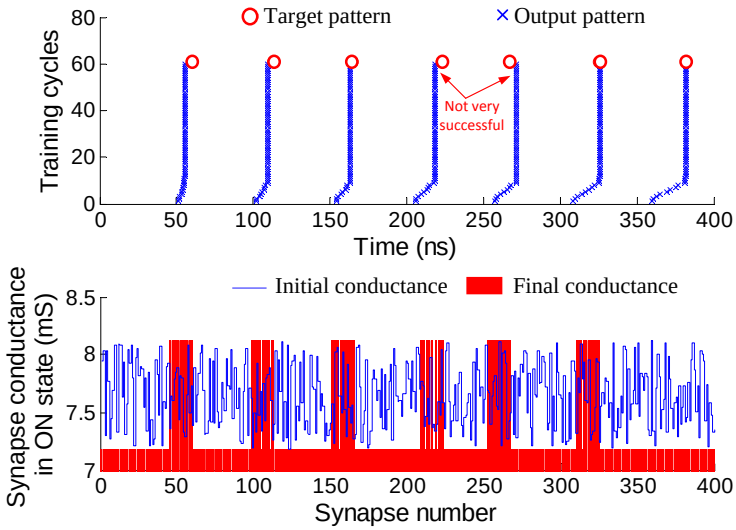

Figure 73: The effectiveness of temporal pattern learning by using STDP.

Table 10: Signal Setup for Spiking Neural Network

	V_{SET}	V_{RESET}	V_{TARGET}	V_{DC}	T_{pulse}	T_{Pcorr}
Recall	$1.1V$	$1.3V$	–	$0.2V$	$0.5ns$	$9.5ns$
STDP	$1.7V$	$1.8V$	–	$0.4V$	$0.5ns$	$9.5ns$
ReSuMe	$1.1V$	$2.2V$	$0.8V$	$0.7V$	$0.5ns$	$9.5ns$

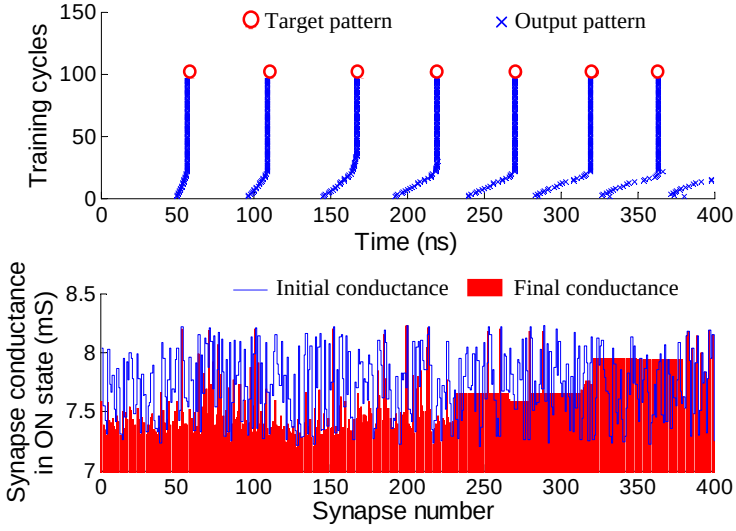

Figure 74: The effectiveness of temporal pattern learning by using ReSuMe.

8.0 CONCLUSION AND FUTURE WORKS

In this thesis, we have proposed a complete and detailed analysis for memristor-based neuromorphic circuit design from the device level to the application level. In each level, both theoretical analysis and experimental data versification are applied to ensure the completeness and accuracy of the work.

At device level, we studied different memristor models and process variations, then we carried out three independent variation models to describe the variation and stochastic behavior of TiO_2 memristors. These models can also extend to other memristor models. Meanwhile, these models are also compact enough for large-scale circuit simulation.

At circuit level, inspired by the large-scale and unique requirement of memristor-based neuromorphic circuits, we designed a circuit simulator for efficient memristor cross-point array simulations. Out simulator is $4 \sim 5$ orders of magnitude faster than tradition SPICE simulators. Both linear and nonlinear memristor cross-point arrays are studied for level-based and spike-based neuromorphic circuits, respectively.

At application level, we first designed a few compact memristor-based neuromorphic components, including "Macro cell" for efficient and high definition weight storage, memristor-based stochastic neuron and memristor-based spatio temporal synapse. We then studied three typical neural network models and their hardware realization on memristor-based neuromorphic circuits: Brain-State-in-a-Box (BSB) model stands for level-based neural network, and STDP/ReSuMe models stand for spiking neural network for temporal learning. Our result demonstrates the high resilience to variation of memritor-based circuits and ultra low power consumption.

