Design of Resistive Synaptic Devices and Array Architectures for Neuromorphic Computing

by

Pai-Yu Chen

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2018 by the
Graduate Supervisory Committee:

Shimeng Yu, Chair
Yu Cao
Jae-sun Seo
Chaitali Chakrabarti

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

Over the past few decades, the silicon complementary-metal-oxide-semiconductor (CMOS) technology has been greatly scaled down to achieve higher performance, density and lower power consumption. As the device dimension is approaching its fundamental physical limit, there is an increasing demand for exploration of emerging devices with distinct operating principles from conventional CMOS. In recent years, many efforts have been devoted in the research of next-generation emerging non-volatile memory (eNVM) technologies, such as resistive random access memory (RRAM) and phase change memory (PCM), to replace conventional digital memories (e.g. SRAM) for implementation of synapses in large-scale neuromorphic computing systems.

Essentially being compact and "analog", these eNVM devices in a crossbar array can compute vector-matrix multiplication in parallel, significantly speeding up the machine/deep learning algorithms. However, non-ideal eNVM device and array properties may hamper the learning accuracy. To quantify their impact, the sparse coding algorithm was used as a starting point, where the strategies to remedy the accuracy loss were proposed, and the circuit-level design trade-offs were also analyzed. At architecture level, the parallel "pseudo-crossbar" array to prevent the write disturbance issue was presented. The peripheral circuits to support various parallel array architectures were also designed. One key component is the read circuit that employs the principle of integrate-and-fire neuron model to convert the analog column current to digital output. However, the read circuit is not area-efficient, which was proposed to be replaced with a compact two-terminal oscillation neuron device that exhibits metal-insulator-transition phenomenon.

To facilitate the design exploration, a circuit-level macro simulator “NeuroSim” was developed in C++ to estimate the area, latency, energy and leakage power of various neuromorphic architectures. NeuroSim provides a wide variety of design options at the circuit/device level. NeuroSim can be used alone or as a supporting module to provide circuit-level performance estimation in neural network algorithms. A 2-layer multilayer perceptron (MLP) simulator with integration of NeuroSim was demonstrated to evaluate both the learning accuracy and circuit-level performance metrics for the online learning and offline classification, as well as to study the impact of eNVM reliability issues such as data retention and write endurance on the learning performance.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my PhD advisor Dr. Shimeng Yu for the continuous support of my PhD study, for his patience, motivation and immense knowledge. His guidance helped me in all the time of research, and I have learned a lot from him in the way of writing, communication and presentation. Besides my advisor, I would like to thank the rest of my dissertation committee: Dr. Chaitali Chakrabarti, Dr. Yu Cao and Dr. Jae-sun Seo, for their insightful comments and encouragement, but also for the sharp questions which incented me to widen my research from various perspectives.

My sincere thanks also go to Dr. Arindam Mallik, for offering me the summer internship opportunity at IMEC in Belgium and providing me direction on my research. I am also grateful of Dr. Sarma Vrudhula and Dr. Jieping Ye, who guided me working on our research projects for the first two years of my PhD study, and Dr. Dragica Vasileska, who inspired me a lot in many of her courses and gave me a strong recommendation in my scholarship application. Without all their precious supports, it would not be possible to make my research path successful.

I would like to thank my fellows in ASU for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last five years: Dr. Ligang Gao, Dr. Jiyong Woo, Runchen Fang, Rui Liu, Zhiwei Li, Xiaoyu Sun, Bin Dong, Xiaochen Peng, Panni Wang, Dr. Naveen Suda, Dr. Zihan Xu, Abinash Mohanty, Xiaocong Du, Deepak Kadetotad, Minkyu Kim, Xiaoyang Mi, Shihui Yin, Manqing Mao, Dr. Wenhao Chen, Dr. Binbin Lin and Dr. Wei-Chieh Kao. Last but not the least, I would like to thank my parents for supporting and encouraging me spiritually throughout my life.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

LIST OF TABLES

1 INTRODUCTION

Electronic devices are invented and developed to improve our life quality in many aspects such as communication, entertainment, safety and healthcare. The way we live, work and interact has been dramatically changed by the growth of modern microelectronics since its emergence. Over the past few decades, Moore's law has been the primary driving factor for the advance of computing capability by continuously scaling down the devices in size, bringing several advantages such as higher speed and lower cost and power consumption. Gordon Moore's observation was that the number of transistors in an integrated circuit (IC) doubles approximately every two years, and David House further predicted that the chip performance would double every 18 months due to more and faster transistors. Although Moore's prediction has been successful over 50 years, Today's silicon CMOS technology, however, is approaching its fundamental physical limits on the size. Moore's law has become progressively challenging and soon reached its end, meaning that the performance gain cannot solely rely on the device scaling anymore. It is necessary to discover new device technologies or new computing principles to meet the ever-increasing demand for computing capability and high performance.

1.1 Neuromorphic Computing for Artificial Intelligence

The artificial intelligence (AI) is an area of computer science that was found in 1960s and concerned with solving tasks that are easy for humans but hard for computers. Traditional problems to which AI methods are applied include handwritten recognition (e.g. MNIST dataset [1]), face recognition (e.g. Facebook's DeepFace [2]), speech recognition (e.g. Amazon's Alexa [3], Apple' Siri [4], Microsoft's Cortana [5]), robotics (e.g. Robot Operating System [6]), autonomous driving (e.g. Tartan Racing [7]), and even broad games

(e.g. Google's AlphaGo [8]) and video games (e.g. Pac-mAnt [9]). Despite a wide variety of applications, the development of AI has experienced several waves of ups and downs during the past 60 years since its advent. However, just a few years ago AI suddenly become the hottest field in technology industry. The resurgence of AI can be traced back to an annual online contest at 2012 — the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which contains over 1 million images and 1,000 object categories and is more complex than other image datasets such as MNIST [1] and CIFAR-10/100 [10]. In 2010 and 2011, the winning system could correctly recognize ~72% and ~75% of the images. In 2012, a team from the University of Toronto, achieved a nearly 10% improvement in recognition accuracy to ~85%, convincingly demonstrating the power of deep neural network (DNN). After this year, rapid improvements in the accuracy were observed. In 2015, the winning system could achieve an accuracy of ~96%, surpassing humans (~95% on average) for the first time.

The remarkable breakthrough in 2012 was widely considered to be the beginning of the deep learning revolution of the 2010s. People across the entire technology industry started to pay attention to this field. Deep learning is a class of machine learning algorithms that are based on artificial neural networks (ANNs). These networks are biologically inspired networks of artificial neurons or brain cells. In a biological brain, each connection between artificial neurons relies on the synapse, which has a strength (weight) and can transmit the signal from one neuron to another. The artificial neuron that receives the signal can process it and then signal artificial neurons that are connected to it. Fig. 1.1 shows the basic ANN structure. A simple ANN at least consists of an input and output layer of neurons, and possibly one or more hidden layers of neurons in between. The input layer is where the

input data (e.g. image samples for training) can be fed into the network. Generally, the input layer is not included when counting the number of layers in a network. After fed into the input layer, the data will travel in a forward direction through the hidden layers and finally come out at the output layer, which is called the feed forward (FF). Along the feed forward path, each neuron is responsible for performing the weighted sum of data from all the incoming synapses, and then controlling the firing of its output by an activation function.

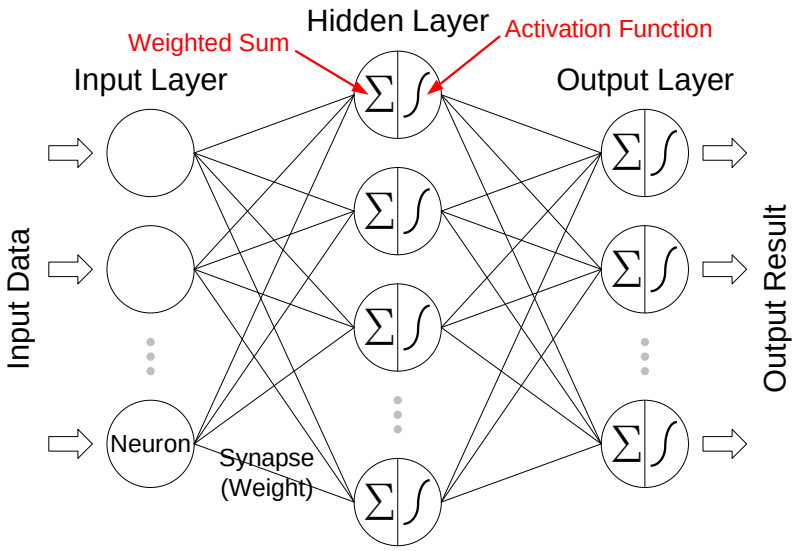


Fig. 1.1 Basic artificial neural network (ANN) structure.

The most widely used learning method is supervised learning, where correct answers (the labels) of data are provided to train the system. Examples include the image recognition, speech recognition, e-mail spam filtering, etc. Typically, the training process in the supervised learning has two phases. One is the aforementioned feed forward, and the other one is the back propagation (BP). After the feed forward, the output result will be compared with its correct answer to calculate its prediction error (the deviation). In back propagation, this error is propagated backward from the output layer to adjust the weights of each layer in a way that the prediction error is minimized. On the other hand,

unsupervised learning does not have the answers (labels) to train the network, and the goal of training is to extract the features and cluster similar examples, discovering the hidden patterns of the data. One classic example of unsupervised learning is the sparse coding algorithm, which will be introduced in Section 2.1.

In the early 1990s, some simple tasks such as recognizing handwritten digits had been achievable with ANNs, but the results became completely unsatisfactory when it was applied to a more complex task, which then requires a deeper neural network (many layers) as it is capable of building up progressively more abstract representations of the input data through each layer. Despite the immature algorithms and techniques, the major bottlenecks that hinder the development of deep learning in earlier years were the lack of training data and computing capability. Implementation of deep learning generally requires a huge amount of unstructured data to be processed for extraction of useful information through tens of layers, resulting in significant cost in time and computational resources. Today, the explosive growth of Internet has made billions of documents, images and videos available for training purposes, and the massively parallel computing power of graphical processing units (GPUs) also provides much better learning performance over weakly parallel computation with several CPUs. In 2005, the first implementation of ANN using GPU reported a threefold speedup over their CPU baseline [11]. In 2007, NVIDIA released the CUDA platform, allowing the use of a CUDA-enabled GPU for general purpose processing, which is an approach called general-purpose computing on GPU (GPGPU). Essentially making the parallel programming easier to use GPU resources, this offered an ideal platform for parallelizing a neural network in GPU and it was then rapidly adopted by deep

learning researchers. In 2009, it had been reported that training a DNN with GPUs could be 70 times faster than that with CPUs [12].

However, either CPU or GPU is a general-purpose computing platform based on the sequential von Neumann architecture, which involves clear separation of the computing unit and memory between a data bus path. Due to the requirement of high bandwidth and power consumption for data communication via this data bus, traditional von Neumann architecture is inadequate for implementing modern DNN architectures, which requires intensive multiply-and-accumulate (MAC) operations with millions of parameters on millions of data in the feed forward computation. This challenge in memory access is recognized as the "von Neumann bottleneck" that degrades the overall efficiency and performance of the system [13]. To fundamentally overcome this problem, the neuromorphic computing has emerged in recent years as an attractive alternative to these conventional computing architectures based on von Neumann systems. The term "neuromorphic computing" is firstly coined by Carver Mead in 1990 [14]. It is a new computing paradigm inspired by the cognitive functionality of brain. Unlike the conventional CPUs/GPUs, a biological brain (e.g. mammalian brain) enables parallel processing of a massive amount of information in a small area with high efficiency and low power consumption. Therefore, the ultimate goal of neuromorphic computing is to develop neuromorphic hardware accelerators that emulate highly efficient processing of biological information to bridge this efficiency gap between the network and real brain [15], which is believed to be the major driving force of the next AI revolution.

Generally, there are two design principles for the neuromorphic hardware accelerators. One principle is inspired from the neuroscience that the hardware system emulates the brain

functionality based on biological learning rules, such as the spike-timing-dependent plasticity (STDP). In this design, the information is typically encoded using spikes to improve the area cost and energy efficiency [16], and thus the network is generally referred to as the spiking neural network (SNN). Well-known examples include University of Manchester's SpiNNaker [17], Qualcomm's Zeroth [18], Stanford's Neurogrid [19], IBM's TrueNorth [20], etc. The other design principle is based on ANN with machine learning algorithms, aiming at accelerating MAC operations while minimizing the energy cost of data movement in hardware. As DNN is currently the hottest topic in the AI field, more research and development has been focused on the digital implementation of DNN, either with the field-programmable gate array (FPGA) or application-specific integrated circuit (ASIC). Due to their reprogrammability, FPGAs offers higher flexibility, lower development cost and shorter design time than ASICs. The most notable example with FPGA is probably the one started as CNP [21], which was further improved and renamed to NeuFlow [22] and later on to nn-X [23]. These designs could achieve 10's to 100's giga operations per second (GOPS) with only <10 W power. On the other hand, ASIC implementation could give higher performance than FPGA in terms of area, speed and power. For a 90 nm technology, it has been reported that the ASIC implementations are 5× faster, 14× higher in power and 35× smaller in area [24]. One classic example is a series of ASIC designs called the "DianNao" family (DianNao [25], DaDianNao [26], PuDianNao [27], ShiDianNao [28]), where the impact of memory buffer design on the performance and energy was specifically emphasized. Another notable example is Google's recently developed tensor processing unit (TPU) [29], which has a performance of 180 tera floating point operations per second (TFLOPS) with 4 chips on the board.

Although there are some design tradeoffs between FPGA and ASIC as mentioned above, these platforms can generally achieve comparable or better performance with low power consumption compared to GPU acceleration. However, both two still have limited computing resources, memory and I/O bandwidths, thus it is impractical to implement a DNN on a chip with similar complexity as a biological brain that has ~10^{13} synapses, because the current silicon CMOS technology is not adequate to provide sufficient on-chip memory resources. Even in ASIC, the smallest unit for the synaptic weight storage is to use the static random access memory (SRAM), which consists of 6 to 8 transistors and requires a cell size of $100F^2$~$200F^2$ in total (F is the lithography feature size). To represent the precision of a single weight value, multiple SRAM cells are further needed to form a synapse, making it even more area-inefficient.

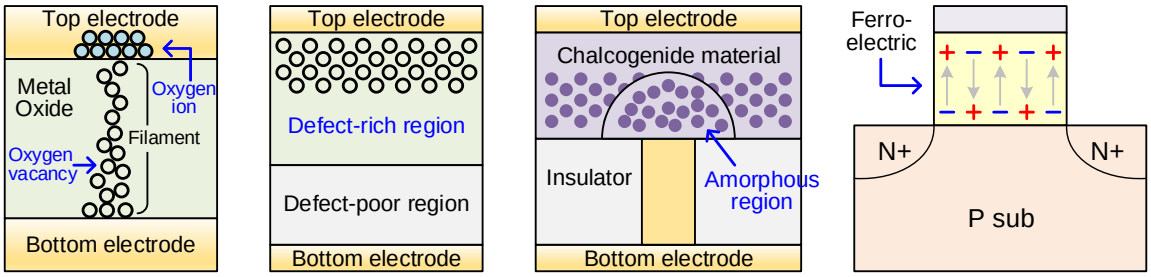
1.2 Emerging Non-volatile Memory for Synaptic Devices

It came to be realized that the "von Neumann bottleneck" problem in modern DNNs cannot be fully addressed by the aforementioned acceleration platforms alone, especially it is expected that the future DNNs will grow rapidly in network depth, model size and computational complexity. In a sense that the number of synapses is far more than the number of neurons (with a complexity of $O(N^2)$ and $O(N)$, respectively), it is very crucial to explore more compact synaptic devices at nanoscale level beyond the traditional silicon CMOS technology, and exploit the analog properties of these synaptic devices rather than use them as binary or multi-level on-chip storage memories. In this way, it can prevent the tremendous hardware cost with CMOS based synapses, meanwhile potentially reducing the computation complexity from $O(N^2)$ to $O(1)$ for a fully parallel operation.

The current progress in nanotechnology is paving the way toward implementation of compact synaptic devices using low cost and ultra-high density memory array [30]. In fact, due to its maturity, the floating-gate memory technology has been successfully implemented on a single chip as synapses for the neuromorphic computing [31]. To achieve even higher integration density, faster speed and lower programming voltage, compact synaptic devices based on emerging non-volatile memories (eNVM) are proposed for the neuromorphic systems, including resistive random access memory (RRAM) [32-38] and phase change memory (PCM) [39-41], etc. These eNVM devices are non-volatile and two-terminal with cell size $4F^2$~$12F^2$, and they use their conductance to represent the stored synaptic weight. These eNVMs have long been considered as promising candidates for future replacement of NAND/NOR FLASH in storage applications. Although they are still not mature yet, some prototype chips have been demonstrated. For example, Samsung has reported an 8-Gb PCM prototype chip at 20 nm that has a write bandwidth of 40 MB/s in 2012 [42], Micron/Sony has reported a 16Gb RRAM prototype chip at 27 nm that could achieve a read/write bandwidth of 1GB/200 MB per sec in 2014 [43], etc. However, the requirement for eNVM to be used as synaptic devices is more stringent. This is because they also need to have the desired "analog" multi-level conductance states to represent the weight and perform accurate computation in neuromorphic applications. Therefore, there is an even larger design space with eNVM being exploited as a synaptic device for efficient hardware implementations of DNNs.

Operation of eNVM devices is quite simple. The transition between conductance states in eNVM devices is triggered by electrical inputs (voltage or current pulse). Generally, the conductance is increased and decreased with positive and negative programming voltage

pulses, which is referred to as SET and RESET or weight increase and decrease, respectively. The detailed conductance switching mechanism is different for different types of eNVM devices. For RRAM devices, they can be either filamentary or non-filamentary (homogeneous interface) depending on the switching mechanisms. The device structure of filamentary RRAM is shown in Fig. 1.2(a). Operation of filamentary RRAM devices relies on the voltage-driven conductance switching of the metal oxides (e.g. HfO_x [44], AlO_x [45], WO_x [46], TaO_x [47] and TiO_x [48]), which is generally attributed with the formation and rupture of the conductive filament that consists of defects (i.e. oxygen vacancies in oxide). Filamentary switching process is fast and low-power, and many of these aforementioned materials are highly silicon CMOS fabrication process compatible, making filamentary RRAM a promising candidate as embedded non-volatile memory technology. However, its filament formation (SET) process is inherently abrupt, making weight increase uncontrollable. The common solution in traditional memory application is to apply compliance current through external circuitry to limit the filament growth, but this is not feasible for analog conductance tuning with multilevel compliance current in neuromorphic computing as the peripheral circuit design will become much more complex. Thus, the use of RESET-only weight tuning was a more realistic approach [37]. Today, making gradual SET is still an active research topic for filamentary RRAMs. Some bilayer oxides structures have been proposed to prevent forming a single strong filament through the tunneling gap distance as the primary variable to determine the conductance based on an exponential relationship. They either allow the filament to grow in lateral size [36] or form multiple weak filaments [49] to achieve a more gradual SET. However, they may have larger variability due to stochastic filament formation.



(a) Filamentary RRAM (b) Non-filamentary RRAM (c) Phase Change Memory (d) Ferroelectric FET

Fig. 1.2 Schematic of (a)-(c) two-terminal emerging non-volatile memory (eNVM) and (d) FeFET based synaptic device structures.

The device structure of non-filamentary RRAM is shown in Fig. 1.2(b). Different than the filamentary RRAM, conductance switching in non-filamentary RRAM is attributed to the field-induced change of the Schottky barrier or tunneling barrier at the interface homogeneously over the entire device area [50], enabling gradual conductance tuning that is highly suitable for implementation of analog synapse. Several RRAM synaptic devices were reported on this type, such as Ag:a-Si [32], TaO_x/TiO_2 [33, 34], PCMO [35, 51]. Despite more gradual change in the conductance, the conductance tuning curve is quite nonlinear and asymmetric due to nonlinear barrier change by defect movement, thus new programming schemes such as non-identical pulse schemes that mitigate this issue may be required. In addition, some of these non-filamentary RRAMs indeed suffer from the so-called "voltage-time dilemma" [50], which describes the conflict between the fast switching speed and long retention time (requiring low and high energy defect diffusion barrier, respectively). Hence it can be found that the conductance tuning speed of some reported non-filamentary RRAMs are much slower than the filamentary ones [34, 35].

On the other hand, PCM relies on the chalcogenide materials (e.g. $Ge_2Sb_2Te_5$ [52]) to switch between the crystalline phase (high conductance state) and the amorphous phase

(low conductance state). Its device structure is shown in Fig. 1.2(c). PCM also has the same asymmetric conductance switching problem as the filamentary RRAM. Its SET process can be made gradual with repetitive pulses, while its RESET process is abrupt because the thermal melting and quench is required for crystalline to amorphous phase transition. To address this issue, one of the architectural approach is to use two PCM devices as one synapse [39, 41], where the gradual SET process is utilized in both devices, and the differential read-out conductance is used to represent the weight increase and decrease. Another approach is to use the non-identical pulse schemes that can be applied to general eNVMs. As demonstrated in [40], the increasing pulse amplitude scheme helps alleviate the conductance overshoot in the beginning and saturation in later stages. However, the non-identical pulse scheme will make the peripheral circuit design complex, which will be emphasized later in Section 2.3.1.

There is also another type of synaptic devices based on ferroelectric field-effect-transistor (FeFET) [53, 54]. FeFET based synapses are three-terminal like a conventional transistor, but with its gate dielectric replaced by a ferroelectric material that has multiple domains of polarization. Its device structure is shown in Fig. 1.2(d). With programming voltage pulses applied on the gate, part of the polarization direction can be changed, enabling gradual tuning of the threshold voltage and thereby the channel conductance to store the analog weights. Unfortunately, non-identical pulse schemes still cannot be avoided to achieve a linear conductance tuning [53, 54].

No matter which type the synaptic device is based on, it has to exhibit desired synaptic device behaviors in order to achieve the expected learning performance. The ideal synaptic device behavior assumes that identical programming voltage pulses can tune the weight

linearly. As shown in Fig. 1.3, however the realistic devices reported in literature do not follow such ideal trajectory, exhibiting "non-ideal" properties such as nonlinear and noisy weight increase/decrease, limited precision and finite ON/OFF ratio. For example, Ag:a-Si devices [32] show a nonlinear and noisy weight increase/decrease; though TaO_x/TiO_2 devices have been improved to exhibit a more linear and smooth weight increase/decrease [34], the ON/OFF ratio is very limited (~2). Such non-ideal behaviors commonly exist in today's synaptic devices [32-36], possibly due to the inherent drift and diffusion dynamics of the ions/vacancies in these materials. Detailed analysis of their impact on the performance of different learning algorithms will be performed in Section 2.3 and 5.3.

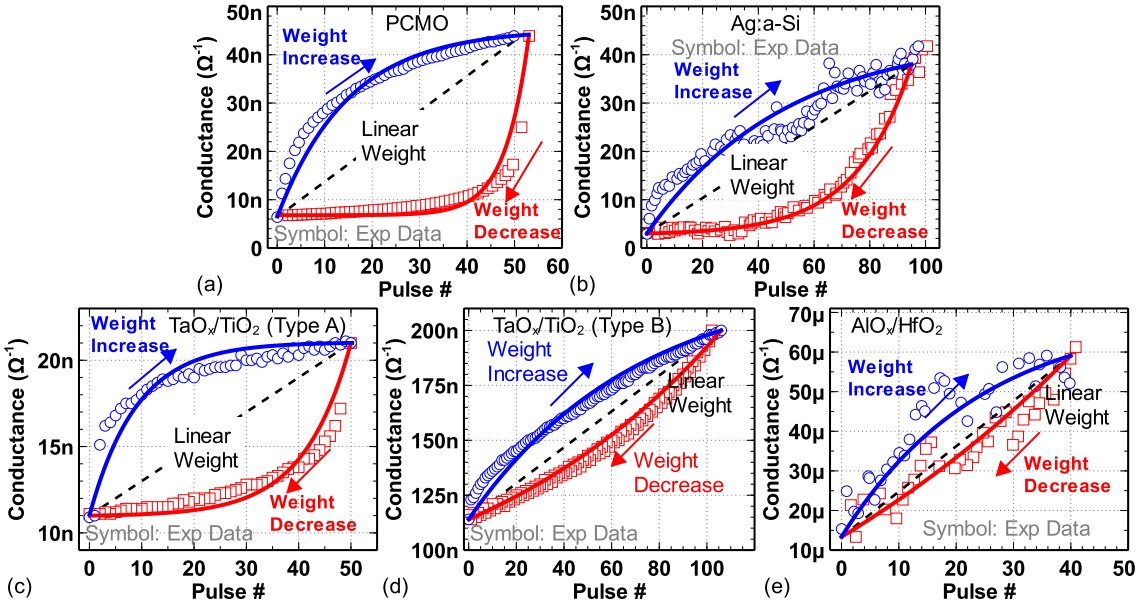


Fig. 1.3 Reported measured experimental data of weight update in (a) PCMO [35], (b) Ag:a-Si [32], (c) TaO_x/TiO_2 (Type A) [33], (d) TaO_x/TiO_2 (Type B) [34] and (e) AlO_x/HfO_2 [36] based synaptic devices. © 2017 IEEE.

1.3 Synaptic Crossbar Array Architecture

A set of synapses that fully connects between two layers of neurons can be viewed as a weight matrix. The most compact and simplest array structure for synaptic devices to

form this weight matrix is the crossbar array structure, where each synaptic device is located at each cross point. The crossbar array structure can achieve a high integration density of $4F^2$/cell. As shown in Fig. 1.4(a), if the input vector is encoded by read voltage signals, the weighted sum operation (vector-matrix multiplication) can be performed in a parallel fashion with synaptic crossbar array [55, 56]. The weighted sum result in terms of the output currents are then obtained at the end of each column. Ideally, it can be expressed in a matrix form:

$$\begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} = \begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nm} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_m \end{pmatrix} \tag{1.1}$$

where each G element in the weight matrix is the conductance of the synaptic devices. Fig. 1.4(b) shows the equivalent RC model of a single cell in the crossbar array structure, which can be duplicated to form the whole array. The wire parasitics (R_w and C_w) not only bring extra latency and energy consumption in the array, but also causes IR drop (reduction of access voltage) along the weighted sum path. Aggressive downscaling of the wire width (W) will make the IR drop more severe. Hence the weighted sum current in Eq. (1.1) may not be accurately obtained. Fortunately, the sneak path problem [57] of the unselected cells in the array for conventional memory application does not exist in the weighted sum operation, if all the cells are participating in the computation. It is preferred that the value of input vector element is encoded by the number of identical voltage pulses, which causes less distortion on the weighted sum compared to the analog encoding scheme with varying voltage amplitude [56]. In the analog encoding scheme, it is also difficult to split the read voltage (typically <1 V) into multi-levels due to noise consideration and practical bias circuit design constraints.

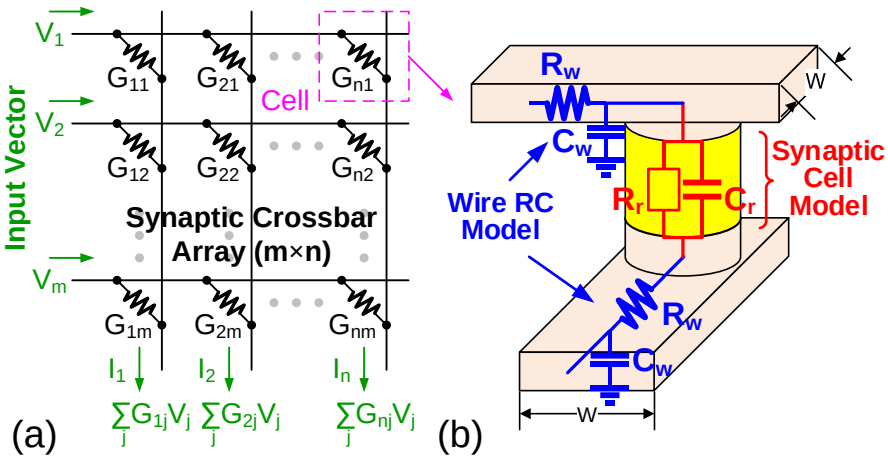


Fig. 1.4 (a) Weighted sum operation in an eNVM based synaptic crossbar array structure. (b) Equivalent RC model of a synaptic device. © 2018 IEEE.

For weight update operation, though a fully parallel write scheme has been proposed [34], programming all the cells in the array may consume too much peak power that the peripheral circuits can provide, and it also requires complex peripheral circuit design thus the hardware cost will be tremendous. Therefore, row-by-row write scheme has to be used. The voltage bias for the row-by-row write scheme in weight update is shown in Fig. 1.5. As the weight increase and decrease need different programming voltage polarities, the weight update process requires 2 steps with different voltage bias schemes. As there is no isolation between cells, it is necessary to apply an intermediate voltage (V/2) at all the unselected rows and columns to prevent the write disturbance on unselected cells during weight update [58]. In this scheme, a lot of energy is consumed to charge up all unselected rows and columns for every single operation. Therefore, the simple crossbar array architecture is not energy-efficient in weight update. In Section 3.1, there will be discussions on the design for improvement of synaptic array architectures that show better performance on weighted sum or weight update operations.

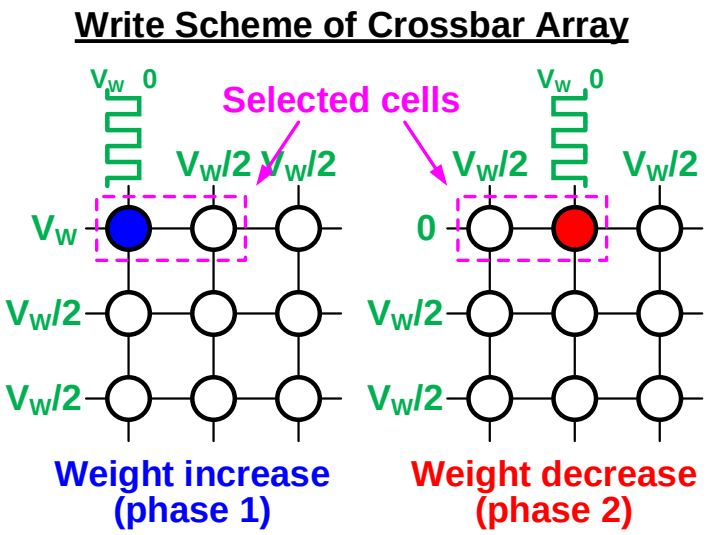


Fig. 1.5 Voltage bias scheme in the write operation of crossbar array. Two separate phases for weight increase and decrease are required. In this example, the left cell of the selected cells will be updated in phase 1, while the right one will be updated in phase 2.

1.4 Overview of Contributions

This dissertation aims at addressing the aforementioned design challenges of neuromorphic computing system based on synaptic devices and array architectures. Firstly, the sparse coding algorithm is used as a starting point. With the assumption of perfect linear weight update on analog eNVM devices, the design methodologies for co-optimizing the synaptic crossbar array with the sparse coding algorithm to implement on-chip learning. By applying a set of reverse scaling rules, the output function error can be minimized at an affordable expense of area, energy and latency [56]. As the output function error may not accurately represent the real learning accuracy, the synaptic device behavioral model is directly incorporated into the weight update operation in the algorithm. The impact of non-ideal synaptic device properties on the learning accuracy of the sparse coding algorithm can then be thoroughly analyzed, and mitigation strategies to remedy the accuracy loss are proposed [59]. The discussions will be presented in Chapter 2.

Although the crossbar array architecture is simple, it suffers from the write disturbance issue and large weight update energy. To reduce the energy consumption in weight update, it is proposed to add a selector device in series with the eNVM to achieve nonlinear I-V characteristics. With selector, the IR drop along interconnects can also be alleviated due to higher cell resistance [58]. Another way to further reduce the energy is to modify the existing conventional 1-transistor-1-resister (1T1R) array to the "pseudo-crossbar" array by rotating the bit lines by 90^o to enable weighted sum operation [60]. In the neuromorphic hardware system, the peripheral circuits play important roles in supporting the synaptic arrays, where one of the key component is the neuron analog-to-digital converter (ADC) that converts weighted sum currents to digital outputs. Following the principle of the integrate-and-fire neuron model, the read circuit is designed to integrate the weighted sum current on the array column capacitance and fire output spikes once the voltage charges up above a certain threshold [61]. However, the read circuit is complex and not area-efficient. More aggressively, a novel design was proposed to replace the entire read circuit with a compact two-terminal device that exhibits metal-insulator-transition (MIT) phenomenon, where its voltage oscillation is utilized as an integrate-and-fire neuron's output waveform [62]. The results will be presented in Chapter 3.

To facilitate the design space exploration of on-chip learning, a circuit-level macro simulator "NeuroSim" was developed in C++ to estimate the circuit-level performance (such as the area, latency, energy consumption and leakage power) of neuromorphic hardware accelerators with memory array based architectures [63]. Dedicated to support neuromorphic hardware accelerators, the hierarchy of the simulator consists of different levels of abstraction from the memory cell and transistor technology parameters, to the

gate-level sub-circuit modules and then to the SRAM and eNVM array architecture including the peripheral circuits. As a case study, the simulator has been used to evaluate the performance of partitioning a large weight matrix into SRAM and eNVM accelerators [64]. The details will be described in Chapter 4.

One of the powerful features of NeuroSim is its ability to support neural network learning algorithms by providing the circuit-level performance estimation, forming an integrated device-to-algorithm simulation framework. To demonstrate this feature, a 2-layer multilayer perceptron (MLP) simulator with integration of NeuroSim was constructed, which is useful for investigating the impact of analog eNVM's non-ideal device properties and benchmarking the design trade-offs (both the learning accuracy and circuit-level performance metrics) between SRAM, digital and analog eNVM based architectures for online learning and offline classification [65, 66]. Besides the learning performance in normal operations, the simulator was also used to study the impact of eNVM reliability issues such as the data retention failure in offline classification and the write endurance degradation in online learning [67]. The results will be presented in Chapter 5.

2 SYNAPTIC CROSSBAR ARRAY DESIGN FOR ON-CHIP SPARSE DICTIONARY LEARNING

The resistive crossbar array architecture has been proposed for on-chip implementation of weighted sum and weight update operations in neuromorphic learning algorithms. However, several limiting factors potentially hamper the learning accuracy, including the nonlinearity and device variations in weight update, and the read noise, limited max/min conductance ratio (ON/OFF ratio) and array parasitics in weighted sum. With unsupervised sparse coding as a case study algorithm, this chapter will employ device-algorithm co-design methodologies to quantify and mitigate the impact of these non-ideal properties on the accuracy.

2.1 Sparse Coding Algorithm

Sparse coding (SC) algorithm [68] is selected as a starting point for on-chip implementation with synaptic devices due to its simplicity. Despite of a simple network with two layers of neurons and one weight synaptic matrix, it can still achieve reasonably high learning accuracy with invariance for pattern's spatial shift and rotation. The sparse coding is found to be a bio-physiological plausible model: neurons in mammalian primary visual cortex can form a sparse representation of natural scenes [69, 70], which is believed to emerge from an unsupervised learning algorithm that attempts to find a factorial code of independent features such as lines, edges and corners. For real-world applications, the sparse coding algorithm has demonstrated its power in many domains such as audio processing, text mining and image recognition. In this work, the goal is to evaluate and optimize the synaptic device properties and crossbar architecture for fast and compact on-chip sparse feature learning as a case study.

Fig. 2.1(a) shows the simplified process flow of the sparse coding algorithm (SC module), which is obtained from [71] with optimization on the algorithm parameters. In the training phase, with a given input vector set $\{ X \}$ (braces mean a collection of objects), the corresponding feature vector set $\{ Z \}$ and the dictionary matrix (D) are trained iteratively by minimizing the objective error function (E):

$$E = \min \sum (\|DZ - X\|^2 + \lambda \|Z\|_1) \tag{2.1}$$

As each X is a sparse linear combination of Z via D, the first term of Eq. (2.1) generally measures how well the dictionary reconstructs the input data. The second term of Eq. (2.1) imposes constraint of the sparsity of the feature vector. Since both D and Z are unknown, the above optimization problem is a non-convex problem. It is proposed to alternatively optimize Z with fixed D by the coordinate descent (CD) method and optimize D with fixed Z by the stochastic gradient descent (SGD) method, which converts the problem into a convex optimization problem. Compared to conventional full gradient descent, SGD is more computation-efficient with large-scale dataset [71]. Using SGD, the D weight update process can be expressed as:

$$D \leftarrow D - \eta R Z^T \quad (R = DZ - X) \tag{2.2}$$

It can be seen that D is modulated by the product of $\eta R Z^T$, where R is the reconstruction error, and η is the learning rate, which is essentially the delta rule. For the ideal software implementation of the algorithm, the exact value of $\eta R Z^T$ can be calculated and applied to the update of D. However, the D update implemented on-chip needs to be translated to the

number of pulses applied on the synaptic devices, and the effect of the programming pulses on the conductance of the devices may not represent the exact value of ηRZ^T due to the realistic properties of synaptic devices as mentioned above. In this work, we model the weight update curve and incorporate this model in the D update code in the SC algorithm.

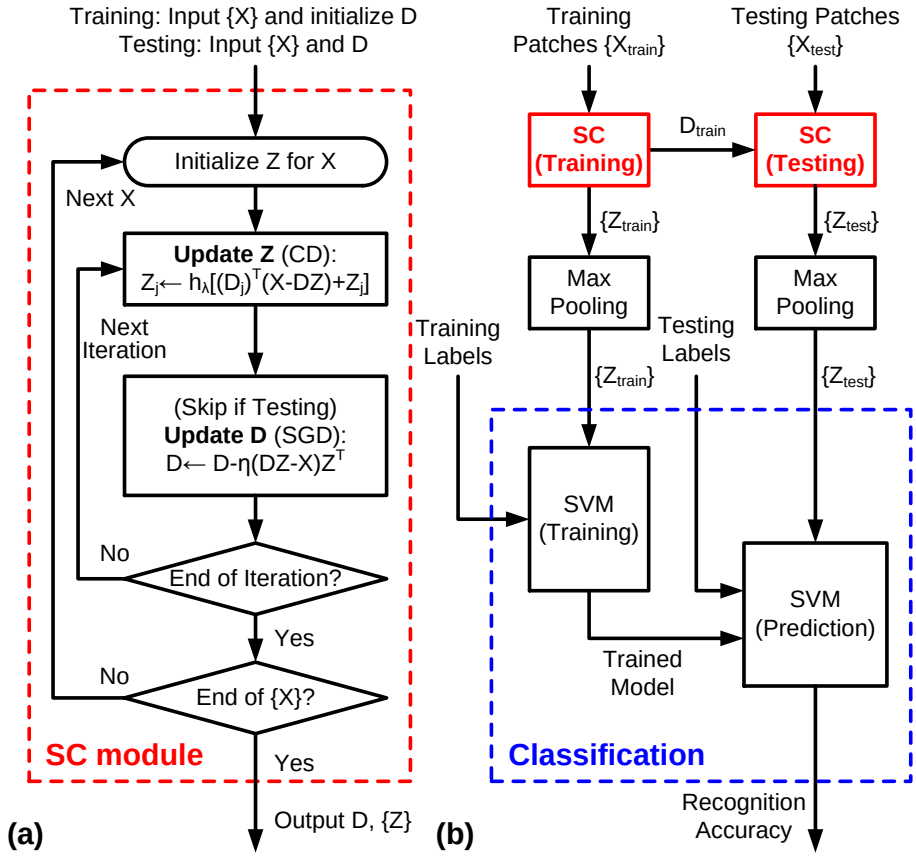


Fig. 2.1 Process flow of (a) the sparse coding (SC) module and (b) the entire process including the unsupervised feature extraction by sparse coding (SC) and the supervised classification by support-vector-machine (SVM). © 2016 IEEE.

Fig. 2.1(b) describes the entire process flow that includes dictionary learning (training phase) and classification (testing phase). In this work, the MNIST handwritten digits [1] are used as the training and testing data set, where the raw images are densely sampled into small patches with 10×10 pixels as X input vector with a dimension of 100, as shown in

Fig. 2.2. In the later analyses, a set of 40k images is used for training and a different set of 5k images is used for testing, as we have found that using the entire 60k training images does not have noticeable increase on the accuracy (only 1%) and its simulation will be much slower.

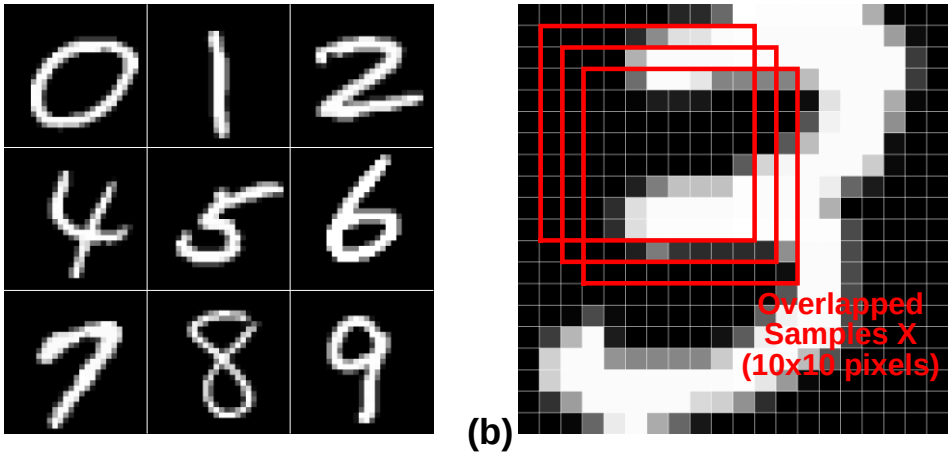


Fig. 2.2 (a) Examples of handwritten MNIST data [1]. (b) Image patches (10×10 pixels) are extracted for training. © 2016 IEEE.

Fig. 2.3 shows the learning accuracy as a function of Z vector dimension. The learning accuracy does not increase much beyond a dimension of 200. In this work, we fix the Z dimension to be 300, thus the size of the D matrix is 100×300 (X×Z). After the training process, the trained dictionary D_{train} is used as a fixed D in the testing phase to generate the testing features $\{ Z_{test} \}$. Before the classification process, a simple maximum pooling operation is employed on both the trained and testing features for each image to select the most active neuron of each feature node:

$$Z_i = \max\left(Z_i^1, \ldots, Z_i^k\right) \tag{2.3}$$

where Z_i^1, …, Z_i^k are the *i*th elements of the feature vectors of total k small image patches per image. The maximum pooling merges all the feature vectors of small image patches into one feature vector per image by selecting the maximum value of each *i*th element. Finally, to classify the 10 digits, the support vector machine (SVM) [72] is used. With the input of testing labels, SVM performs classification and gives out the recognition accuracy.

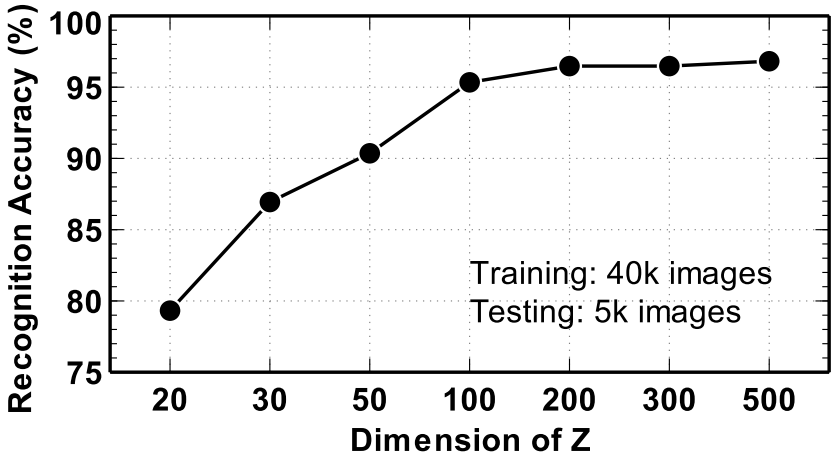


Fig. 2.3 Learning accuracy as a function of Z dimension. © 2016 IEEE.

2.2 Limited On-Chip Precision of SC

For on-chip implementation of the SC algorithm, it is necessary to limit the precision of D and Z in the algorithm as the chip cannot afford the floating-point computation. In the crossbar array architecture, the values in the Z vector are stored on local memories in the peripheral circuitry, and the values of the D matrix are represented by the synaptic weights in the array. Fig. 2.4 shows the learning accuracy with different precisions by truncation of the bits in the SC algorithm. It suggests that a 4-bit Z is sufficient for high learning accuracy and the limited precision of D has more impact on the accuracy. For example, D should be at least 6 bits to achieve an accuracy >95%. This requirement of a high precision in the weight update for the learning is also reported in other recent works [73, 74]. As the training

of these algorithms are error-driven, thus high precision is required to preserve the error information. Since the number of bits D is related to how many levels of conductance that the synaptic device can achieve, a 6-bit D (64 levels) is chosen for later analysis based on the number of multi-level states available in today's synaptic devices (see Fig. 1.3).

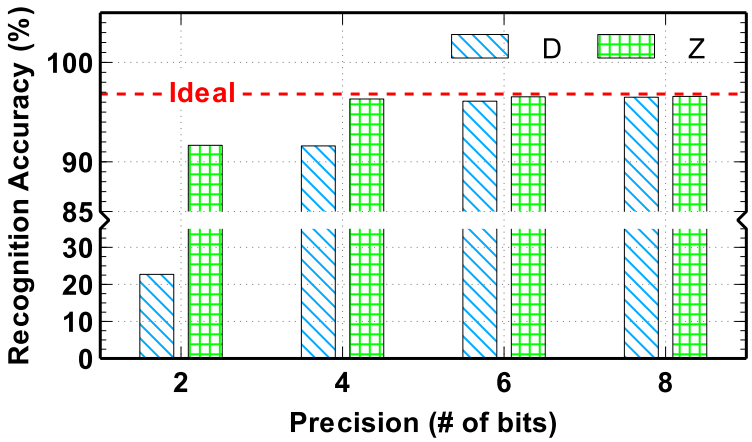


Fig. 2.4 Learning accuracy with different precision bits of D and Z in the SC algorithm. © 2016 IEEE.

2.3 Realistic Device Properties and Mitigation Strategies in Synaptic Array

As previously shown in Fig. 1.3, realistic synaptic behaviors include 1) the nonlinearity and 2) device variations in weight update, and 3) the read noise 4) limited ON/OFF weight ratio in weighted sum. The circuit model of synaptic device is also considered in the array-level analysis. In this section, these realistic properties are modeled individually into the sparse coding algorithm and their impact on the learning accuracy is studied. As a baseline, the limited precision of the synaptic devices (64 levels) is considered.

2.3.1 Nonlinear Weight Update

To analyze the impact of nonlinear weight update on the learning, a general behavior that models the conductance change of weight increase (G_{LTP}) and decrease (G_{LTD}) with the number of pulses (P) is described with the following equations:

$$G_{LTP} = B\left(1-e^{\left(-\frac{P}{A}\right)}\right) + G_{min} \tag{2.4}$$

$$G_{LTD} = -B\left(1-e^{\left(\frac{P-P_{max}}{A}\right)}\right) + G_{max} \tag{2.5}$$

$$B = \frac{G_{max}-G_{min}}{1-e^{\frac{-P_{max}}{A}}} \tag{2.6}$$

where G_{max}, G_{min} and P_{max} can be directly extracted from the experimental data, which represents the maximum conductance, minimum conductance and the maximum pulse number required to switch the device between the minimum and maximum conductance states. *A* is the parameter that controls the nonlinear behavior of the weight update, and *B* is simply a function of *A* that fits the functions within the range of G_{max}, G_{min} and P_{max}. *A* and *B* may be different in Eq. (2.4) and Eq. (2.5). A set of nonlinear weight increase and decrease behaviors can be obtained by adjusting *A* as illustrated in Fig. 2.5(a), where each nonlinear curve is labeled with a nonlinearity value from +6 to -6. Here the plus and minus are merely the signs to label weight increase and decrease, respectively.

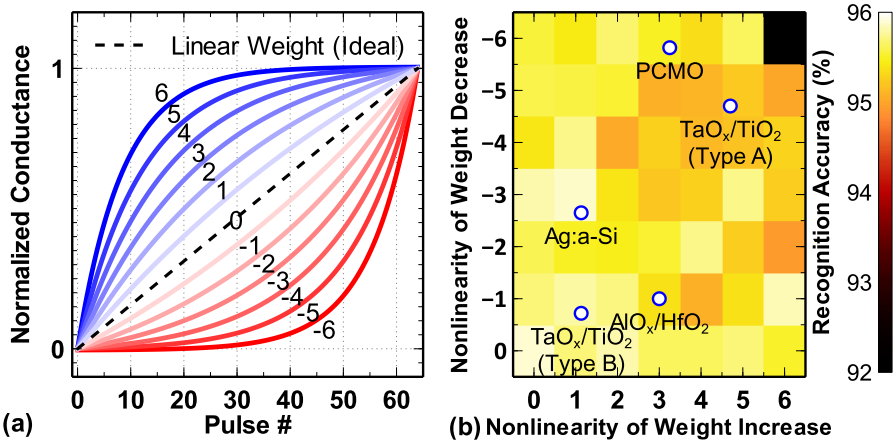


Fig. 2.5 (a) Different nonlinearities of the weight increase and decrease fit from +6 to -6. (b) Learning accuracy with different weight update nonlinearities. Nonlinearities of the reported synaptic devices from Fig. 1.3 are also shown. © 2016 IEEE.