In the future, we are going to extends our work in the following areas: first, carry out a end-to-end design flow based on our analysis for the memristor-based neuromorphic circuit

design; second, modularize the function of neuromorphic circuit for general applications; third, as memristor technology is still fast developing, we are going to improve our work as well as guide the improvement of the memristor devices.

BIBLIOGRAPHY

Hierarchical temporal memory including htm cortical learning algorithms. Technical report, Numenta, Inc., 09 2011. URL http://numenta.com/technology/#resources.

S Ambrogio, S Balatti, F Nardi, S Facchinetti, and D Ielmini. Spike-timing dependent plasticity in a transistor-selected resistive switching memory. *Nanotechnology*, 24(38): 384012, 2013.

J.A. Anderson, J.W. Silverstein, S.A. Ritz, and R.S. Jones. Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review*, 84(5):413, 1977.

A. Asenov, S. Kaya, and A.R. Brown. Intrinsic parameter fluctuations in decananometer mosfets introduced by gate line edge roughness. *Electron Devices, IEEE Transactions on*, 50(5):1254–1260, 2003.

Y. Ban, S. Sundareswaran, R. Panda, and D. Pan. Electrical impact of line-edge roughness on sub-45nm node standard cell. In *Proc. SPIE*, volume 7275, pages 727518–727518–10, 2009.

Y. B. Bazaliy and et al. Modification of the landau-lifshitz equation in the presence of a spin-polarized current in colossal- and giant-magnetoresistive materials. *Physical Review B*, 57:R3213, 1998.

Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience*, 18(24):10464–10472, 1998.

Patrick Camilleri, Massimiliano Giulioni, Vittorio Dante, D Badoni, Giacomo Indiveri, Bernd Michaelis, Jochen Braun, and Paolo Del Giudice. A neuromorphic avlsi network chip with configurable plastic synapses. In *Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on*, pages 296–301. IEEE, 2007.

Natalia Caporale and Yang Dan. Spike timing-dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

L. Chua. Memristor-the missing circuit element. *IEEE Transaction on Circuit Theory*, 18: 507–519, 1971a.

Leon Chua. Memristor-the missing circuit element. *Circuit Theory, IEEE Transactions on*, 18(5):507–519, 1971b.

Leon Chua. Resistance switching memories are memristors. *Applied Physics A*, 102(4): 765–783, 2011.

Massimiliano Di Ventra, Yuriy V Pershin, and Leon O Chua. Circuit elements with memory: memristors, memcapacitors, and meminductors. *Proceedings of the IEEE*, 97(10):1717–1724, 2009.

Richard A Epstein. *The theory of gambling and statistical logic*. Academic Press, 2012.

Siddharth Gaba, Patrick Sheridan, Jiantao Zhou, Shinhyun Choi, and Wei Lu. Stochastic Memristive Devices for Computing and Neuromorphic Applications. *Nanoscale*, 5(13): 5872–5878, 2013.

Tim Gollisch and Markus Meister. Rapid neural coding in the retina with relative spike latencies. *Science*, 319(5866):1108–1111, 2008.

Sieu D Ha and Shriram Ramanathan. Adaptive oxide electronics: A review. *Journal of Applied Physics*, 110(7):071101, 2011.

Wilfried Haensch, Edward J Nowak, Robert H Dennard, Paul M Solomon, Andres Bryant, Omer H Dokumaci, Arvind Kumar, Xinlin Wang, Jeffrey B Johnson, and Massimo V Fischetti. Silicon cmos devices beyond scaling. *IBM Journal of Research and Development*, 50(4.5):339–361, 2006.

Arne Heittmann and Tobias G Noll. Limits of writing multivalued resistances in passive nanoelectronic crossbars used in neuromorphic circuits. In *Proceedings of the great lakes symposium on VLSI*, pages 227–232. ACM, 2012.

Y. Ho, G.M. Huang, and P. Li. Nonvolatile memristor memory: device characteristics and design implications. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 485–490. ACM, 2009.

Jun Hu, Huajin Tang, Kay Chen Tan, Haizhou Li, and Luping Shi. A spike-timing-based integrated model for pattern recognition. *Neural computation*, 25(2):450–472, 2013a.