Then, we apply these nonlinear functions into the weight update in the SC algorithm. Fig. 2.5(b) shows that learning accuracy slightly decreases in the high nonlinearity region of weight increase and decrease, and a relatively larger drop from ~96% to ~92% occurs at maximum nonlinearities (+6/-6 curves). For today's synaptic devices (Fig. 1.3), the nonlinearities of weight increase and decrease are also labeled in the Fig. 2.5(b). It is shown that the nonlinearity in the weight update has a moderate impact on the learning performance.

To improve the nonlinearity, the programming scheme that updates the weight can be smartly designed. Fig. 2.5(a) shows different programming schemes and Fig. 2.5(b) shows the corresponding measured experimental weight update in TaO_x/TiO_2 (Type A) based synaptic device. Scheme A uses a simple pulse train for both weight increase and decrease with identical pulses, which leads to the largest nonlinearity. Scheme B is a reinforcement of Scheme A, which splits a single voltage pulse into a pair of positive and negative pulse with different amplitude and duration. It improves the linearity as the second negative pulse

can cancel some overshoot of first positive pulse at the beginning stages of weight update. Scheme C is a more complicated extension of Scheme A, where the pulse duration varies depending on the current conductance state of the synaptic device. The idea of Scheme B and C is to slow down the weight update at the beginning stages of weight update. It is observed that with identical pulses, the conductance changes very rapidly at the beginning stages of the weight update and then it gradually saturates. The duration of programming pulses can be adjusted in a way that a shorter pulse is applied at the beginning stages while gradually wider pulses are applied at subsequent stages. For this scheme, an empirical function to determine the pulse duration (P_D) is required to program the device from conductance state n to n+1 (n=0: minimum (maximum) conductance for weight increase (decrease)) is expressed as:

$$P_D(n \rightarrow n+1) = P_i \times e^{\left(\frac{m \times n}{50}\right)} \qquad m = \begin{cases} 6, & \text{Weight increase} \\ 4, & \text{Weight decrease} \end{cases} \tag{2.7}$$

where P_i is the initial pulse duration.

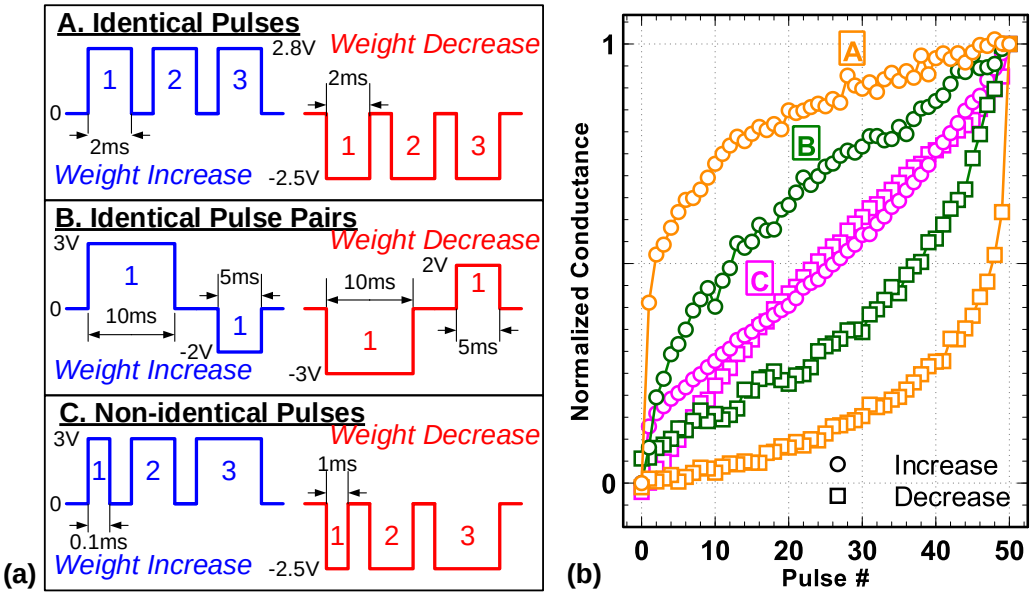


Fig. 2.6 (a) Different programming schemes and (b) the corresponding experimentally measured data of conductance modulation in the TaO_x/TiO_2 (Type A) based synaptic device. © 2015 IEEE.

We can see that with Scheme C, the weight update approaches the linear curve. However, the trade-off is that Scheme C needs a read-before-write step to first identify the conductance state and then apply the correct pulse duration to the device, which inevitably increases the complexity of the peripheral circuitry design as well as the latency and energy consumption. In particular, Scheme C cannot be applied with the parallel weight update scheme [56] due to non-identical pulses. Instead, the weight matrix can only be updated in a sequential row-by-row fashion. In contrast, Scheme B, although it uses a pulse pair as one programming operation, it does not need the read-before-write step as the pulse pair shape is independent of the conductance state. For quick estimation on the overhead of the array, the latency and energy consumption are simply calculated by the applied pulse widths and voltage amplitudes in these schemes. Assuming that the sparse Z vector has 5% nonzero elements, a weight increase from the 30^{th} back to 20^{th} weight level gives ~7.5X

latency for Scheme B and ~60X for Scheme C. The overhead of energy consumption will be a bit less than that of latency as the root-mean-square (RMS) values of voltage in Scheme B and C are smaller. From peripheral circuit design's point of view, identical pulse pairs of Scheme B can be realized by two pulse generators at the two ends of the array to generate a pair of pulses with different polarities. On the other hand, Scheme C needs at least an extra computing unit to calculate Eq. (2.7) and a special pulse generator to produce non-identical pulses with fine-grained duration. Since the algorithm has resilience to the moderate weight nonlinearity, Scheme B may be a better choice for a practical implementation considering the overhead of Scheme C. However, given the accuracy loss of ~4% at the maximum nonlinearities in the sparse coding algorithm, we think it is not crucial to apply the smart programming schemes thus those overheads can be saved.

2.3.2 Device Variations

It is well known that the synaptic devices involving drift and diffusion of the ions/vacancies show considerable variation from device to device, and even from pulse to pulse within one device. Owing to the device-to-device weight update variation, different devices in the array will follow different nonlinearity baselines. Owing to the cycle-to-cycle weight update variation, there will be pulse to pulse noise on top of the nonlinearity baseline. Owing to the read noise, the read-out current of a weight state will have some temporal fluctuation.

The effect of device-to-device variation can be analyzed by introducing the variation into the nonlinearity baseline for each synaptic device, as illustrated in Fig. 2.7(a). For example, if a synaptic device has a +100% device-to-device variation, there will be a +1 deviation of the nonlinearity. As shown in Fig. 2.7(b), the learning accuracy is

insignificantly affected by the device-to-device variation with 30% standard deviation from the baseline.

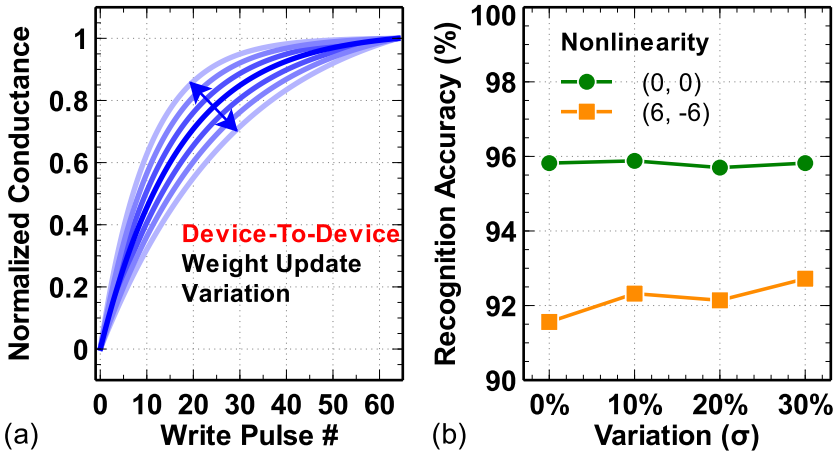


Fig. 2.7 (a) Illustration of the device-to-device weight update variation. Different devices in the array will follow different nonlinearity baselines. (b) Learning accuracy with different standard deviations of device-to-device weight update variation, which has almost no impact. © 2016 IEEE.

The cycle-to-cycle variation of the conductance occurs at every write pulse operation on the synaptic device, as illustrated in Fig. 2.8(a). The amount of cycle-to-cycle variation (σ) is expressed in terms of the percentage of entire weight range. As shown in Fig. 2.8(b), the learning accuracy degrades significantly with larger cycle-to-cycle weight update variation.

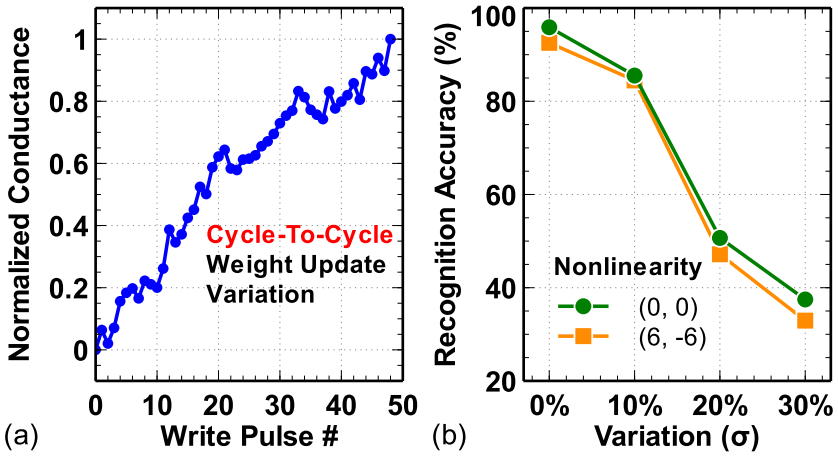


Fig. 2.8 (a) Illustration of the cycle-to-cycle weight update variation. There is pulse to pulse noise on top of the weight update curve. (b) Learning accuracy with different standard deviations of cycle-to-cycle weight update variation, which has significant degradation on the learning accuracy with both nonlinearity baselines (0, 0) and (6, -6). © 2016 IEEE.

2.3.3 Read Noise in Weighted Sum

Similar to the cycle-to-cycle weight update variation, the read noise occurs at every read access to the synaptic device, but the average conductance state is not disturbed. As illustrated in Fig. 2.9(a), the read-out current fluctuates at different conductance states with different number of read pulses. Fig. 2.9(b) shows significant degradation of learning accuracy due to the read noise. The impact is even more critical with nonlinearity baseline (6, -6). We have measured a variation of ~2.89% in the read noise in our TaO_x/TiO_2 based synaptic device, which could cause the accuracy drop below 90% considering this read noise effect only.

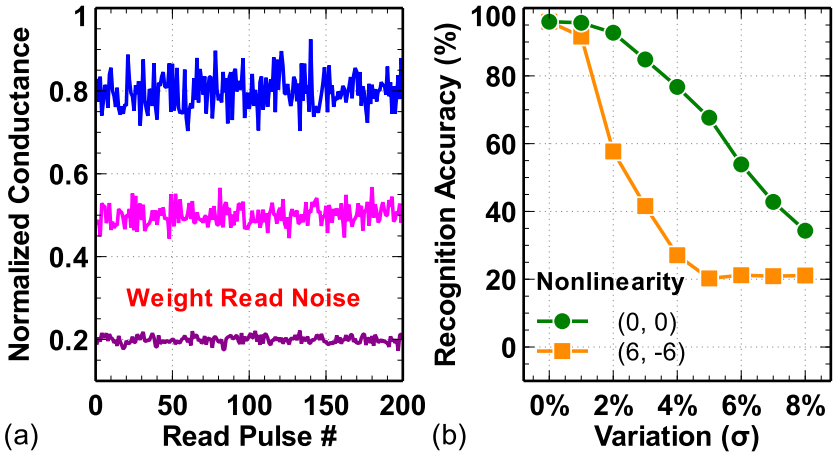


Fig. 2.9 (a) Illustration of the weight read noise. There is read noise on top of each weight level. (b) Learning accuracy with different standard deviations of weight read noise, which causes significant degradation on the learning accuracy and is even severer at nonlinearity baseline (6, -6). © 2016 IEEE.

To alleviate the impact of device variations, we propose using multiple cells as one D weight element. This approach statistically averages out all the conductance variations of synaptic devices. If n cells are used as one weight element, the standard deviation of variations will be reduced by a factor of $1/\sqrt{n}$ assuming that variations are normally distributed. Fig. 2.10 shows an example of the reduction on the variation using 9 cells compared to that using only 1 cell. This strategy is to have considerable improvement on the accuracy loss due to device read-out noise, and it does not have a large overhead in the array area as the area is determined by the pitch of the peripheral circuits in the logic design rule. For example, the array cell height should be aligned with the standard cell height of the array row driver. We estimate that the layout area of 9 resistive synaptic cells is increased by ~20% compared to that of 1 cell at 65nm technology node and 200nm wire width. It should be noted that part of the peripheral circuitry can be placed underneath the synaptic array to save the total area as the synaptic devices are integrated on top of the

CMOS circuits at the interconnect level. However, using multiple cells inevitably increases the energy consumption by n times.

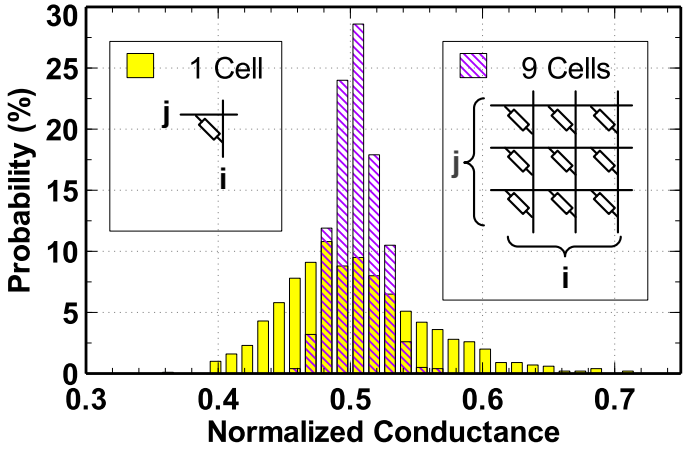


Fig. 2.10 Illustration of multiple cells as one weight element to average out the device variations or read-out noise. © 2016 IEEE.

2.3.4 Limited conductance ON/OFF ratio

Ideally the D values in the SC algorithm are represented by a normalized conductance of synaptic devices, and the range of the D value is from 0 to 1. However, the minimum conductance can be regarded as D=0 only when the ratio between the maximum and minimum conductance (ON/OFF ratio) approaches infinity, which is not feasible in today's synaptic devices. Fig. 2.11 shows the learning accuracy with different ON/OFF ratios. The learning accuracy dramatically decreases when the ON/OFF ratio shrinks below 25, because the calculations involved with small values of D in the algorithm will be significantly distorted. The Ag:a-Si device exhibits a largest ON/OFF ratio of ~15 among the devices in Fig. 1.3, while other devices show even smaller ON/OFF ratio. This means that without any optimization, none of these synaptic devices can achieve high recognition accuracy when used in on-chip implementation of sparse learning.

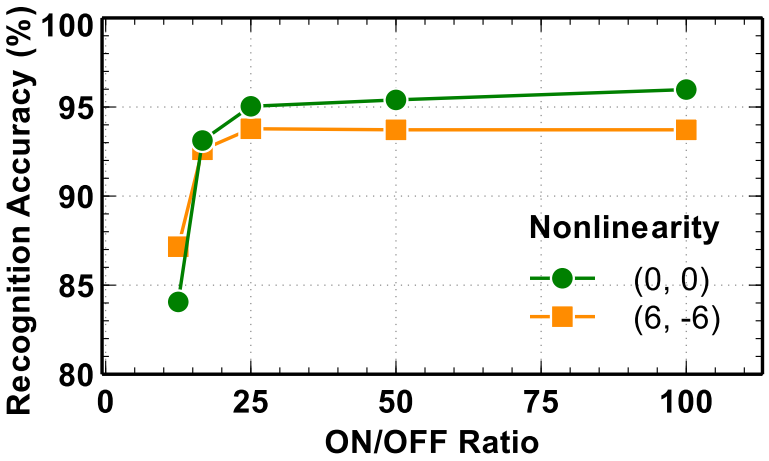


Fig. 2.11 Learning accuracy with different ON/OFF ratios at nonlinearity baselines (0, 0) and (6, -6). The synaptic devices in Fig. 1.3 have a maximum ratio about 15, which results in >10% accuracy loss. © 2016 IEEE.

One approach to remedy this situation is to eliminate the effect of the OFF-state current in every weight element with the aid of a dummy column. The crossbar array architecture with a dummy column is illustrated in Fig. 2.12. The synaptic devices in the dummy column remain in their minimum conductance states, such that the readout value at the output of dummy column represents the weighted sum of the Z vector and the OFF-state conductance. In the peripheral circuitry, we subtract the OFF-state weighted sum from all the partial weighted sums, D_iZ, performed along the columns. Except for spatial variation between the synaptic devices in the same row, this virtually eliminates the effect of OFF-state current in the sparse learning task. An additional column will give 1% overhead on the array area as there are totally 100 columns (X=100), and the area of subtractors is estimated to be ~7.84% of the array area with 9 cells at 65nm technology node and 200nm wire width. However, as the array is able to partially hide the subtractors, its area overhead can be further reduced.

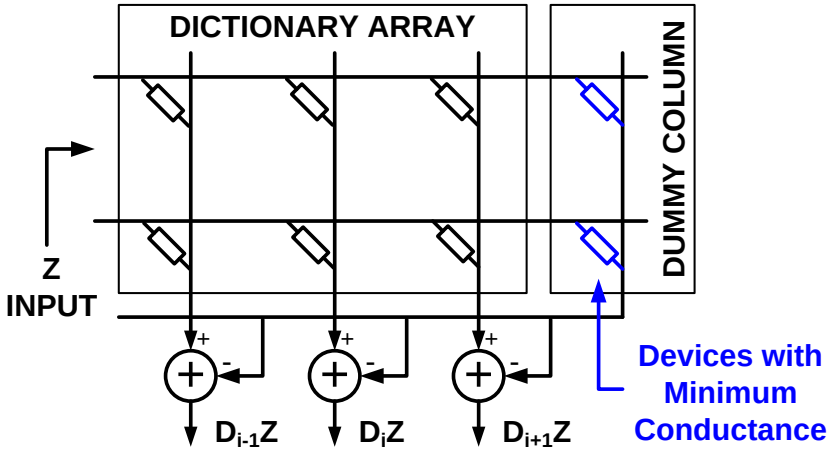


Fig. 2.12 Crossbar array architecture with the dummy column and subtractors to eliminate the common OFF-state current. © 2016 IEEE.

2.3.5 Impact of Weighted Sum IR Drop in Crossbar Array

To simulate the weighted sum operation in SPICE, we model the synaptic device as a resistor in parallel with a capacitor as shown in Fig. 1.4(b). The wire resistances and parasitic capacitances are also considered. The interconnect parameters are obtained from the ITRS table [75]. We extract statistical D, Z and R data at different learning stages from the SC algorithm run by software, and use these values to simulate the weighted sum DZ and DR (in the CD method in Fig. 2.1(a)) by SPICE. The deviation of weighted sum by SPICE is then calculated and incorporated back into the SC algorithm to evaluate its impact on the learning accuracy. Fig. 2.13 shows the learning accuracy with different wire widths. Wires with smaller width have larger wire resistance, thus the weighted sum becomes inaccurate and the learning accuracy is significantly reduced. To alleviate this, we propose reverse scaling on the wire's geometrical dimension, preferably with a wire width larger than 100 nm. Such reverse scaling plus the redundant cells for reduction of device variations dramatically increase the array area, but this may be acceptable considering the

size of peripheral logic gates is complicated and it is thus comparable to the cell pitch of a synaptic cell in the array design.

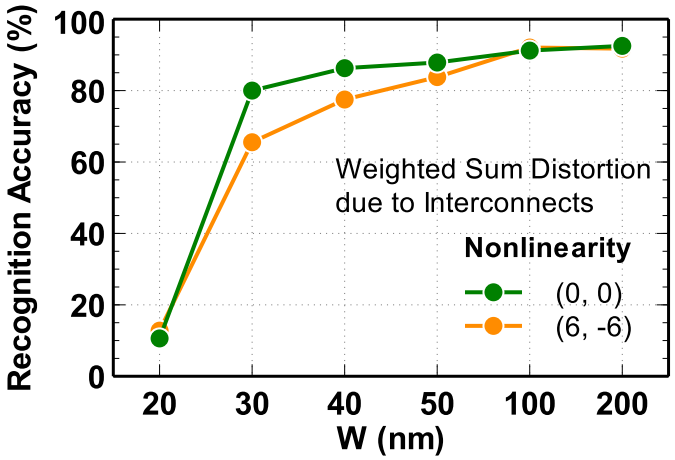


Fig. 2.13 Learning accuracy with different wire width. Smaller wire width will degrade the learning accuracy due to interconnect effect. © 2016 IEEE.

2.4 Accuracy Improvement by Proposed Strategies

If we combine all the non-ideal device effects and array parasitics mentioned above, the learning accuracy of the system drops terribly low to ~30%. Now we implemented the proposed mitigation strategies into the SC algorithm. Specifically, it is assumed that the following improvements on the realistic properties are achieved: 1) the ON/OFF weight ratio is increased by 4X from 12.5 (within the range of the Ag:a-Si device) to 50, using a dummy column but assuming that the OFF-state current is not completely removed due to device-to-device variation; 2) 9 cells as a weight element is used to reduce the variation of read noise from ~2.89% to ~0.96%. It is also assumed that the nonlinearity is large ((4.7, -4.7) for the TaO_x/TiO_2 (Type A) based synaptic device) and the array wire width is relaxed to be 200 nm. As shown in Fig. 2.14, the recognition accuracy of synaptic devices can closely approach that of the ideal algorithm, achieving an accuracy improvement of >65%. However, the proposed strategies will bring some overhead onto the chip area, latency and

energy. Compared to the design without strategies, the area overhead mainly comes from the redundant cells with relaxed wire width (~20% for 9 cells and 200nm wire). The area overhead of the subtractors can be smaller (<7.84%) if they are partially hidden underneath the array. The total latency of weighted sum operation will be similar if the weighted sum current readout is based on the principle of integrate-and-fire neuron model [61], where both the weighted sum current and parasitic column capacitance are increased by 9X and these two effects cancel out each other. The total latency of the weight update will also be similar because the 9 cells are physically wired together and being programmed simultaneously. However, the energy consumption of both the weighted sum and weight update will be increased by ~9X because 9 cells are used.

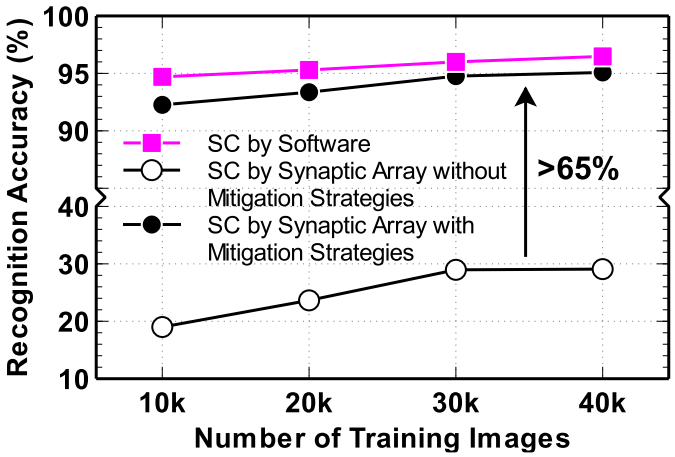


Fig. 2.14 Comparison of the recognition accuracy of the MNIST handwritten digits trained by the sparse coding algorithm using the software approach running and implemented on the hardware architecture with realistic synaptic devices and arrays. With the proposed design methodologies, the recognition accuracy can approach the ideal value of the algorithm. © 2016 IEEE.

2.5 Summary

Synapses are the core elements of a neuromorphic system to establish communication between groups of neurons. Synaptic devices available today exhibit non-ideal device properties, e.g., the nonlinearity in weight update, device variations, read noise and limited ON/OFF weight ratio. The wire parasitics in nanoscale crossbar architecture also cannot be ignored. Sparse coding algorithm is used to provide a platform to evaluate the performance of unsupervised learning using realistic synaptic devices and arrays for image applications. It is found that the non-ideal synaptic device properties and the wire parasitics can lead to significant degradation on image recognition accuracy from ~96% to ~30%. The mitigation strategies to remedy this issue are proposed in this chapter, including 1) the use of multiple cells for each weight element to alleviate the impact of device variations and read noise; 2) a dummy column to eliminate the off-state current; 3) larger wire width to reduce the IR drop along interconnects thereby increase the accuracy of weighted sum. By applying these strategies with tolerable trade-offs on the chip area, latency and energy, the synaptic behavior is greatly improved and the recognition accuracy could come back to ~95%, viably enabling the synaptic devices for practical hardware implementation of the sparse learning algorithm on a chip. We believe that the device-algorithm co-design methodologies presented in this chapter can also be applied to other neuromorphic learning algorithms in general.

3 DESIGN FOR IMPROVEMENT OF SYNAPTIC ARRAY AND NEURON PERIPHERAL CIRCUITS

Traditional crossbar array architecture is a straightforward design to perform fully parallel weighted sum operation, but it suffers from the write disturbance issue and is not energy-efficient, as discussed in Section 1.3. Alternatively, we propose two array architectures, the 1-selector-1-resistor (1S1R) array architecture and pseudo-crossbar array architecture, to improve the current crossbar array design. In this chapter, the design of the neuron peripheral circuits is also discussed. It is found that a compact two-terminal device that exhibits metal-insulator-transition (MIT) phenomenon can potentially replace the existing CMOS integrate-and-fire neuron to achieve smaller area and better performance.

3.1 Reformation of Array Architecture

3.1.1 1S1R Array Architecture

Fig. 3.1(a) shows the schematic of the proposed architecture of the 1S1R array architecture. There is one selector in series with one synaptic device at each cross-point. The selector introduces nonlinear I-V characteristics for the synaptic device and is helpful for weight update and/or weighted sum operations when there are unselected rows/columns, which will be discussed later in this section. Same as the traditional crossbar array, a read voltage (V_R) is applied in parallel to each row to compute the weighted sum in the read operation. Because of the selector, the current at each cross-point is not exactly the multiplication of V_R and the conductance of the synaptic device. Therefore, the resistance of the selector at V_R must be much smaller compared to the resistance of the synaptic device.

The weight update operation in 1S1R array resembles the one in traditional crossbar array, but the voltage biasing is a little bit more complicated. As shown in Fig. 3.1(b), the

write scheme to increase (decrease) the weight in the write operation is to select one row at a time with the write voltage (V_W) (0 V) applied at the edge, while other unselected rows and columns are biased at an intermediate voltage V_X to prevent the write disturbance. Then, the V_X-0-V_X negative (V_X-V_W-V_X positive) write pulses are then applied to increase and decrease the weight in the selected cells, respectively.

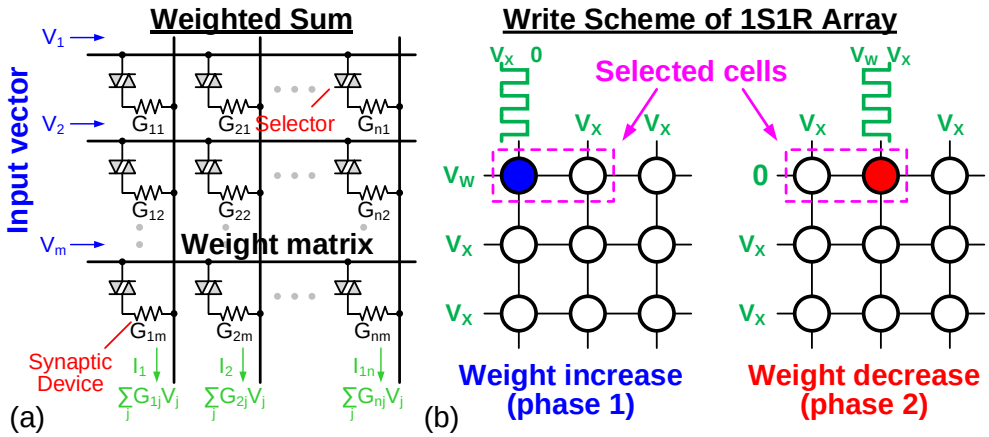


Fig. 3.1 (a) The proposed 1S1R array architecture. The selector is in series with the synaptic device. Read scheme is shown for performing the weighted sum in parallel, which is similar to the traditional crossbar array. (b) The weight update operation with weight increase and decrease phase. The selected row is biased at V_W and 0 V, and V_X-0-V_X negative and V_X-V_W-V_X positive write pulses are applied to increase and decrease the weight in the selected cells, respectively. © 2016 IEEE.

As mentioned in Section 1.3, the proposed weight update scheme suffers from the leakage problem, as the crossbar array is partially selected and the leakage paths exists in the half-selected cells on other unselected rows or columns. The half-selected cells can see a voltage drop of V_X (weight increase) or V_W-V_X (weight decrease) during the weight update. Therefore, the selector is proposed to connect in series with the synaptic device to suppress the leakage current at these voltages. Fig. 3.2 shows the I-V characteristics of a TaO_x/TiO_2 (Type A) based synaptic device in ON state, the selector and the series of these

two devices. In this study, we use the mixed-ionic-electronic-conduction (MIEC)-based selector with high nonlinearity (~85 mV/dec) [76] and set the original V_W=2 V and V_R=1 V for a single synaptic device. Without the selector, V_X is designed to be 1 V, which is the V/2 write scheme in traditional memory application [77]. With the selector, the overall cell resistance is increased, which reduces the IR drop along interconnects in weighted sum while only affecting little on the mapping from device conductance to weight values because the conductance of selector is relatively higher than the conductance range of synaptic device at 1 V. Also, the selector can reduce the leakage on the half-selected cells in weight update, and it does not affect the weight update because at sufficient large voltage it is already turned on. In this case, V_W should be increased to 3 V and the V_X for weight increase and decrease are then 1 V and 2 V, respectively. It can ensure the voltage drop on the selected cells to be 2 V, which is the same as the original write condition for a single synaptic device. Also, the voltage drop on the half-selected cells will then be 1 V, where the leakage reduction is ~10X as shown in Fig. 3.2. Since most of the cells during the weight update are half-selected, the energy consumption is greatly reduced compared to the traditional V/2 write scheme in crossbar array where the voltage drop of half-selected cells are $V_W/2$=1.5 V.

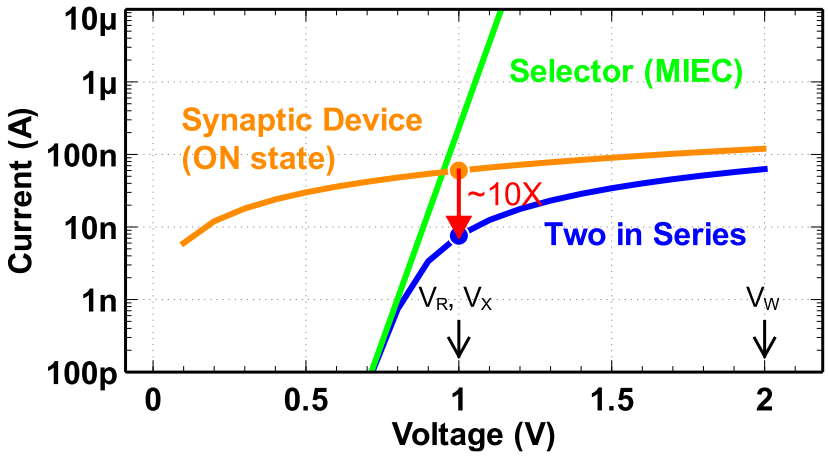


Fig. 3.2 The I-V characteristics of resistive synaptic device, MIEC selector [76] and the series of the above two devices. With selector, the cell resistance is increased to alleviate the IR drop along interconnects in the weighted sum and the leakage on the unselected cells (~10X reduction), without affecting the weight update. V_W, V_R and V_X labeled here are the voltages for the original synaptic array without the selector. © 2016 IEEE.

3.1.2 Pseudo-crossbar Array Architecture

The write disturbance problem in crossbar array architecture is also a concern in the conventional memory application. A common design solution is to add a access transistor in series with the eNVM device, forming the one-transistor one-resistor (1T1R) array architecture, as shown in Fig. 3.3(a). The word line (WL) controls the gate of the transistor, which can be viewed as a switch for the cell. The source line (SL) connects to the source of the transistor. The eNVM cell's top electrode connects to the bit line (BL), while its bottom electrode connects to the drain of the transistor through a contact via. In such case, the cell area of 1T1R array is then determined by the transistor size, which is typically >$6F^2$ depending on the maximum current required to be delivered into the eNVM cell. Larger current needs larger transistor gate width/length (W/L). However, the conventional 1T1R array is not able to perform the weighted sum operation that follows Eq. (1.1). In this case,

we have to modify the conventional 1T1R array by rotating the BLs by 90°, which is named as the pseudo-crossbar array architecture [60]. In weighted sum operation, all the transistors will be transparent when all WLs are turned on. Thus, the input vector voltages are provided to the BLs, and the weighted sum currents are read out through SLs in parallel. It should be noted that the IR drop problem still exists in the pseudo-crossbar array, and the wire RC model in Fig. 1.4(b) can also be applied here to study the IR drop problem.

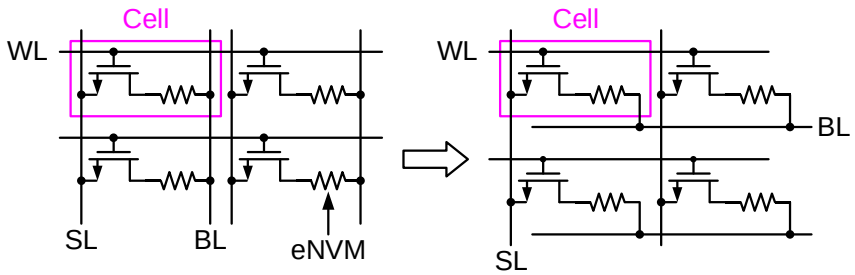


Fig. 3.3 Transformation from (a) conventional 1T1R array to (b) pseudo-crossbar array by 90° rotation of BL to enable weighted sum operation. © 2018 IEEE.

The voltage bias schemes for weight update is shown in Fig. 3.4. As the weight increase and decrease need different programming voltage polarities, the weight update process requires 2 steps with different voltage bias schemes, which is similar to the crossbar array. In weight update, the selected cells will be on the same row, and programming pulses or biases (if no update) are provided from the SL, allowing the selected cells to be tuned differently in parallel. To perform weight update for the entire array, a row-by-row operation is still necessary. Typically, the entire row will be selected at a time to ensure the maximum parallelism. With only the selected WL activated, the unselected cells at all other rows can be free from the write disturbance, meanwhile achieving significant reduction on the energy consumption in biasing these unselected rows.

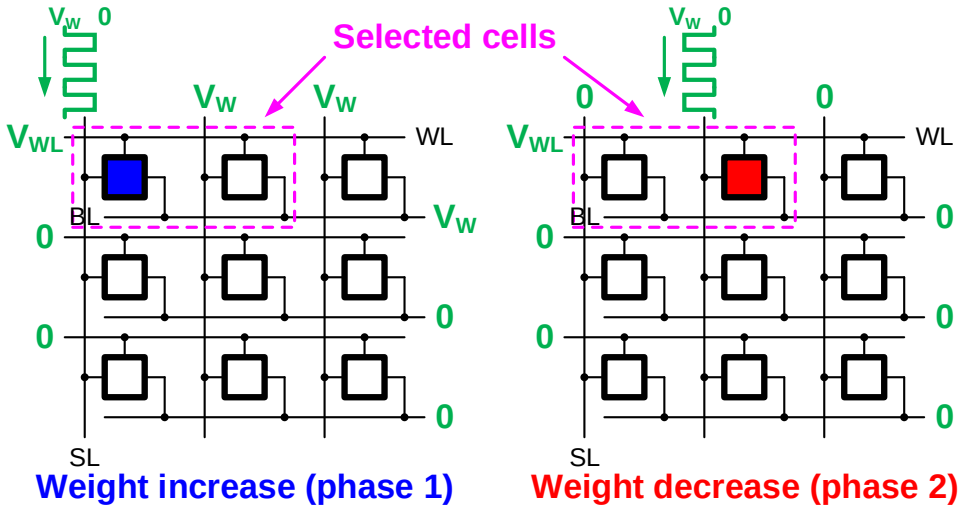


Fig. 3.4 Voltage bias scheme in the write operation of pseudo-crossbar array. Two separate phases for weight increase and decrease are required. In this example, the left cell of the selected cells will be updated in phase 1, while the right one will be updated in phase 2. © 2018 IEEE.

3.2 Design of Neuron Peripheral Circuits

Besides the synaptic array, several neuron peripheral circuits are needed to construct a standalone weighted sum computation unit. Fig. 3.5 shows the circuit block diagrams for crossbar and pseudo-crossbar array architecture. In weighted sum operation, crossbar and pseudo-crossbar array need WL and BL switch matrix to pass the input vector voltages, and the weighted sum results will be read out through the multiplexer (Mux), read circuits and shift-add circuits. In weight update operation, both arrays need two switch matrixes implement the array write scheme such as Fig. 3.4. In this section, these key neuron peripheral circuits will be introduced in detail.

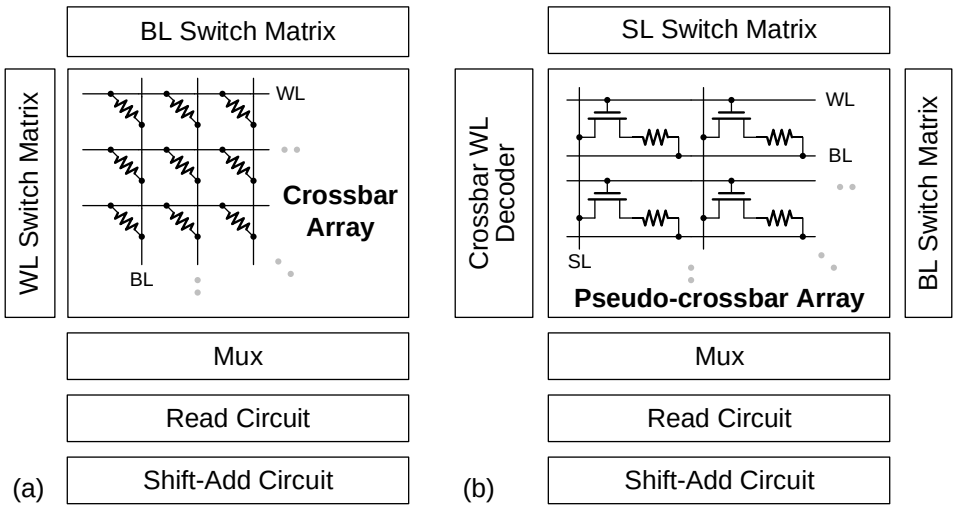


Fig. 3.5 Circuit block diagram for the (a) crossbar and (b) pseudo-crossbar array architectures.

3.2.1 Crossbar WL Decoder

The WL decoder is modified to be "crossbar WL decoder" in pseudo-crossbar array, which has an additional feature to activate all the WLs for making all the transistors transparent for weighted sum. Inspired from [78], the crossbar WL decoder is constructed by attaching the follower circuits to every output row of the traditional decoder, as shown in Fig. 3.6. If *ALLOPEN*=1, the output of the decoder will not be taken into account, and all the transmission gates in the follower circuits become open, which allows the input voltage (V_{IN}) pass through all the transmission gates thus all the WLs are activated. If *ALLOPEN*=0, the crossbar WL decoder will function as a traditional WL decoder, which activates one WL at a time. It should be noted that the follower circuits are designed using transmission gates with V_{IN} instead of digital logic gates with V_{DD} as the WL voltage, because the WL voltage may have to be different for the weighted sum (read) and weight update (write) operation.

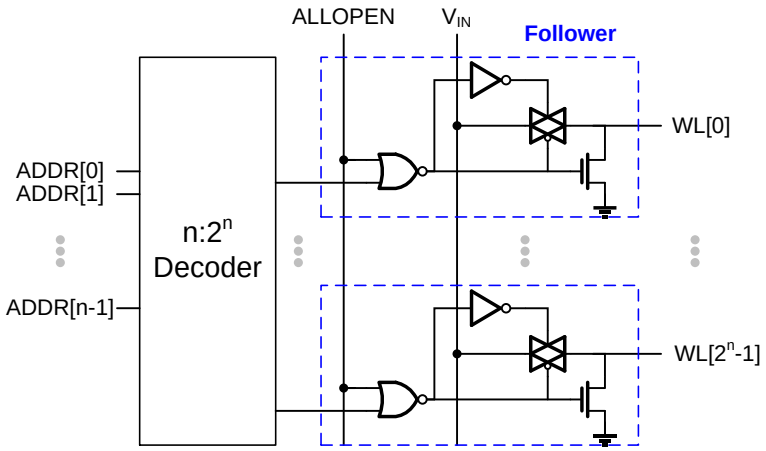


Fig. 3.6 Circuit diagram of the crossbar WL decoder. Follower circuit is attached to every row of the decoder to enable activation of all WLs when *ALLOPEN*=1. © 2018 IEEE.

3.2.2 Switch Matrix and Input Vector Encoding

In both crossbar and pseudo-crossbar array architectures, switch matrixes are used for fully parallel voltage input to the array rows or columns. Fig. 3.7(a) shows the BL switch matrix for example. It consists of transmission gates that are connected to all the BLs, with control signals (B_1 to B_n) of the transmission gates stored in the registers (not shown here). In the weighted sum operation, the input vector signal is loaded to B_1 to B_n, which decide the BLs to be connected to either the read voltage or ground. In this way, the read voltage that is applied at the input of transmission gates can pass to the BLs and the weighted sums are read out through SLs in parallel. If the input vector is not 1 bit, it should be encoded using multiple clock cycles. As mentioned earlier, the reason why we do not use analog voltage to represent the input vector precision is due to the I-V nonlinearity of eNVM, which will cause the weighted sum distortion or inaccuracy [56]. As shown in Fig. 3.7(b), B_1 to B_n are a vector of bit streams. To obtain the final weighted result, the shift and add circuit in Fig. 3.5 will perform shift and add on the weighted sum results of all bit cycles.

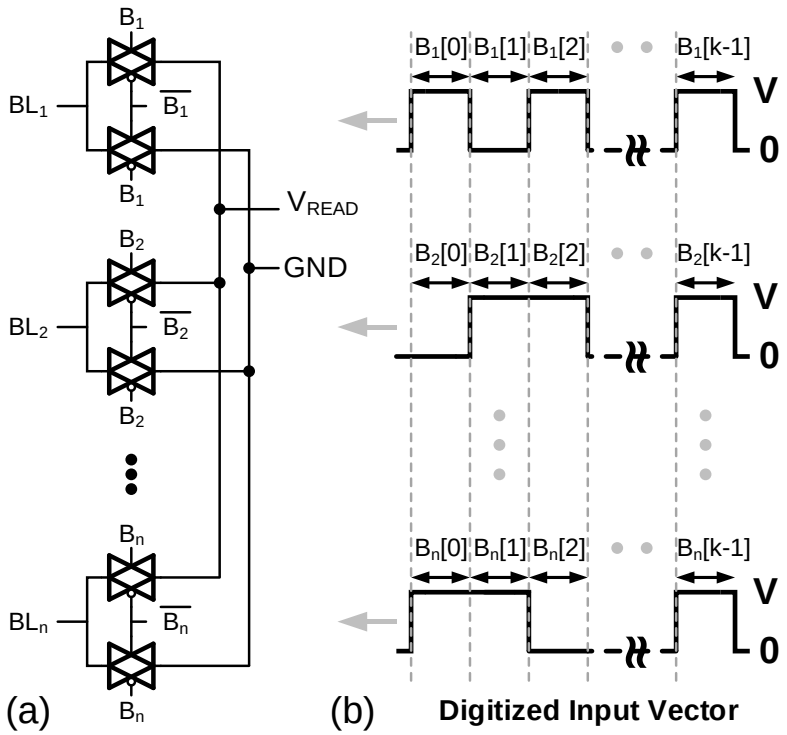


Fig. 3.7 (a) Transmission gates of the BL switch matrix in the weighted sum operation. A vector of control signals (B_1 to B_n) from the registers (not shown here) decide the BLs to be connected to either a voltage source or ground. (b) Control signals in a bit stream to represent the precision of the input vector. © 2018 IEEE.

3.2.3 Read Circuit as ADC

To convert these analog weighted sum currents to digital outputs, we designed the read circuit [61] to employ the principle of the integrate-and-fire neuron model, as shown in Fig. 3.8. The read circuit integrates the weighted sum current on the finite capacitance of the array column. Once the voltage charges up above a certain threshold, the read circuit fires an output pulse and the capacitance is discharged back. The counter after the read circuit then converts the number of output spikes to digital data. The precision required for this analog-to-digital conversion (ADC) determines the pulse width in each bit of the input vector. As the cell size in 1T1R array is much smaller compared to the ADC size, multiple

synaptic array columns may share one ADC through a Mux to improve the area efficiency. However, this inevitably increases the latency of weighted sum as time multiplexing is necessary because of the sharing.

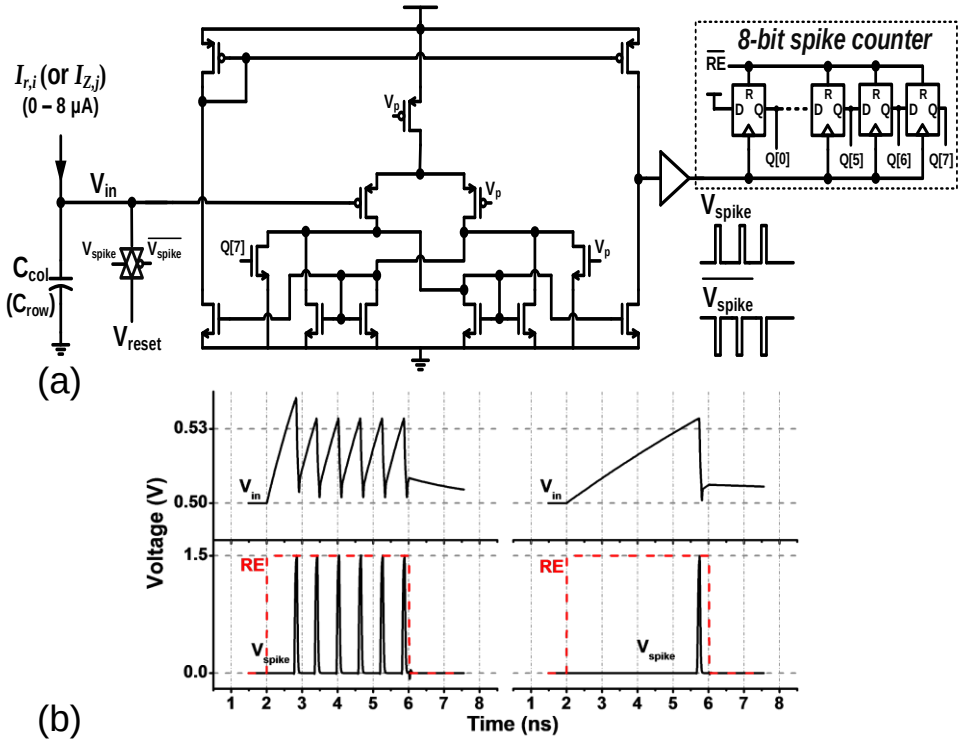


Fig. 3.8 (a) Read circuit that converts the analog column current to digital outputs. (b) The simulated integration-and-fire waveform of the read circuit. © 2015 IEEE.