M. Hu, H. Li, Y. Chen, X. Wang, and R. Pino. Geometry variations analysis of tio_2 thin film and spintronic memristors. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011a.

M. Hu, H. Li, Y. Chen, X. Wang, and R.E. Pino. Geometry variations analysis of tio 2 thin-film and spintronic memristors. In *Asia and South Pacific Design Automation Conference*, pages 25–30. IEEE Press, 2011b.

M. Hu, H. Li, Q Wu, and G.S. Rose. Hardware realization of bsb recall function using memristor crossbar arrays. In *Design Automation Conference*, pages 498–503. ACM, 2012.

Miao Hu, Hai Li, Yiran Chen, Qing Wu, and Garrett S Rose. Bsb training scheme implementation on memristor-based circuit. In *Computational Intelligence for Security and Defense Applications (CISDA), 2013 IEEE Symposium on*, pages 80–87. IEEE, 2013b.

Miao Hu, Yu Wang, Qinru Qiu, Yiran Chen, and Hai Li. The stochastic modeling of tio2 memristor and its usage in neuromorphic system design. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 831–836, Jan 2014. doi: 10.1109/ASPDAC.2014.6742993.

ITRS. International technology roadmap for semiconductors, 2013. URL http://www.itrs.net.

Z. Jiang and et.al. Characterization of line edge roughness and line width roughness of nano-scale typical structures. In *International Conference on Nano/Micro Engineered and Molecular Systems*, pages 299–303, 2009.

Zhuangde Jiang, Fengxia Zhao, Weixuan Jing, Philip D Prewett, and Kyle Jiang. Characterization of line edge roughness and line width roughness of nano-scale typical structures. In *Nano/Micro Engineered and Molecular Systems, 2009. NEMS 2009. 4th IEEE International Conference on*, pages 299–303. IEEE, 2009.

S.H. Jo, T. Chang, I. Ebong, B.B. Bhadviya, P. Mazumder, and W. Lu. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Letter*, 10(4):1297–1301, 2010.

Justin Keat, Pamela Reinagel, R Clay Reid, and Markus Meister. Predicting every spike: a model for the responses of visual neurons. *Neuron*, 30(3):803–817, 2001.

DC Kim, S. Seo, SE Ahn, D.S. Suh, MJ Lee, B.H. Park, IK Yoo, IG Baek, H.J. Kim, EK Yim, et al. Electrical observations of filamentary conductions for the resistive memory switching in NiO films. *Applied physics letters*, 88(20):202102–202102, 2006.

K.M. Kim, B.J. Choi, Y.C. Shin, S. Choi, and C.S. Hwang. Anode-interface localized filamentary mechanism in resistive switching of TiO thin films. *Applied physics letters*, 91: 012907, 2007.

Z. Li and S. Zhang. Domain-wall dynamics driven by adiabatic spin-transfer torques. *Physical Review B*, page 024417, 2004.

Walter E Lillo, David C Miller, Stefen Hui, and Stanislaw H Zak. Synthesis of brain-state-in-a-box (bsb) based associative memories. *Neural Networks, IEEE Transactions on*, 5(5): 730–737, 1994.

X. Liu, X.-J. Liu, and M.-L. Ge. Dynamics of domain wall in a biaxial ferromagnet interacting with a spin-polarized current. *Physical Review B*, page 224419, 2005.

X. Lou, Z. Gao, D.V. Dimitrov, and M.X. Tang. Demonstration of multilevel cell spin transfer switching in MgO magnetic tunnel junctions. *Applied Physics Letters*, 93:242502, 2008.

Editor M. H. Hassoun. Associative neural memories: Theory and implementation. In *Oxford University Press*, 1993.

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

S. Matsunaga, A. Katsumata, M. Natsui, S. Fukami, T. Endoh, H. Ohno, and T. Hanyu. Fully Parallel 6T-2MTJ Nonvolatile TCAM with Single-Transistor-Based Self Match-Line Discharge Control. In *IEEE Symposium on VLSI Circuits*, pages 28–29, 2011.

Carver Mead and Mohammed Ismail. *Analog VLSI implementation of neural systems*. Springer, 1989.

Gilberto Medeiros-Ribeiro, Frederick Perner, Richard Carter, Hisham Abdalla, Matthew D Pickett, and R Stanley Williams. Lognormal switching times for titanium dioxide bipolar memristors: origin and resolution. *Nanotechnology*, 22(9):095702, 2011.