3.3 Compact Oscillation Neuron Exploiting Metal-Insulator-Transition

Today's CMOS integrate-and-fire neuron typically requires tens of transistors. As shown in Fig. 3.8, such complex CMOS neuron causes the column pitch matching problem. As discussed in the previous section, multiple columns have to share one neuron, thereby reducing the parallelism as the time-multiplexing is needed to sequentially read out all the weighted sum from the array. In such context, we propose a compact oscillation neuron using the metal-insulator-transition (MIT) device in order to replace the CMOS neuron. Prior eNVM designs [79-81] mostly focused on the synaptic array core instead of the

peripheral neuron node. A recent experimental work demonstrated the oscillation neuron with small-scale synaptic array [82], however, how to design a large-scale synaptic array with oscillation neuron remains unexplored. In this section, we will analyze the impact of MIT device characteristics on the weighted sum accuracy, study the feasibility of oscillation neurons connected to the resistive synaptic array, and benchmark circuit-level performances with the CMOS neuron at both sub-circuit and array level.

3.3.1 Metal-Insulator-Transition Phenomenon

The metal-insulator-transition (MIT) phenomenon occurs in strongly correlated oxides, where the oxides switch between a metallic state and an insulating state under certain external excitation, thermally or electrically [83]. The MIT device shows a threshold switching I-V characteristics with hysteresis and theoretically 2-5 orders of magnitude ON/OFF ratio. For the Mott transition in strongly correlated oxides, the bandgap collapses when the carrier density in the materials is larger than the critical carrier density n_c, resulting in the insulator-to-metal transition. Carrier density in the materials can be increased by either thermal injection or electric injection. Therefore, the threshold switching has a critical temperature (T_C) or a critical threshold voltage (V_{TH}). Among all the Mott oxides, the research in the literature extensively focused on VO_2 as the representative material system for studying the physical mechanism. However, VO_2 is not suitable for on-chip integration because its T_C ~67 ^oC [84] is relatively low, and the threshold switching behavior disappears above T_C. What makes the circuit design more challenging is the fact that the V_{TH} of VO_2 strongly depends on the environment temperature even below T_C. For this reason, we select an emerging material NbO_2 with an

extremely high T_C ~808 ^oC [84] that has superior thermal stability. Recent experiments have shown the on-chip integration of NbO_2 with the CMOS platform [85].

The MIT device has been listed as an emerging device candidate in the ITRS roadmap for logic switch [75], still lacking demonstrations to be competitive in practical applications. For example, the steep-slope field-effect transistor with strongly correlated oxides as the channel material suffers from the low carrier mobility [86]. The recent revival of MIT device is owing to its capability to serve as a two-terminal selector device for the crossbar memory array to suppress the sneak paths [85]. Different from these works, we propose to use MIT device as the oscillation neuron in neuromorphic computing. Using the coupled-oscillators as phase encoding for the computation-hard optimization problems have been proposed [87-89], however here we take a different approach of using the oscillators: we utilize the oscillation as an integrate-and-fire neuron's output waveform.

Fig. 3.9(a) shows the hysteresis I-V characteristic of a typical MIT device [83]. We have built a Verilog-A behavior model to capture the switching characteristics with parameters such as the resistance in the ON/OFF state (R_{ON}/R_{OFF}), the threshold voltage (V_{TH}), and the hold voltage (V_{HOLD}). The MIT device is initially in the OFF state, and it will switch to the ON state once the applied voltage exceeds V_{TH}. When the applied voltage across the MIT device is smaller than V_{HOLD}, it will switch back to the OFF state. Therefore, the resistive switching in the MIT device is essentially "volatile", unlike the "non-volatile" resistive switching in the eNVM.

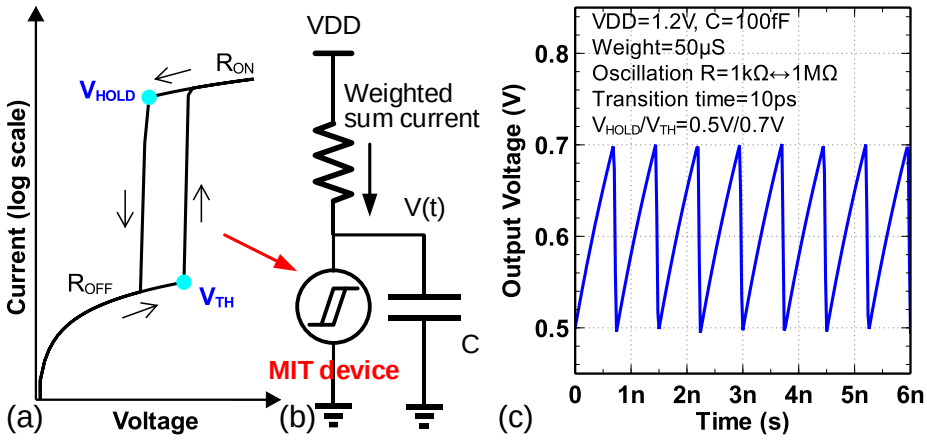


Fig. 3.9 (a) Hysteresis I-V characteristics of a MIT device. [83] (b) Circuit configuration of an oscillation neuron node with MIT device and eNVM synaptic weight. (c) SPICE simulation waveform of the oscillation neuron in (b). © 2016 IEEE.

The intrinsic transition time in the MIT device is defined as the time required to switch between R_{ON} and R_{OFF}. To make the neuron node oscillate, we have to connect a resistor (e.g. an eNVM synapse) with the MIT device, as shown in Fig. 3.9(b). We assume the eNVM resistance (R) is between MIT device's R_{ON} and R_{OFF}, and there is parasitic capacitance at the neuron node. Initially when the voltage VDD is applied, the node voltage on the capacitor will be charged because most of the voltage drop should be on the MIT device (R_{OFF}>R). Once the node voltage exceeds V_{TH}, the MIT device switches to R_{ON}, and the capacitor starts discharging since the voltage drop on the MIT device becomes small (R_{ON}<R). Once the node voltage decreases below V_{HOLD}, the MIT device switches to R_{OFF}. This charging and discharging process repeats, thus the voltage of the neuron node oscillates between V_{HOLD} and V_{TH}. Fig. 3.9(c) shows the SPICE simulation waveform for the circuit configuration in Fig. 3.9(b). As the charging is through the eNVM and the discharging is through the MIT device at R_{ON}, the RC delay of the charging is larger than that of the discharging, making the voltage oscillation a triangular waveform. The

oscillation of the MIT device in such circuit configurations has been widely observed in various experiments [82, 90-93], showing its feasibility as the oscillation neuron.

By solving the Kirchhoff's Law of Fig. 3.9(b), the analytical solution of the charging time t_{rise} from V_{HOLD} to V_{TH} can be obtained, which is expressed as

$$t_{rise} = -R_r C \times \log\left(\frac{V_{TH} - V_{DD}\frac{R_r}{R}}{V_{HOLD} - V_{DD}\frac{R_r}{R}}\right) \tag{3.1}$$

where $R_r=(R||R_{OFF})$. Similarly, the discharging time from V_{TH} to V_{HOLD} can be calculated as:

$$t_{fall} = -R_f C \times \log\left(\frac{V_{HOLD} - V_{DD}\frac{R_f}{R}}{V_{TH} - V_{DD}\frac{R_f}{R}}\right) \tag{3.2}$$

where $R_f=(R||R_{ON})$. If we assume $R_{OFF}>>R>>R_{ON}$, then $R_r \approx R$ and $R_f \approx R_{ON}$, which makes t_{rise} proportional to eNVM resistance and t_{fall} to be a constant much smaller than t_{rise}. We can obtain the ideal oscillation frequency f by using Eq. (1):

$$f = \frac{W}{C \times \log\left(\frac{V_{HOLD} - V_{DD}}{V_{TH} - V_{DD}}\right)} \tag{3.3}$$

where W=1/R is the weight of the eNVM synapse. *f* is then proportional to the eNVM weight. Therefore, the oscillation frequency represents a weighted sum if the MIT device connects to all the eNVM weights in one column.

3.3.2 Design for Accurate Weighted Sum

In this section, we will set up appropriate MIT device parameters, and then discuss the dependence of the oscillation frequency on applied voltage (VDD) and eNVM weight. The simulation is based on the circuit configuration of Fig. 3.9(b).

A. Setup of MIT Device Parameters

Prior experimental study has shown that V_{HOLD} is dependent on the electrode work function and can be as low as 0.5V, while V_{TH} can be reduced to 1V with smaller oxide thickness [93]. In this case, the VDD is preferred to be 0.5V+1V=1.5V to make the voltage swing of oscillation centered at half VDD. However, this may disturb the eNVM resistance as the voltage across eNVM can reach 1V. In this work, we assume a VDD of 1.2V assuming that V_{TH} can be further scaled down to 0.7V by device engineering towards smaller oxide thickness. We also assume a resistance ON/OFF ratio of 1000 can be achieved with R_{ON}=1kΩ and R_{OFF}=1MΩ to support a wide range of eNVM weight in large-scale arrays, where the parasitic capacitance of one column can be at least several 10's fF and here we will use 100fF as a default parameter. It is noted that the ON/OFF ratio of MIT devices reported today are typically ~100, while the theoretical predicts in single-crystalline phase it can be up to 10^5 [83], or 10^6 if new material, e.g. SiTe, is used [94].

B. Effect of Intrinsic Transition Time

As discussed earlier, the weighted sum will be proportional to the oscillation frequency if t_{rise} is much larger than t_{fall}. However, this statement is under the assumption that the MIT's intrinsic transition time is negligible. To investigate the impact of transition time, we simulate the oscillation frequency as a function of transition time at two different weights 10μS and 100μS, as shown in Fig. 3.10(a). Compared to the analytical results obtained by using Eq. (3.1) and (3.2), the deviation becomes more noticeable with increasing transition time larger than 10ps. Even if the oscillation frequency is small (<300 MHz) with smaller eNVM weight 10μS, the need for fast transition ~10ps is not relaxed. The reason can be attributed to the voltage undershoot below V_{HOLD} that leads to larger t_{rise},

as shown in Fig. 3.10(b). If the transition time is comparable to the RC delay in the discharging phase, the discharging would not stop until the MIT device switches back to a resistance that is high enough to start charging the neuron node. Therefore, the transition time has to be smaller than the discharging RC time to avoid this undershoot issue. It has been reported that the oscillation frequency of MIT devices with the circuit configuration in Fig. 3.9(b) can be up to several 10's to 100's MHz [93, 95]. It is highly probable that the reported frequency is limited by the parasitic RC delay in the off-chip electrical measurement setup. Fortunately, it has been reported the intrinsic transition time in the MIT device can be as fast as picosecond or even in the femtosecond range, suggested by the optical laser pump-probe methods [84].

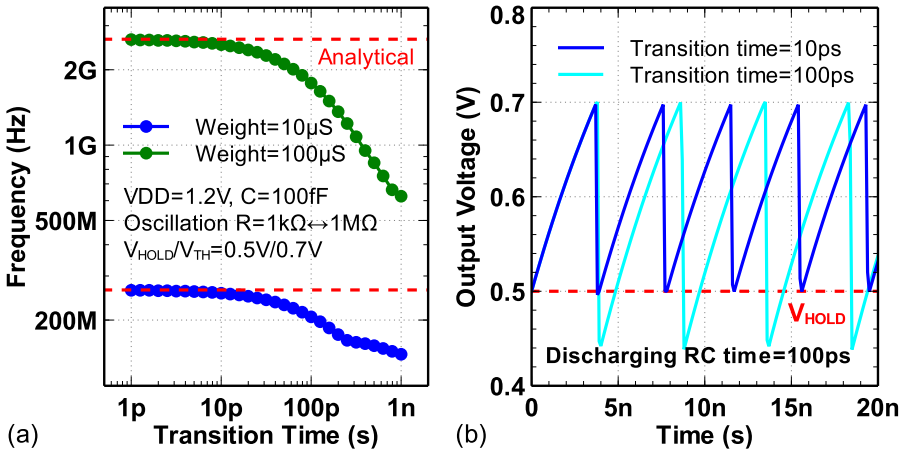


Fig. 3.10 (a) Oscillation frequency as a function of the MIT's intrinsic transition time. Frequency deviates from the analytical value at larger transition times. (b) Undershoot of the voltage discharging below the hold voltage. The transition time needs to be smaller than discharging RC time to prevent the undershoot. © 2016 IEEE.

C. Effect of Applied Voltage Change

Fig. 3.11(a) shows the oscillation frequency as a function of VDD for different weights. It can be seen that the onset of oscillation happens at VDD=V_{TH}=0.7V. The frequency is

roughly proportional to VDD beyond ~1V. This simulation result can be directly verified by using Eq. (3.1) and (3.2). Varying VDD seems to be useful as an encoding scheme of the input vector for the weighted sum operation. However, this might not work in an array because there will be current leakage from one row to another when the row voltages are different. Moreover, VDD should not be large enough (~1.5V) to cause possible disturbance on the eNVM device as mentioned earlier. Within this limited range from 1V to 1.5V, it is difficult to split the VDD into multiple levels due to the noise consideration and practical bias circuit design constraints. Therefore, it is preferred that the input vector to be represented by digital pulses with the same VDD to avoid these issues. We will discuss this later where the oscillation neurons are integrated with the crossbar array and perform array-level operations.

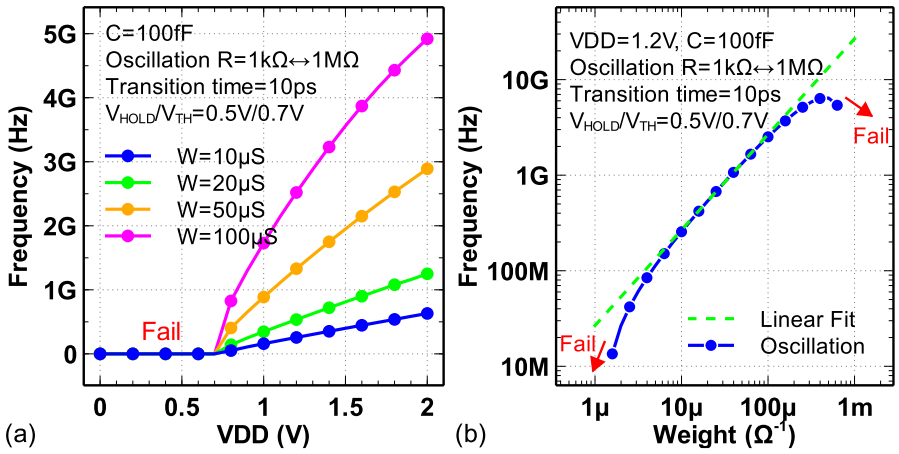


Fig. 3.11 (a) Oscillation frequency as a function of VDD with different weights. Oscillation will not occur when VDD is below V_{TH}. (b) Oscillation frequency as a function of weight. The oscillation neuron has a limited linear weight range. © 2016 IEEE.

D. Effect of Weight Change

The general criterion for the eNVM weight is that its resistance should be within the range of the MIT device resistance (from R_{ON} to R_{OFF}) to make the neuron node oscillate.

It is also preferred that the resistance can satisfy the condition $R_{OFF}>>R>>R_{ON}$ to ensure an accurate weighted sum. Fig. 3.11(b) shows the frequency as a function of the eNVM weight. Since R_{ON}=1kΩ and R_{OFF}=1MΩ, the oscillation would fail when the weight is approaching 1μS and 1mS. The linear region is located at weight values from ~10μS to 100μS. This can be explained by the following: For small weights (large eNVM resistance), the eNVM resistance cannot be ignored compared to the large R_{OFF}, thus the voltage drop on the MIT device is smaller than expected, leading to larger t_{rise} and lower oscillation frequency. For large weights (small eNVM resistance), the eNVM resistance cannot be ignored compared to the small R_{ON}, thus t_{fall} becomes noticeable and the oscillation will slow down. In addition, the intrinsic transition time serves as a hard limit for the oscillation frequency, which will also have insignificant impact on large weights as the frequency is approaching this limit.

3.3.3 Array Implementation for Weighted Sum Operation

A. Crossbar Array Architecture

As mentioned in Section 1.3, the resistive crossbar array architecture with synaptic devices has been proposed to perform the weighted sum operation in a neural network, where the crossbar array represents the weight matrix, with the algorithm weight values mapped to the eNVM device weight range. In this work, we assume the algorithm weight values are normalized between 0 and 1, corresponding to the eNVM minimum and maximum weight, respectively. Fig. 3.12 shows the weighted sum operation in the crossbar array architecture. The input vector is encoded into a digital number of pulses, which controls the transmission gates at each word line (WL) row. Each row will be connected to a fixed voltage if it is selected ($S_i=V_S$), otherwise the transmission gate is turned off and

the row becomes floating (unselected). Then, the total weight of a column is the sum of weights at the selected rows, where the equivalent circuit of a column becomes the configuration in Fig. 3.9(b). With the MIT device connected to the bit line (BL) column, each column can oscillate at different frequencies based on the total weight of the column. The inverter at each column helps restore the oscillation waveform to the rail-to-rail rectangle pulses (VDD to 0), and the ripple counter can convert the number of pulses into a digital value (in binary fashion). However, typically the resistance of a synaptic eNVM device with continuous weight tuning has a limited ON/OFF ratio <10 [33, 35], which makes the minimum eNVM weight not small enough thus it cannot represent a 0 value in the algorithm. To solve this problem, we add a dummy column with all the cells at the minimum weight to eliminate this weight offset, which is the same as the technique presented in Section 2.3.4. Eq. (3.3) shows that ideally the oscillation frequency is proportional to the weight, thus we can subtract the output value of the dummy column from that of the array column to obtain the accurate weighted sum. Finally, to complete the entire weighted sum task, we have to shift and add the weighted sum value at different input vector cycle and get the final weighted sum since the input vector is formed with digitized pulses using a binary representation.

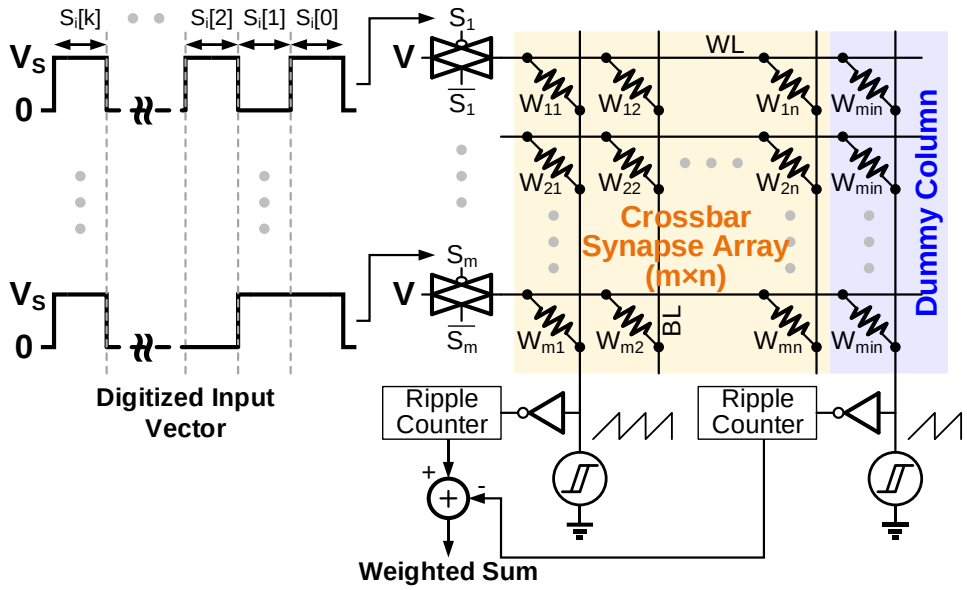


Fig. 3.12 Weighted sum operation with the crossbar array architecture. The input vector is digitized into several read cycles. The dummy column with synapses at minimum weight is added to eliminate the OFF-state weight. The inverter and ripple counter together converts the number of oscillation cycles into digital value. The total weighted sum values are then obtained by subtracting the partial weighted sum value of the dummy column. © 2016 IEEE.

Although the crossbar array has its simple structure to perform the weighted sum operation, it has the well-known sneak path problem that causes interference (or cross-talk) between cells. This problem can be found with the oscillation neuron as well. When the unselected rows are floating, they become the leakage paths between different columns as they have different oscillation frequencies, thus the frequency of each column can get disturbed by other columns. The worst case is when one column has a total weight W_1, and the other columns have the same total weight W_2 for each of them. Then, the voltage oscillation at W_1 column may be significantly affected by the group oscillation behavior of all W_2 columns.

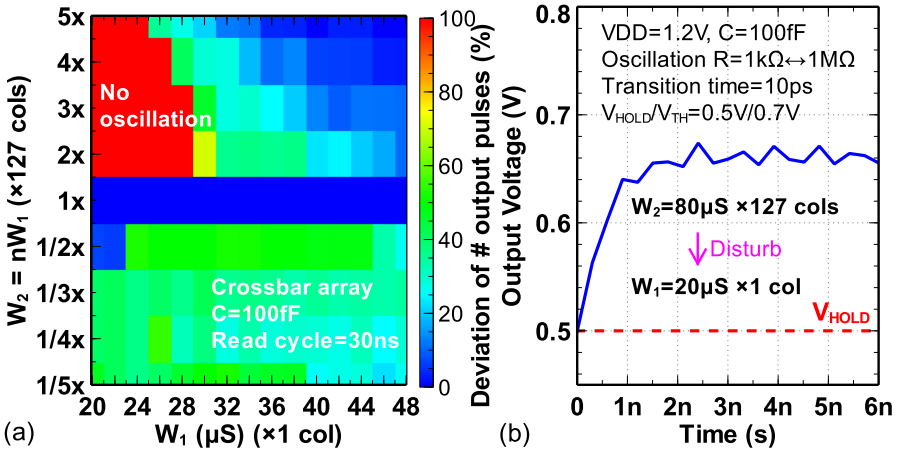


Fig. 3.13 (a) Deviation of the number of output pulses (value after the ripple counter) within 30 ns of a column with total column weight W_1, while each of the other 127 columns has a total column weight W_2 (=nW_1). Oscillation completely stops when W_1 is low and W_2>W_1. (b) An example of failure case with W_1=20μS and W_2=80μS, where the oscillation behavior is interrupted by the W_2 columns. © 2016 IEEE.

To conduct the array-level SPICE simulation, we set the array size to be 128×128, and the minimum and maximum value of a single eNVM weight are 0.4μS and 2μS (ON/OFF ratio=5), respectively. In this case, the total weight of a column can be easily added up to several 10's to 100's μS, which is within the resistance range of the MIT device from the earlier setup. We then simulate all the possible worst cases in the array with different values of W_1 and W_2 at the linear weight range to analyze how much interference can occur between columns, as shown in Fig. 3.13(a). The value of W_2 is taken as n×W_1, where n is from 1/5 to 5 because the eNVM weight ON/OFF ratio is 5. The weight difference between columns is at most 5 times with the same number of rows activated. We measure the number of pulses after the counter within 30 ns, and the results in Fig. 3.13(a) suggest that the deviation from the ideal number of output pulses is generally large at many combinations of W_1 and W_2. There are even extreme cases where no oscillation occurs at

low W_1 with W_2>W_1. Low W_1 could have more floating rows, leading to larger interference from W_2 columns. In addition, if W_2>W_1, the faster oscillation of W_2 can constantly interrupt the oscillation behavior of W_1. An oscillation waveform of a failure case with W_1=20μS and W_2=80μS is shown in Fig. 3.13(b), where the MIT device never switches and the voltage just fluctuates with a small magnitude.

B. 2-Transitor-1-Resistor (2T1R) Array Architecture

To eliminate the sneak path current that causes interference between columns in the crossbar array, a transistor can be added in series with the eNVM device as in conventional 1-transistor-1-resistor (1T1R) array architecture for memory applications. The 1T1R array architecture has been used for performing weighted sum operation with modification on the BL direction, making it to be the input row like the pseudo-crossbar array [60]. Similarly, the WL is in parallel with BL and it controls all the transistors on a row, thus there is no interference if the transistors on the entire row are turned off. However, in 1T1R array, different number of selected rows will affect the total parasitic capacitance on the source line (SL) column, which may hamper the weighted sum accuracy according to Eq. (3.3). The reason for this capacitance variation is due to the transistor drain capacitance, as it can be isolated from the SL column if the transistor is turned off, otherwise it will contribute to the parasitic capacitance of the SL column. To alleviate this effect, we extend the 1T1R array by adding one more transistor adjacent to the existing transistor, constructing a 2-transistor-1-resistor (2T1R) array architecture, as shown in Fig. 3.14. The additional transistor is controlled by the inverting WL signal with its drain floating. In this way, the additional transistor serves as a complementary parasitic capacitance for the SL column. Each cell will contribute one drain and two source parasitic capacitance

independent of WL signal as one of transistors will be turned on with the other one turned off.

With a 2T1R array size of 128×128, the total SL column capacitance is measured to be ~125 fF based on the transistors in a 65nm CMOS technology. Following the same simulation setup as in the previous section, we have simulated the deviation of number of output pulses across the wide range of weight values, and the results show that the maximum deviation is only ~2%, which is a significant improvement over the results in Fig. 3.13(a). Although the 2T1R architecture seems to have a larger overhead in the synapse array area compared to the simple crossbar architecture, it should be noted that the array area is determined by the pitch of the peripheral circuits in the logic design rule. For example, the array cell height should be aligned with the standard cell height of the WL driver, which is basically the height of two transistors. Therefore, the array area overhead with the 2T1R array can be considered negligible.

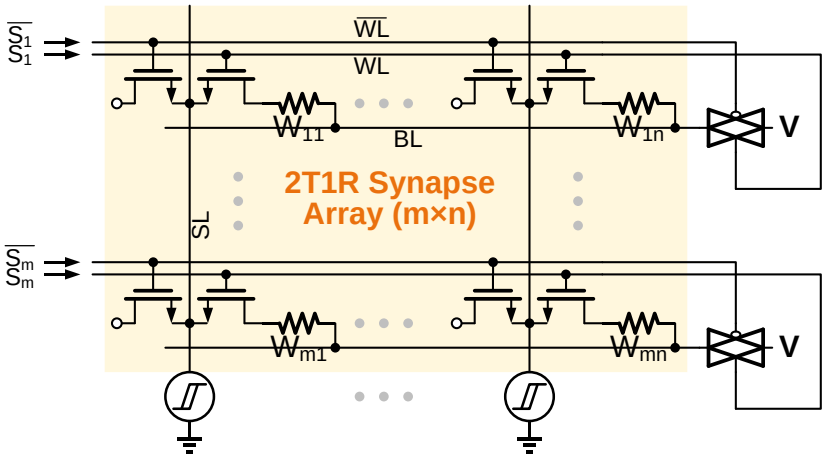


Fig. 3.14 Schematic of 2-transistor-1-resistor (2T1R) array architecture. The transistor in series with eNVM could cut off the interference paths between columns. The other transistor with floating drain helps eliminate the capacitance variation when different number of rows are activated (S_i=V_S). Here the dummy column and the readout circuitry are omitted. © 2016 IEEE.

C. Simulation of Weighted Sum Operation in Array

As the accuracy deviation due to the array architecture is largely resolved, we have to revisit the effect of eNVM weight change to optimize the weighted sum accuracy. Fig. 3.15(a) shows the oscillation frequency as a function of weight similar to Fig. 3.11(b), but with a parasitic capacitance of 125fF as in the 2T1R array. From the algorithm perspective, it is expected that the weighted sum of one column in an 128×128 array should have a maximum value of 128 if all the inputs are 1 (S_i=V_S) and all the algorithm weight values are also 1. On the circuit side, we have to determine the read cycle time of input vector that can translate the oscillation frequency to the desired number of output pulses to match the value from the algorithm. Due to the nonlinearity in Fig. 3.15(a), the read cycle time has to be calibrated at the linear weight region with the corresponding algorithm value to prevent overestimation, since the actual frequency will slightly decrease outside of the linear region. For the array implementation, the calibration should be done with both the actual column and dummy column. Therefore, a better approach is to measure the deviation between the slope of the two curves (in log-log scale) in Fig. 3.15(a), as shown in Fig. 3.15(b). We select two weights with the same deviation that can cancel out each other, and measure the read cycle time required for the corresponding algorithm weighted sum value. In this case, since the weight of real column (70μS) is 5× larger than the weight of dummy column (14μS), we have to calibrate the read cycle time that gives 70μS/2μS=35 pulses, and it is measured to be ~30ns.

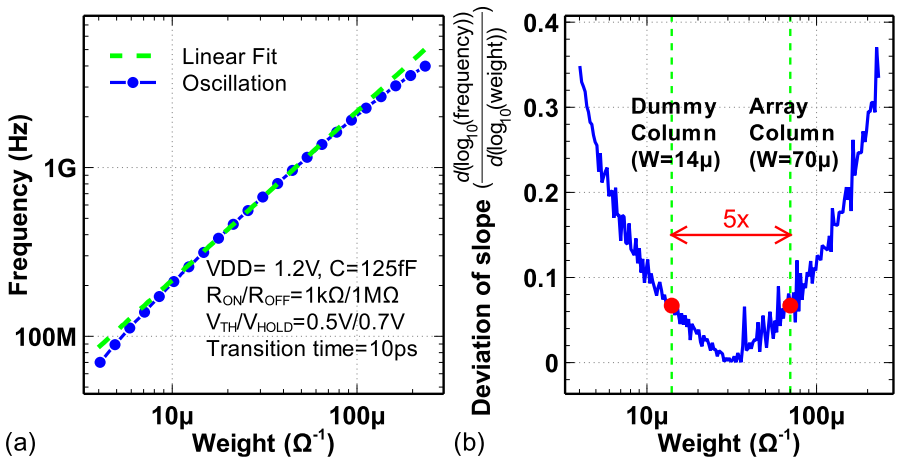


Fig. 3.15 (a) Oscillation frequency as a function of weight at C=125 fF. (b) The deviation between the slope of oscillation frequency and the linear fit in (a). The linear region is centered at W~30μS. To improve weighted sum accuracy, the mapping from algorithm to real weighted sum result should be calibrated in a case where the slope deviation of array and dummy column can cancel out. The 5× means the maximum weight difference between columns. © 2016 IEEE.

Then, we run the Monte Carlo simulation with 12,800 weighted sum tasks in a 128×128 2T1R array based on the calibrated read cycle time. We assume both the input vector and weights are 4-bit values in uniform distribution. As shown in Fig. 3.16, the weighted sum tasks with the calibrated read cycle time ~30ns has only a small weighted sum accuracy deviation (average is ~2.5%). However, if the application can tolerate more accuracy deviation than this, we can accelerate the read process by using a shorter read cycle. If the read cycle is reduced by 2^n times, then the final weighted sum result needs to be shifted by n bits toward the left to match the algorithm weighted sum range. Fig. 3.16 shows a clear tradeoff between the accuracy and the read cycle time. We also simulated the weighted sum tasks with doubled read cycle time (~60ns), however it does not show noticeable accuracy improvement over the 30ns case.

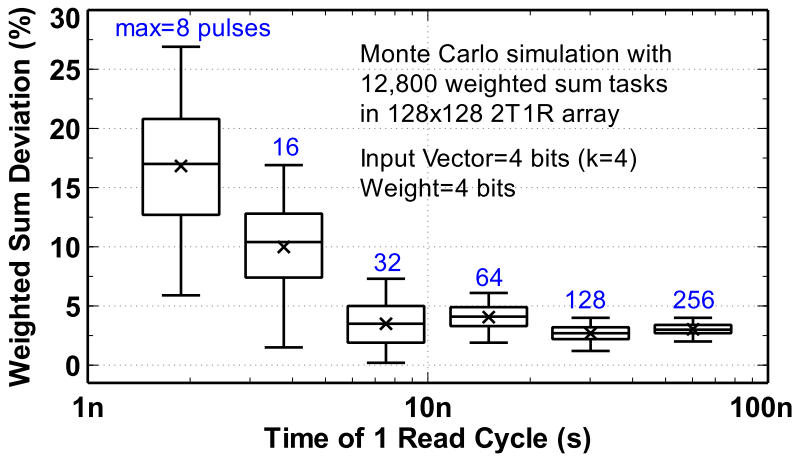


Fig. 3.16 Statistical deviation of final weighted sum accuracy with different read cycle time. As the array row size is 128 and the maximum value of an algorithm weight is normalized to 1, the weighted sum of a column should be 128, corresponding to a read cycle of ~30 ns. The read cycle time can be reduced with a tradeoff of lower accuracy of the final weighted sum. © 2016 IEEE.

Finally, the performance of the proposed oscillation neuron is benchmarked with that of the CMOS neuron [61] at the 65nm technology node. Table 3.1 shows the sub-circuit level benchmark results. To make a fair comparison, we follow the same simulation setup as [61]. The performance is evaluated within 8 integrate-and-fire cycles with eNVM weight to be ~53μS. Despite a ~40% increase in latency, the compact oscillation neuron circuit achieves tremendous reduction in area, energy and leakage power. Table 3.2 shows the array level benchmark results. The synaptic array size is set to be 128×128 and there are 4 pulse cycles for the input vector. In practical array design, multiple columns usually share one neuron to improve the area efficiency. From the array's point of view, the oscillation neuron does not gain much benefit in total area (synapse array area + peripheral neuron area) because the total area is still dominated by the array core. However, the oscillation neuron eventually outperforms the CMOS neuron in latency. As the oscillation neuron is

more compact, the number of columns shared by one neuron can be reduced from 8 to 4, thereby increasing the parallelism.

Table 3.1 Sub-circuit Level Benchmark. © 2016 IEEE.

	CMOS Neuron [61]	Oscillation Neuron	Reduction
Area	11.24 μm^2	0.89 μm^2	>12.5 X
Latency	4.5 ns	6.2 ns	-37.8%
Energy Consumption	1.346 pJ	0.265 pJ	>5 X
Leakage Power	104.9 µW	35.84 nW	~3,000 X

Table 3.2 Array Level Benchmark (1 Weighted Sum Task). © 2016 IEEE.

	Array with CMOS Neuron [61]	Array with Oscillation Neuron	Reduction
Area	36918 μm^2	35571 μm^2	~4 %
Latency	144 ns	99.2 ns	>30 %
Energy Consumption	693.2 pJ	139.5 pJ	~5 X
Leakage Power	1.73 mW	44.12 µW	~40 X

3.4 Summary

In the array design, the selector is used to alleviate the IR drop along interconnects and the leakage power on the unselected cells, without affecting weight update. To further reduce the energy consumption and prevent write disturbance problem in weight update, conventional 1T1R array is modified to be pseudo-crossbar array, with BLs rotated by 90^o to enable weighted sum operation. For the crossbar and pseudo-crossbar array, key neuron peripheral circuits are introduced in detail. To replace the existing complex read circuit, the MIT device has been proposed as an oscillation neuron for the parallel weighted sum operation in the eNVM synaptic array. In this work, we studied the impact of MIT device

parameters and provided design guidelines for future MIT device engineering. To enable weighted sum in large-scale arrays, a MIT device that has large ON/OFF resistance ratio is desired. The feasibility of the eNVM synaptic array with oscillation neurons is also studied. To prevent oscillation interference between array columns, the 2T1R array architecture is preferred over the crossbar architecture at negligible expense of array area. The read cycle is calibrated in the array design to improve the weighted sum accuracy. Monte Carlo simulation of weighted sum tasks shows the tradeoff between the weighted sum accuracy and the read latency. Compared to the CMOS neuron [61], oscillation neuron shows >12.5X reduction of area at single neuron node level, and shows a reduction of ~4% total area, >30% latency, ~5X energy and ~40X leakage power at 128×128 array level, demonstrating its advantage for neuro-inspired computing.

4 NEUROSIM: DEVICE-CIRCUIT-ALGORITHM BENCHMARK SIMULATOR FOR NEURO-INSPIRED ARCHITECTURES

In this chapter, NeuroSim was developed to be a circuit-level macro model that estimates the area, latency, dynamic energy and leakage power to facilitate the design space exploration of neuro-inspired architectures with mainstream and emerging device technologies. NeuroSim provides flexible interface and a wide variety of design options at the circuit and device level. Therefore, NeuroSim can be used by many neural network (NN) algorithms as a supporting tool to provide circuit-level performance evaluation. With NeuroSim, an integrated framework can be built with hierarchical organization from the device level (synaptic device properties) to the circuit level (array architectures) and then to the algorithm level (NN topology), enabling instruction-accurate evaluation on the learning accuracy as well as the circuit-level performance metrics at the run-time of learning. In this chapter, we will demonstrate the use of NeuroSim alone to evaluate the performance of partitioning a large weight matrix into several small SRAM and eNVM arrays. In the next chapter, we will demonstrate the use of NeuroSim to support the learning algorithm for circuit-level performance benchmark.

4.1 NeuroSim Architecture

4.1.1 Overview

NeuroSim is a circuit-level macro model developed in C++ that can be used to estimate the area, latency, dynamic energy and leakage power of neuromorphic hardware accelerators with SRAM and eNVM based architectures to facilitate the design space exploration. The framework of NeuroSim follows the principles of CACTI [96] for SRAM cache and NVSim [97] for NVM. These simulators focus on the design for traditional

memory application, and do not support the memory design for neuromorphic computation. In contrast, NeuroSim is dedicated to support neuro-inspired architectures. The hierarchy of the simulator consists of different levels of abstraction from the memory cell parameters and transistor technology parameters, to the gate-level sub-circuit modules and then to the array architecture including the peripheral circuits. Fig. 4.1(a) shows an overview of the high-level architecture with neuromorphic hardware accelerator to implement NNs. NNs generally require multiple (or deep) layers for better learning performance, where each layer contains the synaptic core and neuron periphery. A synaptic core is specifically designed for weighted sum and weight update. It takes the digital input vector and gives out the weighted sum result in the digital format. Thus the digital communication is used between synaptic cores while any analog computation will just be done within the core only. The synaptic core further consists of the synaptic array and array periphery. The synaptic array (such as Fig. 1.4(a) or Fig. 3.3) is the core unit of weighted sum computation and the array periphery helps transform the results to be the digital format if necessary. NeuroSim supports various digital and analog synaptic cores, as shown in Fig. 4.1(b)-(e). On the other hand, the neuron periphery is responsible for nonlinear activation function and communication from one synaptic core to another. Currently, NeuroSim can implement nonlinear activation function using a SRAM/eNVM array based look-up table (LUT), while it also supports the low-precision activation function such as thresholding with step function. As the circuit implementation of neuron periphery is more flexible and can vary between different NNs, we will only show an example one in later case study.

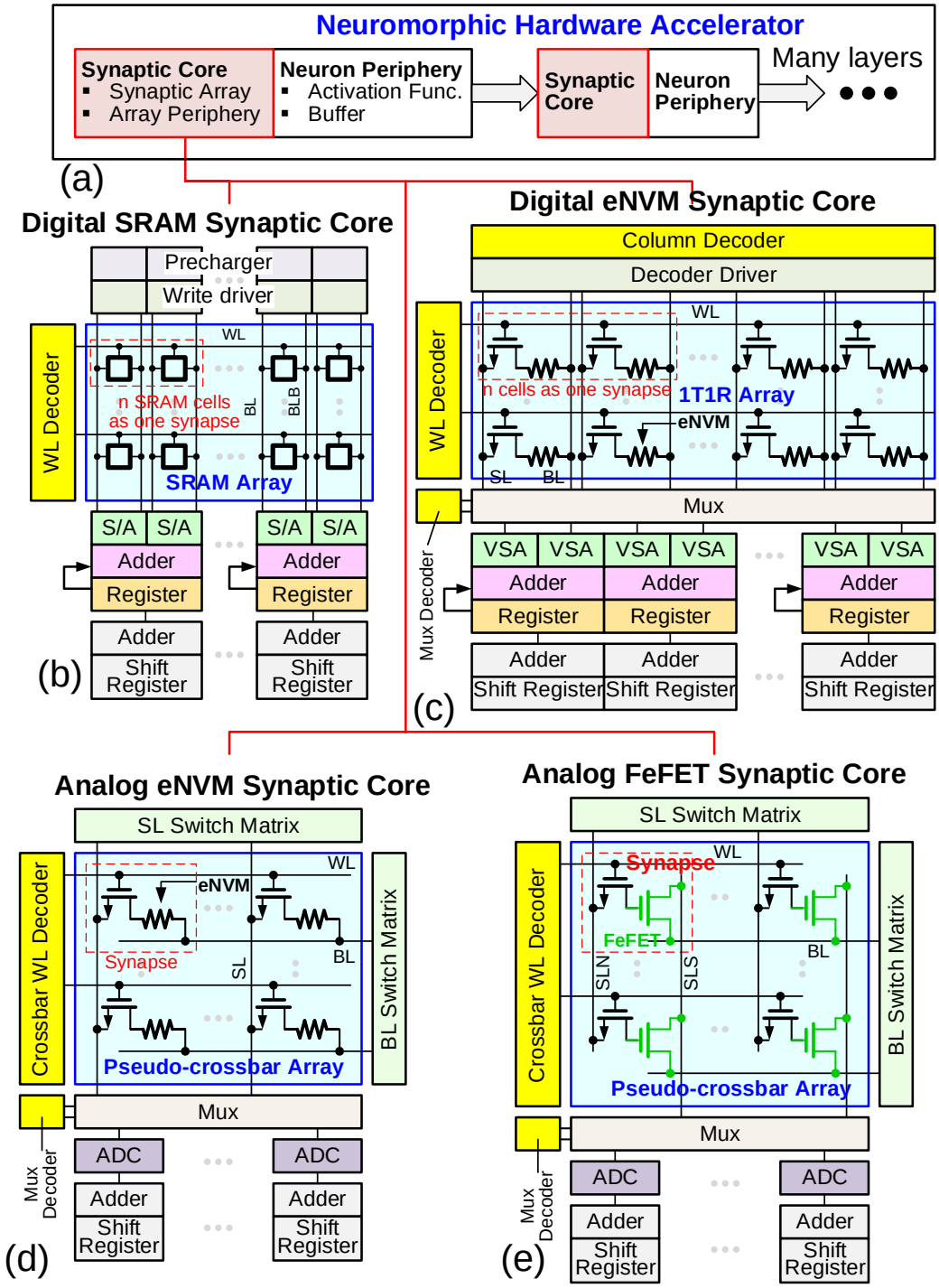


Fig. 4.1 (a) Overview of high-level architecture with neuromorphic hardware accelerator. (b) Circuit block diagram of SRAM synaptic core. (c) Circuit block diagram of digital eNVM synaptic core with 1T1R array. (d)-(e) Circuit block diagram of analog eNVM and FeFET synaptic core with the pseudo-crossbar array, respectively.

4.1.2 Synaptic Core

In this section, we introduce different synaptic core architectures, which are considered to be at one hierarchy level higher than the sub-circuit modules, as they consist of both memory array and peripheral circuits that are closely jointed to form a standalone weighted sum computation unit. Important parameters at this level include synaptic array types and sizes, operating modes of peripheral circuits, and the number of synapses that can be accessed in parallel during weighted sum and weight update, etc.

A. SRAM Synaptic Core

The circuit block diagram of SRAM synaptic core is shown in Fig. 4.1(b). As SRAM cells can only store binary bits, we group multiple SRAM cells along the row as one synapse to represent a higher weight precision. The weighted sum and weight update operation in the SRAM based synaptic core are essentially row-by-row based, which is similar to the read and write operation in a conventional SRAM memory.

In the weighted sum operation, the input vector is encoded using multiple clock cycles to represent its precision. For each row, an input vector bit of 1 means the row will be selected for read, otherwise the row will be skipped. To select a row, the WL is activated through the WL decoder. To access all the cells on the selected row, the BLs are pre-charged by the pre-charger and the write driver in weighted sum and weight update, respectively. After the memory data are read by the sense amplifier (S/A), the adder and register are used to accumulate the partial weighted sum in a row-by-row fashion. To make sure the overflow will not occur during the accumulation, the adder and register need to have a longer bit-width than the weight precision of a synapse. The adder and shift register pair at the bottom performs shift and add of the weighted sum result at each input vector

bit cycle to get the final weighted sum. The bit-width of the adder and shift register needs to be further extended depending on the precision of input vector. If the values in the input vector are only 1 bit, then the adder and shift register pair is not required. For the write operation, new weights will be provided from the input of the write driver. All the cells on the same row can be updated at the same time, thus the weight update operation is also row-by-row based.

B. Digital eNVM Synaptic Core

The circuit block diagram of digital eNVM synaptic core is shown in Fig. 4.1(c). By replacing the SRAM core memory with eNVM without much modification on the whole digital circuit architecture, we potentially get smaller synaptic core area. The way the digital eNVM synaptic core works is very similar to the SRAM one, thus it can just use the traditional 1T1R array as the synaptic array. Similarly, we have to group multiple binary 1T1R cells along the row as one synapse to represent a higher weight precision.

The weighted sum operation in digital eNVM synaptic core is also row-by-row based. After the memory data are read out by the voltage S/A, adder and register will perform accumulation on the partial weighted sum through row by row. One key difference compared to the SRAM synaptic core is the use of Mux. As the cell size in 1T1R array is much smaller, it will not be area-efficient to put all the read periphery circuits underneath the array. Therefore, it is necessary to use a Mux to share the read periphery circuits among synaptic array columns. However, this inevitably increases the latency of weighted sum as time multiplexing is needed because of the sharing. For the weight update, the column decoder can select a group of synapses at a time depending on the design, and the

programming voltages will be provided from the decoder driver. Two phases are required to program the cells to be ON and OFF because they need different WL voltages.

C. Analog eNVM/FeFET Synaptic Core

In NeuroSim, the analog eNVM synaptic core supports two types of the synaptic array architecture: the crossbar and the pseudo-crossbar array architectures, as described earlier in Section 3.2. In this chapter, we will only show the pseudo-crossbar array architecture (Fig. 4.1(d)-(e)). The details of pseudo-crossbar array architecture has been discussed in Section 3.2. Briefly speaking, the crossbar WL decoder is used to activate all the WLs during weighted sum, while activate one WL during weight update. The switch matrix can activate multiple rows or columns at a time, thus it enables parallel voltage inputs of a vector in weighted sum, and can realize the weight update scheme that requires different voltage biasing in selected/unselected rows and columns. To be general, ADC is labelled rather than the read circuit for the reason that other neuron circuit designs (such as the oscillation neuron in Section 3.3) can also be used. Similarly, the adder and shift register pair will perform shift and add on the weighted sum results of all input vector bit cycles to obtain the final weighted result. On the other hand, the analog FeFET synaptic core is only different than the eNVM one in the synaptic array structure, as shown in Fig. 4.1(e). It also has an access transistor for each cell to prevent programming on other unselected rows during row-by-row weight update. As FeFET is a three-terminal device, it needs two separate SLs for the weighted sum (SLS) and weight update (SLN), respectively.

4.1.3 Transistor and Cell Models

At the device level, NeuroSim is featured with various design options in transistor technologies and memory cells. The transistors can be configured to be high-performance

(HP) or low-standby-power (LSTP) type with different technology nodes from 130 nm down to 7 nm, where FinFET is used at 14 nm and beyond. The transistor models are calibrated based on Predictive Technology Model (PTM) [98]. Compared to industry transistor models, PTM is available to the public and it has a wide range of technology nodes, which is suitable for the design space exploration at the early design stage. Important parameters in transistor models include device W/L, the operating voltage (V_{DD}), threshold voltage (V_{TH}), gate and parasitic capacitance, and NMOS/PMOS saturation/off current density across different temperatures, etc. In particular, V_{TH} is extracted at the gate voltage (V_{GS}) where the drain current density (J_{DS}) is 300 nA/μm under V_{DS}=V_{DD}. In bulk MOSFET, the total gate capacitance is the sum of ideal, fringe and overlap gate capacitance, while the total drain capacitance includes the capacitance in the diffusion region from junction to bottom, channel and the other three sidewalls. Based on these parameters, the area and intrinsic RC model of standard logic gates (INV, NAND, NOR, transmission gates, etc) can be calculated analytically thus the circuit-level performance metrics of each sub-circuit module can be estimated.

The design of SRAM and eNVM cells in NeuroSim is also flexible. We use conventional 6T SRAM (extendable to 8T SRAM), where all transistors' W/L can be adjusted. The transistor technology defined for other digital circuits also applies to SRAM's transistors. On the other hand, eNVM cells have parameters such as max/min conductance, read/write voltage and pulse width, number of conductance states (weight precision) and I-V nonlinearity degree, etc. These parameters play an important role in the array-level performance and will further affect the peripheral circuit design in the synaptic core.

4.1.4 Customization

NeuroSim is designed to have as much flexibility as possible for customization, without increasing the workload for the users to do so. For the customization of modules, they can be discussed in three different levels as below:

A. Architecture Level

We define the architecture level to be the level between the algorithm and sub-circuit module level. Synaptic cores are at this level. The design of synaptic cores is not limited to the four types discussed in Section 4.1.2. They are more like standard templates and the users are always encouraged to build their own synaptic cores (or other computation units) following the hierarchical structure in NeuroSim. For a more complex design, the users may need to insert one or more hierarchical layers that uses the synaptic cores as building blocks, for example, when it comes to the partition strategy on the synaptic cores for performance optimization [64].