Markus Meister and Michael J Berry II. The neural code of the retina. *Neuron*, 22(3): 435–450, 1999.

Paul Merolla, John Arthur, Filipp Akopyan, Nabil Imam, Rajit Manohar, and Dharmendra S Modha. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. In *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pages 1–4. IEEE, 2011.

Feng Miao, J Joshua Yang, John Paul Strachan, Duncan Stewart, R Stanley Williams, and Chun Ning Lau. Force modulation of tunnel gaps in metal oxide memristive nanoswitches. *Applied Physics Letters*, 95(11):113503, 2009.

Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

D. Niu, Y. Chen, C. Xu, and Y. Xie. Impact of process variations on emerging memristor. In *Design Automation Conference (DAC)*, pages 877–882, 2010.

G. Malinowski O. Boulle and M. Klaui. Current-Induced Domain Wall Motion in Nanoscale Ferromagnetic Elements. *Mat. Sci. Eng. R*, 72:159–187, 2011.

Kyoung-Su Oh and Keechul Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

Phil Oldiges, Qinghuang Lin, Karen Petrillo, Martha Sanchez, Meikei Ieong, and Mike Hargrove. Modeling line edge roughness effects in sub 100 nanometer gate length devices. In *Simulation of Semiconductor Processes and Devices, 2000. SISPAD 2000. 2000 International Conference on*, pages 131–134. IEEE, 2000.

Amos R Omondi and Jagath C Rajapakse. *FPGA implementations of neural networks*. Springer, 2006.

Yonmook Park. Optimal and robust design of brain-state-in-a-box neural associative memories. *Neural Networks*, 23(2):210–218, 2010.

S. Parkin. Racetrack memory: A storage class memory based on current controlled magnetic domain wall motion. In *Device Research Conference (DRC)*, pages 3–6, 2009.

J. Partzsch and R. Schuffny. Analyzing the scaling of connectivity in neuromorphic hardware and in models of neural networks. *Neural Networks, IEEE Transactions on*, 22(6):919–935, 2011.

Renzo Perfetti. A synthesis procedure for brain-state-in-a-box neural networks. *Neural Networks, IEEE Transactions on*, 6(5):1071–1080, 1995.

Matthew D Pickett, Dmitri B Strukov, Julien L Borghetti, J Joshua Yang, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. Switching dynamics in titanium dioxide memristive devices. *Journal of Applied Physics*, 106(7):074508, 2009.

Padmanabhan Pillai and Kang G Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 89–102. ACM, 2001.

Filip Ponulak and Andrzej Kasinski. Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural Computation*, 22(2): 467–510, 2010.

Qinru Qiu, Qing Wu, and Richard Linderman. Unified perception-prediction model for context aware text recognition on a heterogeneous many-core platform. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1714–1721. IEEE, 2011.

Qinru Qiu, Qing Wu, Morgan Bishop, Robinson E Pino, and Richard W Linderman. A parallel neuromorphic text recognition system and its implementation on a heterogeneous high-performance computing cluster. *Computers, IEEE Transactions on*, 62(5):886–899, 2013.

G. Roy, A.R. Brown, F. Adamu-Lema, S. Roy, and A. Asenov. Simulation study of individual and combined sources of intrinsic parameter fluctuations in conventional nano-MOSFETs. *IEEE Transactions on Electron Devices*, 53(12):3063–3070, 2006.

Abraham Schultz. Collective recall via the brain-state-in-a-box network. *Neural Networks, IEEE Transactions on*, 4(4):580–587, 1993.

Jae-sun Seo, Bernard Brezzo, Yong Liu, Benjamin D Parker, Steven K Esser, Robert K Montoye, Bipin Rajendran, José A Tierno, Leland Chang, Dharmendra S Modha, et al. A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pages 1–4. IEEE, 2011.

Greg S Snider. Spike-timing-dependent learning in memristive nanodevices. In *Nanoscale Architectures, 2008. NANOARCH 2008. IEEE International Symposium on*, pages 85–92. IEEE, 2008.

Venkatesh Srinivasan, David W Graham, and Paul Hasler. Floating-gates transistors for precision analog circuit design: An overview. In *Circuits and Systems, 2005. 48th Midwest Symposium on*, pages 71–74. IEEE, 2005.

John Paul Strachan, Antonio C Torrezan, Feng Miao, Matthew D Pickett, J Joshua Yang, Wei Yi, Gilberto Medeiros-Ribeiro, and R Stanley Williams. State dynamics and modeling of tantalum oxide memristors. *Electron Devices, IEEE Transactions on*, 60(7):2194–2202, 2013.

D. Strukov, J. Borghetti, and S. Williams. Coupled ionic and electronic transport model of thin-film semiconductor memristive behavior. *SMALL*, 5:1058–1063, 2009.

D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453:80–83, 2008a.

D.B. Strukov and R.S. Williams. Exponential ionic drift: fast switching and low volatility of thin-film memristors. *Applied Physics A: Materials Science & Processing*, 94(3):515–519, 2009. ISSN 0947-8396.

Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The missing memristor found. *Nature*, 453(7191):80–83, 2008b.

G. Tatara and H. Kohno. Theory of current-driven domain wall motion: spin transfer versus momentum transfer. *Physical Review Letters*, 92:086601, 2004.

Andy Thomas. Memristor-based neural networks. *Journal of Physics D: Applied Physics*, 46(9):093001, 2013.

Alice Wang, Benton Highsmith Calhoun, and Anantha P Chandrakasan. *Sub-threshold design for ultra low-power systems*. Springer, 2006.

X. Wang and Y. Chen. Spintronic memristor devices and applications. In *Design, Automation & Test in Europe Conference and Exhibition (DATE)*, 2010.

X. Wang, Y. Chen, H. Xi, H. Li, and D. Dimitrov. Spintronic memristor through spin-torque-induced magnetization motion. *IEEE Electron Device Letters*, 30:294–297, 2009.

Q. Wu, M. Bishop, R. Pino, R. Linderman, and Q. Qiu. A multi-answer character recognition method and its implementation on a high-performance computing cluster. In *FUTURE COMPUTING 2011, The Third International Conference on Future Computational Technologies and Applications*, pages 7–13, 2011.

Wm A Wulf and Sally A McKee. Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

Qiangfei Xia, Warren Robinett, Michael W Cumbie, Neel Banerjee, Thomas J Cardinali, J Joshua Yang, Wei Wu, Xuema Li, William M Tong, Dmitri B Strukov, et al. Memristor-cmos hybrid integrated circuits for reconfigurable logic. *Nano letters*, 9(10):3640–3645, 2009.

J Joshua Yang, Feng Miao, Matthew D Pickett, Douglas AA Ohlberg, Duncan R Stewart, Chun Ning Lau, and R Stanley Williams. The mechanism of electroforming of metal oxide memristive switches. *Nanotechnology*, 20(21):215201, 2009.

J Joshua Yang, M-X Zhang, John Paul Strachan, Feng Miao, Matthew D Pickett, Ronald D Kelley, G Medeiros-Ribeiro, and R Stanley Williams. High switching endurance in taox memristive devices. *Applied Physics Letters*, 97(23):232102, 2010.

J Joshua Yang, M-X Zhang, Matthew D Pickett, Feng Miao, John Paul Strachan, Wen-Di Li, Wei Yi, Douglas AA Ohlberg, Byung Joon Choi, Wei Wu, et al. Engineering nonlinearity into memristors for passive crossbar applications. *Applied Physics Letters*, 100(11):113501, 2012.

J Joshua Yang, Dmitri B Strukov, and Duncan R Stewart. Memristive devices for computing. *Nature nanotechnology*, 8(1):13–24, 2013.

Wei Yi, Frederick Perner, Muhammad Shakeel Qureshi, Hisham Abdalla, Matthew D Pickett, J Joshua Yang, Min-Xian Max Zhang, Gilberto Medeiros-Ribeiro, and R Stanley Williams. Feedback write scheme for memristive switching devices. *Applied Physics A*, 102(4):973–982, 2011.

Shimeng Yu, Yi Wu, Rakesh Jeyasingh, Duygu Kuzum, and H-SP Wong. An Electronic Synapse Device Based on Metal Oxide Resistive Switching Memory for Neuromorphic Computation. *IEEE Transactions on Electron Devices*, 58(8):2729–2737, 2011.

S. Zhang and Z. Li. Roles of nonequilibrium conduction electrons on the magnetization dynamics of ferromagnets. *Physical Review Letters*, page 127204, 2004.

Victor V Zhirnov, Ralph K Cavin, James A Hutchby, and George I Bourianoff. Limits to binary logic switch scaling-a gedanken model. *Proceedings of the IEEE*, 91(11):1934–1939, 2003.