In the top hierarchical layer, the user needs to make sure the interface is well defined and has the ability to communicate with the algorithm side. Additionally, considering the cases where the NN simulator has its own circuit/device-level configurations (e.g., the users have their own embedded synaptic array in the neural network), a hierarchical layer at the architecture level of NeuroSim should be able to link its configurations with the ones in the upper layer and provide this link to the lower layer as well. In this way, the configurations can be shared among all the layers of NeuroSim and the NN simulator, rather than just being duplicated to each of them. Therefore, if some of the configurations are modified in either NeuroSim or the NN simulator, these modifications will also be reflected in the other one. This is necessary in some design optimization cases where

complex interdependence of the configurations between these two simulators is unavoidable.

B. Sub-circuit Module Level

The sub-circuit modules included in NeuroSim are mostly shown in Fig. 4.1(b)-(e). If the users fail to find a sub-circuit module that serves their needs, they can create a new module and integrate it into NeuroSim. To do this, the user need to figure out the circuit components in terms of the standard logic gates, and develop the performance estimation model by either using analytical equations or simply providing the performance values that are obtained from SPICE simulation or measured from real hardware. The detailed structure of a sub-circuit module will be discussed in the next section.

Sometimes, the users can just introduce a new mode in the existing module, without bothering to make similar modules with only minor modifications. For example, the decoder module currently has 4 modes (row/column + regular/Mux). As shown in Fig. 4.1(b)-(e), it can be used as a regular row decoder (WL decoder) or a Mux row decoder. The difference of regular and Mux decoder lies in the output buffer, where the Mux decoder has the enable function to disable the Mux's connectivity. It should be noted here that the crossbar WL decoder in Fig. 3.6 is just the combination of the decoder and the follower module. We think it is better to package the follower as an individual sub-circuit module instead of a new mode in the decoder due to its complexity and design flexibility. But no matter which way the users prefer, the sub-circuit modules need to be clearly defined in the interface, reducing the complexity and efforts to connect them at the architecture level.

C. Device Level

As mentioned in Section 4.1.3, the device level covers from the transistor technology and memory devices up to the RC model of standard logic gates. The transistor parameters are pre-defined with different technology configurations. In the simulation, the only thing we need to do is to select a technology configuration. For memory devices, there may be cases where the devices need to be configured at the architecture or algorithm level if the users consider the memory device parameters are part of the design parameters in the NN (e.g., the weight precision of synapses required in the algorithm determines the number of conductance states in eNVM devices in NeuroSim), or if the users prefer to introduce the device properties from elsewhere to NeuroSim.

Regarding the customization at the device level, we list a few possible situations as below:

- *New operating mode of transistor*: Currently NeuroSim supports HP or LSTP transistors. If the users want to add a new operating mode, they have to provide relevant transistor parameters, such as V_{DD}, V_{TH}, gate and parasitic capacitance (per unit gate length), NMOS/PMOS saturation/off current density across different temperatures, etc.
- *New technology node*: The users have to provide the parameters for the existing operating modes (HP and LSTP) in the new technology node, and it should be noted that FinFET is used at 14 nm and beyond. There are a few differences in the parameters and layout of bulk MOSFET and FinFET.
- *New transistor technology*: If the users propose to explore the design with novel transistor technologies other than the conventional MOSFET (e.g. tunnel-FET,

negative-capacitance FET, etc.), the users need to come up with an equivalent transistor model and provide relevant parameters. It would not be recommended to consider a new transistor with its structure completely different than MOSFET, because the design principle of gate-level logics is still based on conventional CMOS technology.

- *New memory device*: To make the most effective use of NeuroSim, the users are strongly encouraged to introduce new types of memory devices (especially analog eNVM) for performance benchmark. In NeuroSim, conductance states of analog eNVM devices is assumed to be tuned by the number of voltage pulses. Equations on dynamic performance metrics need to be modified if the new device uses a different programming strategy.

4.1.5 Usage of NeuroSim

As a circuit-level macro model, NeuroSim does not incorporate the learning algorithms, and it estimates the circuit-level performance of a synaptic core by taking either the data patterns of the input vector and weight matrix from the algorithm, or the average parameters of these patterns to have a good approximation of performance evaluation. For the latter one, for example, we can assume the activity of the input vector is 0.5 (50% 1 and 0 in the vector), but not exact the data pattern of the input vector. At the device level, it may assume an average conductance of the synaptic devices and an average number of pulses for the weight update operation in analog eNVM synaptic core, but not the conductance pattern or programming pulse information for the entire synaptic array. To illustrate how NeuroSim works, we have considered three different usage scenarios as shown in Fig. 4.2, and they are described below.

A. NeuroSim for Architectural Performance Estimation

In this scenario, NeuroSim alone is used to estimate the circuit-level performance metrics of neuro-inspired architectures. As mentioned earlier, a synaptic core in NeuroSim takes weighted sum or weight update instruction with specified data pattern or average parameters to calculate the circuit-level performance per instruction, and it will quickly show the performance breakdown results from the synaptic core to its subcomponents. Thus, using NeuroSim alone is very handy for quick circuit-level performance benchmark without the need to run a full SPICE simulation.

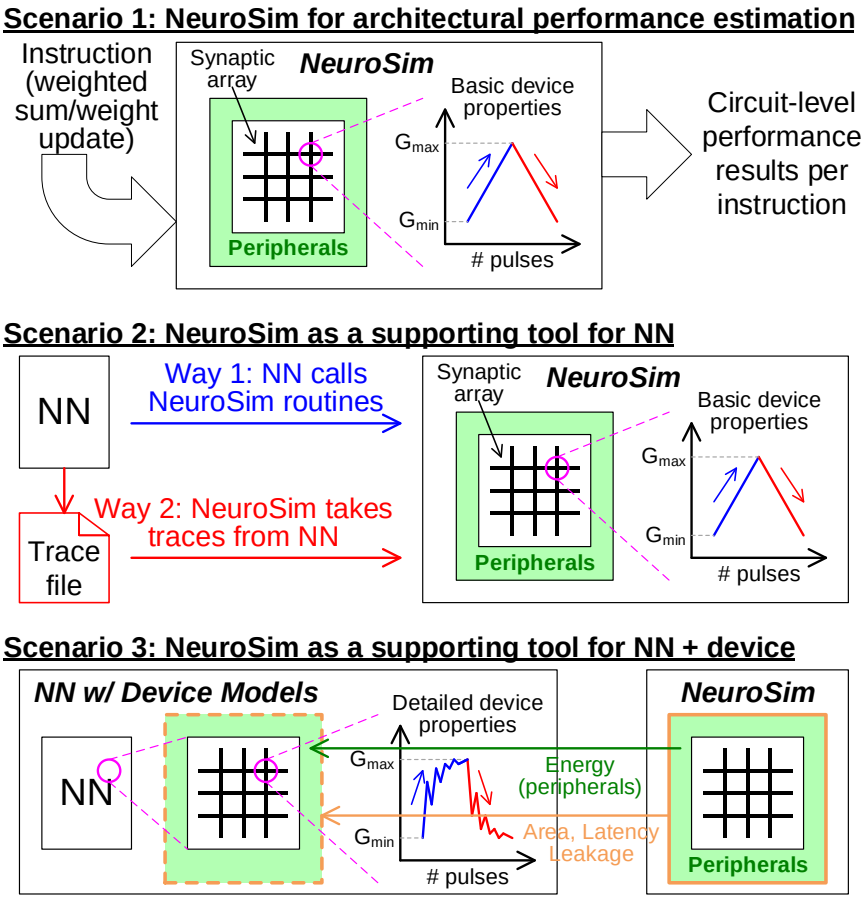


Fig. 4.2 Different usage scenarios for NeuroSim. © 2018 IEEE.

B. NeuroSim as a Supporting Tool for NN

In this scenario, NeuroSim is used as a supporting module to provide circuit-level performance estimation for NN simulators, which is helpful for NN researchers to explore the design space of NN architectures at early design stage. This scenario can be done in two ways. The first way is that the NN simulator calls NeuroSim routines at every weighted sum and weight update operation, which makes the performance evaluation instruction-accurate. However, this approach may not be applicable if the NN simulator is not compatible with C++ NeuroSim interface. The second way is that NeuroSim takes the trace of data patterns that are recorded during the run-time of the NN simulator, which is essentially a trace-based simulation. This approach is much simpler than the first one and has no limitation on the platform, but it is much less efficient and requiring more simulation time to fetch the data from a trace file as it can be very large in size.

C. NeuroSim as a Supporting Tool for NN+Device

This scenario is very similar to the second one, except the difference that part of the circuit-level performance estimation that NeuroSim provides may only be on the array peripherals, because the NN simulator has already incorporated a more complex synaptic array and device behavioral model. In this example, the NN simulator can estimate the energy consumption of the synaptic array more precisely and efficiently with its device model, thus NeuroSim is only responsible for the energy consumption on the array peripherals. For other performance metrics, NeuroSim still provides the estimation based on the whole architecture because they are more at the scope of architecture or circuit level. In fact, several works [39, 59, 99] published by the device community have demonstrated such an NN+device framework for evaluation of learning accuracy with various synaptic

array and device characteristics, but they cannot address the circuit-level performance. Thus, we believe that NeuroSim can be a good supporting tool to fill up the gap between the algorithm and device for these frameworks as well as enabling the co-optimization of circuit and device for the device engineers.

4.1.6 Limitations of NeuroSim

Despite that NeuroSim features a wide variety of design options for the usage/support of circuit-level performance benchmark in neuro-inspired architectures, there are still several aspects that NeuroSim has not incorporated yet, leaving the room for future improvement. These include 1) the ability to automatically map NN to several partitions of synaptic core and neuron periphery; 2) the interconnection, routing and network topology of synaptic cores at the architecture level; 3) the overhead of off-chip memory access; 4) a complete set of modules in support of machine learning NNs, such as convolutional NN (CNN) or recurrent NN (RNN); 5) the ability to adapt other neural network types, such as spiking NN (SNN).

For 1), currently the users have to manually instantiate the synaptic cores by providing the synaptic array sizes that equal to the weight matrix sizes of the algorithm, thus only custom design is supported. The automatic mapping of the weight matrix sizes to arbitrary synaptic array sizes is to be developed for reconfigurable design. For 2), the overhead of latency and energy due to interconnection or routing between synaptic cores may become noticeable as the synaptic array size scales up. This will be the issue to solve after 1) is done. For 3), the overhead of off-chip memory access cannot be ignored if only part of the weights are stored on-chip. NeuroSim may have to be integrated with some third-party C++ DRAM modules (e.g. DRAMSim2 [100]) to take this overhead into account. For 4),

currently NeuroSim partially supports CNN but more modules are still under development. For example, it has the module for the convolutional kernel and average pooling but no maximum pooling or batch normalization. On the other hand, RNN requires a different type of synaptic core that can achieve recurrent connections, which is not included in NeuroSim yet. Therefore, the users may have to bring their own design to NeuroSim if there is no existing module available there. For 5), the synaptic core and other sub-circuit modules in NeuroSim are designed to support the key operations in machine learning NNs in a synchronous fashion. Event-driven asynchronous SNN works in a different way that the key operations rely on the timing between spikes to encode information, which NeuroSim cannot implement with its current form. Considering the limitations listed above are more at the algorithm and architecture level, at the current stage we would like to position NeuroSim as a circuit-level macro model that is most suitable for the device engineers to quickly benchmark various synaptic devices and neuro-inspired architectures with a basic NN algorithm.

4.2 Performance Estimation Models

As a circuit-level estimation tool, NeuroSim is beneficial in exploring the design space of neuro-inspired architectures at early design stage. Typical circuit-level performance metrics include the area, latency, dynamic energy and leakage power. Compared to the time-consuming SPICE simulation, NeuroSim provides fast estimation of the performance metrics using analytical models or pre-defined values provided by the user with reasonable accuracy. In this section, we introduce the performance estimation models in NeuroSim.

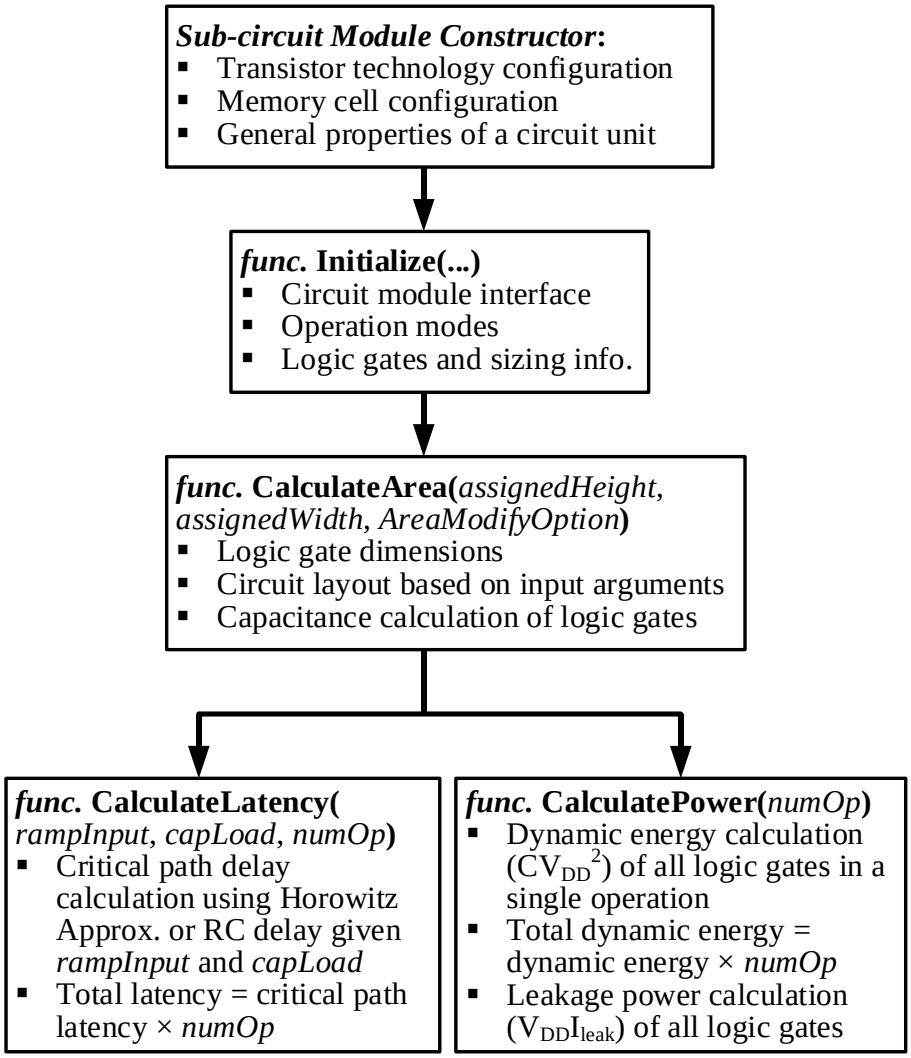


Fig. 4.3 Software execution flow of sub-circuit module functions to estimate the performance metrics. © 2018 IEEE.

4.2.1 Model Setup

Fig. 4.3 shows the basic execution flow of sub-circuit module functions to obtain the performance results of sub-circuit modules. Before performance estimation, the sub-circuit module has to be constructed and initialized. Upon constructed, the sub-circuit module links its transistor and memory cell configurations with the ones from the upper level, and the general properties of a circuit unit, such as the layout height and width, read/write

performance metrics, etc., are also declared. In the initialization step, functionality of the sub-circuit module is outlined. The module interface, operating modes and logic gates with sizing information (transistor W/L) need to be defined in this step. In general, we pre-define the transistor W/L for the logic gates in sub-circuit modules according to the drivability that are needed. Specifically, we design the transistor W/L for the transmission gates that drives the array, such as the ones in the decoder driver, switch matrix and Mux of the eNVM based synaptic array. We consider the worst case where the synaptic array has all its eNVM at the lowest resistance, and calculate the maximum effective resistance of the transmission gates (R_{TG}) under a coefficient of IR drop tolerance (*IR_DROP_TOL*):

$$R_{TG} \leq R_{WORST\_ROW/COL} \times IR\_DROP\_TOL \tag{4.1}$$

where $R_{WORST\_ROW/COL}$ is the total resistance of all eNVM cells in parallel in a row or column depending on either the transmission gate connects to the array row or column. By setting up a small *IR_DROP_TOL* (0.1 by default), we can make sure the input voltage can be delivered into the array without noticeable degradation in most cases.

At the architecture level, the flow is similar to the one for sub-circuit modules. We show the execution flow of a synaptic core as a basic example of architecture in Fig. 4.4. In the initialization step of synaptic core, initialization of all sub-circuit modules that belong to this synaptic will be performed. The same organization is also applied for the rest of performance estimation functions. In this way, a well-defined nested hierarchy from sub-circuits to architectures can be constructed, enabling bottom-up level-by-level performance estimation.

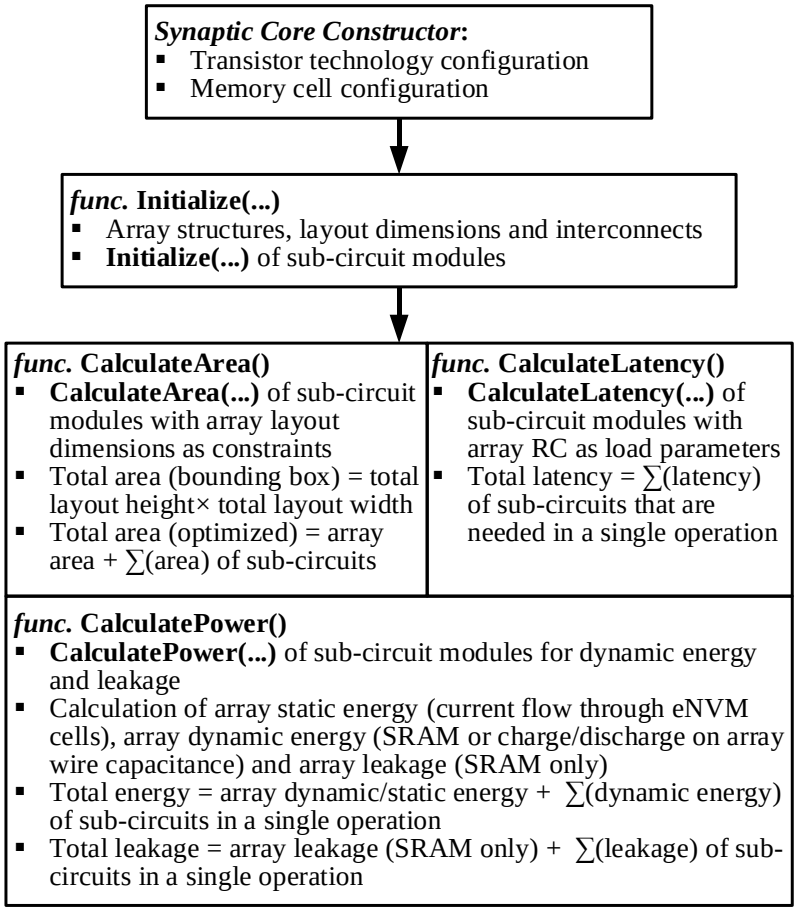


Fig. 4.4 Software execution flow of performance estimation functions at the architecture level (a synaptic core for example). © 2018 IEEE.

4.2.2 Area Estimation

Once the transistors' sizing in each logic gate is known, the logic gates' layout height or width can then be calculated given the other dimension fixed. For example, if the driver has a large transistor W/L and its layout height is constrained by the array row pitch, then NeuroSim will try to find the minimum layout width of the driver that can accommodate this W/L. Typically transistor with large W/L needs be folded, which makes the layout width a quantized value. Thus, even if the layout width of a logic gate is given or constrained, NeuroSim will still adjust the width by rounding it down to the nearest

quantized value when calculating the layout height. In FinFET, the area estimation model becomes a little bit different because the number of fins need to be an integer.

In general, we use the same cell layout height for most of the logic gates in sub-circuit modules, and calculate its cell layout width depending on its transistor W/L. For the synaptic array, the layout dimensions of a memory cell can be pre-defined by the user. If the transistor size of the 1T1R or pseudo-crossbar array is estimated (using the same method as Eq. (4.1)) to be larger than the pre-defined memory cell size, NeuroSim will report an error and request a larger pre-defined size. Considering the array row or column pitch may be smaller compared to the peripheral circuits, NeuroSim also provides an option to relax the memory cell size to match with the minimum layout dimensions of a logic cell. This may increase the area efficiency as well as the total area of synaptic core, but it can prevent some extreme cases where the synaptic array only has a few rows or columns that cannot even accommodate a single periphery circuit unit.

After the synaptic array dimensions are determined, NeuroSim will estimate the layout dimensions of sub-circuit modules. There are three input arguments for the area estimation function of sub-circuit modules, as shown in Fig. 4.3. The first two arguments, *assignedHeight* and *assignedWidth*, are the constraints on the layout height and width, respectively. If one of them is provided, the logic gates at the same stage may need to be placed in multiple rows or columns based on this constraint, and the other dimension can then be estimated to obtain the total area. If neither of these two arguments is provided, there will be no constraint on the layout dimensions and the logic gates will be placed in the most straightforward way for total area estimation. The third input argument,

AreaModifyOption, can be specified for special adjustment of area after the area estimation, which has the following options:

- *NONE*: This option is the regular one, indicating no further adjustment after the area estimation. When choosing this option, the users have to make sure at most one of the first two input arguments *assignedHeight* and *assignedWidth* can be specified.
- *MAGIC*: In this option, the logic gates are pre-placed in the most straightforward way just for quick estimation on the total area. Then, if either of the two constraints *assignedHeight* or *assignedWidth* is given, the other dimension can be obtained by simply dividing this total area with the given constraint. It is assumed that the layout of sub-circuit module can be "magically" folded into any shape while conserving its total area, guaranteeing no waste of area. This option is designed for simple estimation because it does not need to consider the folding of circuit, but it will give the most optimistic estimation result. This option also does not allow both input constraints to be specified.
- *OVERRIDE*: In this option, the estimated layout dimensions will be just overridden by the input arguments *assignedHeight* and *assignedWidth* for the total area, thus both arguments need to be provided. This option is designed for the users to provide their own layout dimensions, or for the cases where both layout dimensions need to be constrained.

In the sub-circuit module's area calculation function, the capacitance at logic gate level is estimated at the last step (Fig. 4.3) because the total drain capacitance is dependent on

the layout structure of logic gate. For example, logic gates with different number of folding have different area and sidewall length of diffusion region.

At the architecture level, NeuroSim provides two different total area estimations, as shown in Fig. 4.4. The first one is to estimate the bounding box area that encloses the entire layout of architecture, which is the total box height × total box width. The other one is to directly sum up the area of array and sub-circuit modules, which may be optimistic but it actually reflects the real case where the layout is always optimized to save chip cost. For the area results in this and the next chapter, we use the latter one (the optimized one).

4.2.3 Latency Estimation

Once the capacitances at logic gate level are all known, the latency and dynamic energy consumption can then be estimated based on RC analyses. We follow the same methods of estimation in CACTI [96] and NVSim [97]. For digital logic gates, the latency is defined as the time required for the output voltage to reach the switching voltage threshold after the input voltage reaches it. We use Horowitz equation to calculate the latency in digital logic gates:

$$\text{Latency} = \tau_f \sqrt{\ln(v_s)^2 + \frac{2}{rampInput \times \tau_f}\beta(1-v_s)} \tag{4.2}$$

where v_s is the normalized switching voltage threshold (typically 0.5). *rampInput* is the input voltage ramp rate, and 1/*rampInput* represents the rise time of the input voltage signal. $\beta=1/(g_m R)$ is the reciprocal of the normalized input transconductance g_m times the output resistance R. $\tau_f=RC$ is the total RC time constant at the output node (assuming a step input), which not only includes the intrinsic output RC time constant of an individual logic gate, but also counts the input capacitance of the logic gates at the next stage. If the output node

is connected to the array, an equivalent lumped RC model of the total wire resistance and capacitance will be involved in the calculation of output RC time constant. After the latency estimation, the output ramp rate of digital logic gates *rampOutput* will also be evaluated:

$$rampOutput = (1\text{-}v_s)/Latency \tag{4.3}$$

which can be provided as the *rampInput* to the next stage of the digital logic gate. For transmission gates used to pass analog voltage signals, we use 2.3*RC* (0-90% voltage rise time) instead of Eq. (4.2) to estimate the latency. The latency estimation at all levels always considers the worst-case scenario. For example, the worst-case input pattern for a NAND logic's evaluation to be 1 is when only one input is 1 because there is only one PMOS pulling up the output node. Under the worst-case input pattern, the latency of a sub-circuit module can then be obtained by summing up the latency of each logic gate along the critical path. Generally, there are three input arguments to the latency calculation function of a sub-circuit module, as shown in Fig. 4.3. *rampInput* determines the voltage ramp rate to the input of the sub-circuit module. *capLoad* is the load capacitance at the output node of sub-circuit module. *rampInput* and *capLoad* are required for the critical path latency calculation. The third argument, *numOp*, is the number of repeated operations considered in the latency calculation, which is designed for the convenience of the higher levels that may need multiple times of access to a single sub-circuit module. The total latency of a sub-circuit module can then be regarded as the critical path latency multiplied by *numOp*.

At the architecture level, the total latency can be calculated as the sum of latency of the sub-circuit modules, as shown in Fig. 4.4. For the weighted sum operation of an eNVM synaptic core, the array RC is considered as the load parameters for the sub-circuit modules

that drives the array. For the weight update operation of an eNVM synaptic core, the latency of device weight tuning is included in the latency calculation of switch matrixes. This write pulse information does not need to be specified as input arguments because it has been already known by all sub-circuit modules upon constructed (Fig. 4.3).

4.2.4 Power Estimation

In the power estimation function of sub-circuit modules, the dynamic energy consumption and leakage power are calculated, as shown in Fig. 4.3. Dynamic energy consumption tells how much of the energy is consumed due to charge/discharge of the capacitance during circuit operation, which is expressed as CV_{DD}^2. Since all the capacitances at logic gate level are known, the dynamic energy consumption of sub-circuit module can then be calculated by summing up the CV_{DD}^2 at all nodes. Similarly, if the input argument *numOp* is given, the total dynamic energy consumption in a number of operations can be obtained.

In eNVM synaptic array, the energy consumption is mainly static energy consumption (i.e. the current flow through eNVM cells), as shown in Fig. 4.4. The energy consumption on the selected analog eNVM cell at weight increase/decrease phase can be simply written as:

$$E_{\text{cell}}=V_{\text{W}}^2 GNT_{\text{PULSE}} \tag{4.4}$$

In Eq. (4.4), G is the conductance of a cell. V_{W} is the write voltage for weight increase/decrease. N is the number of applied write pulses and T_{PULSE} is the pulse width. Besides the eNVM cell, the dynamic energy consumption on the array wire capacitance as well as SRAM cells (for SRAM architecture) will also be calculated. Then, the total energy

consumption for a synaptic core can be estimated as the sum of the dynamic/static energy consumption of array and the dynamic energy consumption of sub-circuit modules.

On the other hand, leakage power represents the power consumption due to subthreshold leakage current (I_{leak}) in the transistor channel when the transistor is turned off. The simplest form of expressing the leakage power is $V_{DD}I_{leak}$. For a simple logic like INV, I_{leak} is just the average of NMOS and PMOS off current (obtained from the transistor technology configuration). For a NAND or NOR logic that has more than one input, I_{leak} will be the PMOS or NMOS off current multiplied by the number of inputs, respectively, for the worst case. However, it is preferred to estimate the leakage current based on the average case. Thus, an additional pre-defined ratio will be applied to the leakage power calculation result. For example, the leakage of a NAND3 can be expressed as:

$$Leakage_{NAND3}=V_{DD}I_{off,PMOS}\times 3\times AR\_LEAK_{NAND3} \quad (4.5)$$

where AR_LEAK_{NAND3} represents the average ratio for leakage current in a NAND3 logic. In the synaptic array, the total leakage power will be simply the sum of leakage of SRAM cells (for SRAM architecture) and all sub-circuit modules, as shown in Fig. 4.4. eNVM cells do not need power to maintain their data thus they do not have leakage.

In fact, since leakage power does not have to do with the capacitance in the estimation model, the power estimation function can be directly called after the initialization step without going through the area estimation step if the users only want to estimate the leakage power. It should also be noted that there is no execution order for the performance estimation functions at the architecture level, as shown in Fig. 4.4. This is unlike the flow of sub-circuit modules in Fig. 4.3, where all capacitances need to be calculated in the area

estimation function before they are ready to be used in the latency and power estimation functions.

4.2.5 Validation

NeuroSim offers a wide variety of design options for benchmarking neuro-inspired architectures. Being the essential bases for the entire simulation framework, the parameters in sub-circuit modules, memory cell and transistor models should be accurate enough to support the validity of NeuroSim. In such context, we have performed SPICE and layout-level calibration of sub-circuit modules to validate the analytical models. As mentioned in Section 4.1.3, the transistor model parameters are calibrated based on PTM. The area estimation, including logic gates and sub-circuits, is based on generic design rules. As shown in Fig. 4.5, we have calibrated the area estimation of an analog eNVM synaptic core with an array size of 256×256 at 45 nm technology node by comparing to its layout using FreePDK45 process design kit [101]. As is shown in the layout, the peripheral circuits (i.e. switch matrix) take substantial area due to the requirement of relaxing W/L for transmission gate for minimizing the IR drop to maintain good accuracy in the analog computation in the synaptic array. The entire layout area is measured to be 15,810 μm^2, with a cell size of 0.0324 μm^2 (4F×4F), while the area estimation by NeuroSim (optimized) is 15,454 μm^2, achieving an error rate of -2.5%.

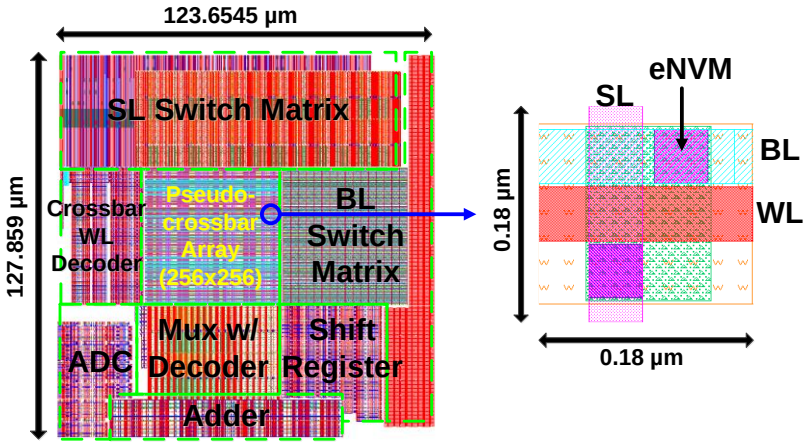


Fig. 4.5 Example layout of the analog eNVM synaptic core (256×256 array size) at FreePDK 45 nm. © 2017 IEEE.

For latency, dynamic energy and leakage power consumption, we pick the representative modules for validation, such as the decoder, adder, Mux and switch matrix. As shown in Fig. 4.6, we calibrated the analytical equations in these performance estimation models at different synaptic array sizes from 8×8 to 256×256 with SPICE simulation based on PTM at 22 nm, 32 nm and 45 nm. In Fig. 4.6, the latency of the decoder is more like a staircase function with respect to the array size. This is because the decoder has two stages and every two address bits will be pre-decoded, thus the decoder structure will have less changes from 2^{N-1} to 2^{N} address bits where N is an even number. On the other hand, the latency of Mux and switch matrix does not increase with larger array size, because all the signal paths are independent and parallel. In Fig. 4.6, the leakage power of Mux is not shown, because it only has transmission gates where the subthreshold leakage current does not exist and the gate leakage current can be negligible. In Fig. 4.6, the average absolute error rates of the sub-circuit modules at these technology nodes are ~14.86%, ~10.51% and ~13.96% for the latency, dynamic energy and leakage power, respectively. The validation results are reasonably accurate considering these performance metrics are

modeled by simplified analytical equations as described earlier in Section 4.2, which we believe is sufficient for a quick estimation of the circuit-level performance at early design stage.

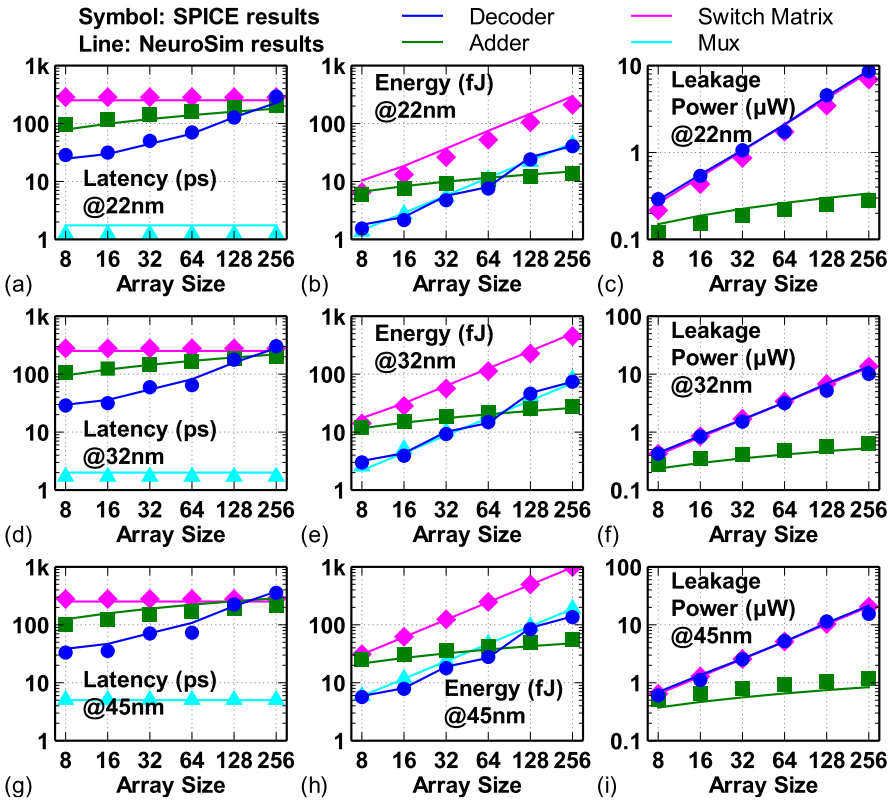


Fig. 4.6 Validation of latency, dynamic energy, and leakage power on main circuit modules (decoder, switch matrix, adder, mux) with different synaptic array sizes at 22 nm, 32 nm and 45 nm technology node. © 2017 IEEE.

4.3 Case Study by Using NeuroSim: Synaptic Array Partitioning

The neural network generally consists of a massive number of synapses that connect between groups of neurons, thus the weight matrix size is large. For instance, unsupervised sparse coding algorithm needs a dictionary array size 100×500 to achieve reasonable learning accuracy [56]. For deep convolutional neural network (CNN), the number of synapses required for the convolution process of the first layer could reach to 121×3025 if

all the kernels are grouped into one array, and the last fully connected classification layer could reach to 2048×2048 [102]. If such large weight matrix is stored on the digital SRAM or analog eNVM arrays, accessing to such architecture may be slow and consumes a lot of energy. Partitioning the array architecture into multiple smaller synaptic arrays is then attractive to improve the overall system performance with an increased computation parallelism. On the other hand, an excessively large number of small arrays is highly area inefficient. Therefore, we analyze the problem of how to efficiently partition the weight matrix using the SRAM and analog eNVM based synaptic cores in this case study. With NeuroSim, we investigate the partition strategy required for performance optimization and its associated trade-offs and overhead.

4.3.1 Partition Scheme and Simulation Setup

As shown in Fig. 4.7(a), we propose to partition the large synaptic array into N×N small arrays in a hierarchical fashion. The partitioning could speed up the weight update operation as the weight elements in different small arrays can be updated in parallel. For the weighted sum operation, the vector and matrix are distributed into these array partitions and computed in parallel, but the results from all small arrays must be collected and summed up. We use multi-stage adders and registers (A&Rs) to obtain the final weighted sum. As shown in Fig. 4.7(b), the summation flow is similar to a binary tree structure for each array column. Each A&R is placed between two small arrays, and the results will be added and passed toward the center A&R of each array column stage by stage. The circuit block diagram of A&R is illustrated in Fig. 4.7(c). The A&R consists of multiple adders and registers depending on the number of adders and read circuits in a SRAM and eNVM

array, respectively. It should be noted that the adders have 1 bit increment in the bit-width stage by stage to account for the summation overflow.

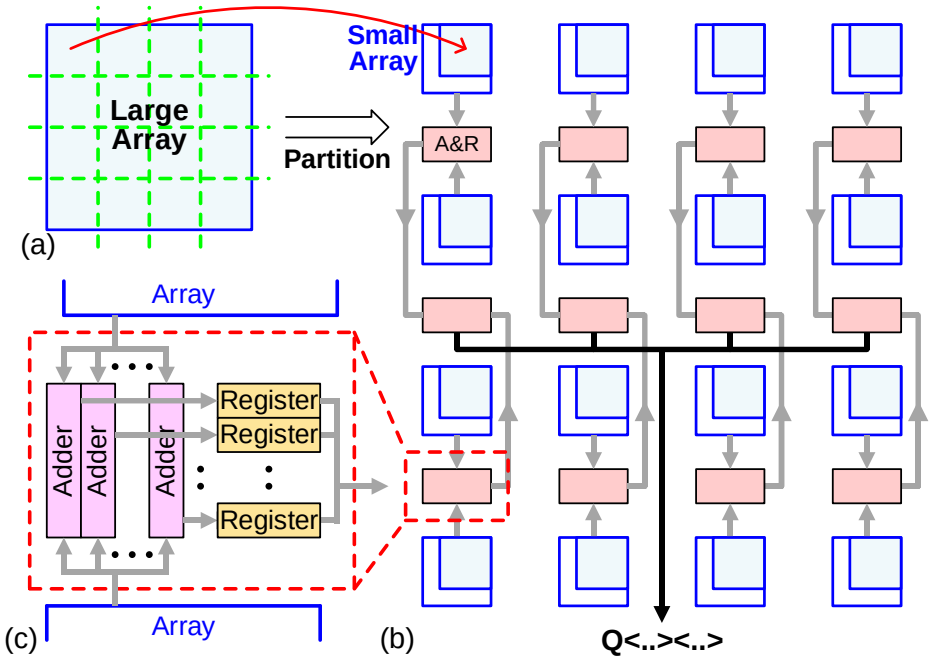


Fig. 4.7 (a) Large synaptic array can be partitioned into small ones and form multiple arrays. (b) In the macro level, multi-stage adders and registers (A&R) are shared between small arrays to accumulate the partial weighted sums from all small arrays. (c) The circuit block diagram of A&R. © 2016 IEEE.

The summary of simulation parameters is listed in Table I. We consider the activity factor in the weighted sum and weight update operation for both synaptic cores. For example, if the read activity for rows is assumed to be 50%, it means 50% of the rows will be read out. In the eNVM synaptic core, we limit the sharing of read circuit to be 8 columns per read circuit to preserve the height/width ratio in the layout to be <~5. In the weight update operation, we assume the weight of each cell is updated by 8 levels in average, which requires 8 write pulses and each pulse is 5 ns. Although sub 10-ns pulse write has

not been demonstrated yet in any synaptic device, we think 5 ns is the best expected pulse width in eNVM for digital memory application.

Table 4.1 Simulation Parameters. © 2016 IEEE.

Parameters	**Values**
Read activity for rows/columns	50% / 100%
Write activity for rows/columns	50% / 50%
Number of bits for the input vector	4
Number of bits per weight element	4
Technology node (F)	32 nm
Clock frequency	2 GHz
6T SRAM cell area	146 F^2 (F= tech node)
Pseudo-crossbar eNVM cell area	16 F^2 (F= tech node)
eNVM resistance	100 kΩ – 10 MΩ (Avg: 1 MΩ)
eNVM read/write voltage	1 V / 2 V
Number of columns per eNVM read circuit	8
Avg/max number of eNVM write pulses	8 / 16
Duration of one eNVM write pulse	5 ns

4.3.2 Simulation Results and Discussion

A. Area

In this case study, the total area of the macro is defined as the bounding box of the whole architecture, which may leave some space unused at the corners. Fig. 4.8 shows the occupied and unused area of SRAM and eNVM synaptic cores with different number of partitions. Here the number of partitions (N) means the array is divided into N×N sub-arrays. The SRAM synaptic core generally has a larger area because the SRAM cell area (6 transistors) is much larger and it also uses multiple cells to represent one weight element.

With more partitions into the small arrays, both synaptic cores need more space for multiple copies of the peripheral circuits and A&Rs.

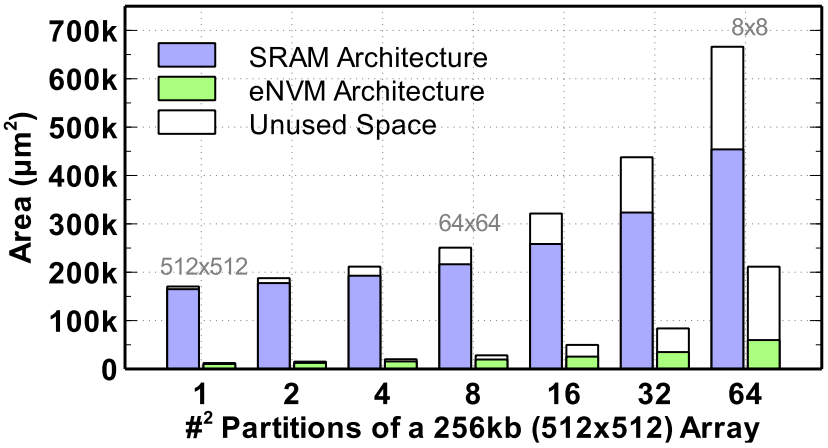


Fig. 4.8 The area of SRAM and eNVM synaptic cores with different number of partitions on a 256 kb (512×512) array. Grey labels are the array sizes of the partitioned sub-arrays. The eNVM synaptic core can achieve much smaller area with small cell size and multiple bits per cell, while the unused space will dominate the macro area as more partitions are applied. © 2016 IEEE.

B. Latency

The weighted sum and weight update latency of SRAM and eNVM synaptic cores with different number of partitions are shown in Fig. 4.9. Without partitioning applied, SRAM is slower in read due to row-by-row access, but faster in write because many write pulses are needed for eNVM to tune its conductance, and each pulse is 5 ns. With more partitions, it is expected that more partial weighted sums can be processed in parallel to reduce the weighted sum and weight update latency. For eNVM, partitioning could relax the precision requirement of the partial weighted sum in each sub-array thus sub-array latency can be smaller. However, it requires more stages of A&R for the weighted sum operation thus the latency of A&R accumulate. The results of eNVM suggest that the sum of weighted sum

latency for all A&R stages eventually becomes greater than the array weighted sum latency. To this point, the partitioning is no longer an effective way to reduce the weighted sum latency. In addition, due to the time-multiplexing required for the eNVM synaptic core in the weighted sum operation, the overall weighted sum latency of eNVM can be larger than that of SRAM beyond the partition point of 16×16.

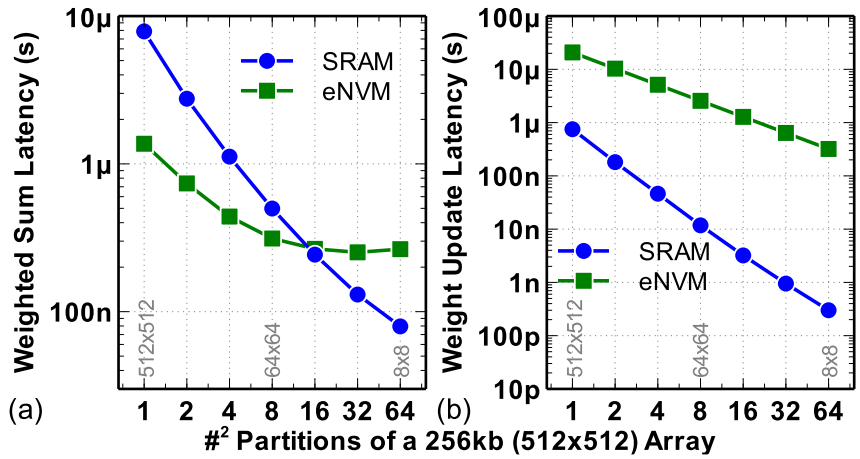


Fig. 4.9 The weighted sum and weight update latency per operation for the SRAM and eNVM synaptic cores with different number of partitions on a 256 kb (512×512) array. Partitioning introduces parallelism for the weighted sum and weight update operation, but A&R may become the critical path when more stages are used, especially in the eNVM weighted sum operation with time-multiplexing applied on the A&Rs as well. © 2016 IEEE.

C. Energy Consumption

Fig. 4.10 shows the weighted sum and weight update energy consumption for SRAM and eNVM synaptic cores with different number of partitions. The energy consumption refers to the dynamic energy consumption per weighted sum and weight update operation. The results have a similar trend with the latency, where the SRAM synaptic core consumes more energy in the weighted sum operation and less in the weight update operation. The reason can be attributed to the row-by-row based read and digital weight update in SRAM.

The results also reveal that there is a minimum weighted sum energy consumption for the eNVM synaptic core at a partition point of 8×8, indicating energy from A&Rs will dominate beyond this point.

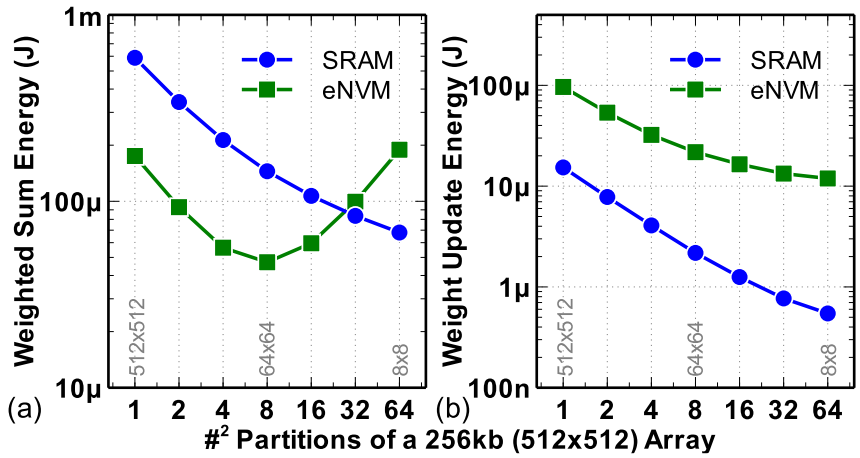


Fig. 4.10 The weighted sum and weight update energy consumption per operation for the SRAM and eNVM synaptic cores with different number of partitions on a 256 kb (512×512) array. Reduction of energy consumption with more partitions is not as clear as that of latency because A&R is rather power-consuming. The result suggests that 8×8 (or an array size of 64×64) may be a suitable partition point for the eNVM synaptic core. © 2016 IEEE.

D. Leakage Power Consumption

The leakage power consumption is calculated in the standby mode of the circuits. As shown in Fig. 4.11, the leakage power of SRAM is much larger than that of eNVM, primarily because eNVM cells are non-volatile. The leakage power consumption of eNVM synaptic core comes from the peripheral circuits, which is small compared with that of the SRAM array as the SRAM cells are the major contributor of leakage power.

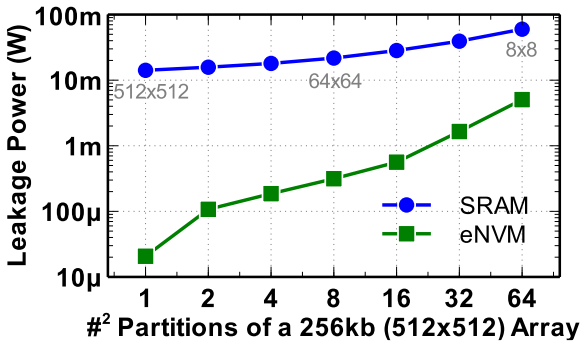


Fig. 4.11 The leakage power consumption of the SRAM and eNVM synaptic cores with different number of partitions on a 256 kb (512×512) array. The SRAM synaptic core has much larger leakage power because the power supply is needed for all SRAM cells to maintain the data. © 2016 IEEE.

4.4 Summary

In this chapter, we have introduced the synaptic array architectures, circuit modules, memory device/transistor models, functions and features in NeuroSim with detailed description. As a circuit-level macro model, NeuroSim alone can be a handy tool to estimate the circuit-level performance metrics of neuro-inspired architectures by taking trace of data patterns or average parameters. With clear abstractions of all hierarchical layers and well-defined interfaces of modules, NeuroSim can also be used as a supporting module to provide circuit-level performance estimation in neural network learning algorithms.

In the case study, NeuroSim was used to evaluate the performance of SRAM and eNVM synaptic core for weighted sum and weight update in the learning algorithms. The eNVM synaptic core outperforms the SRAM in the area and leakage and is suitable for read-intensive applications. The results of partitioning suggest that the SRAM synaptic core with more partitions and finer granularity can achieve significant reduction on the latency and energy consumption due to computation parallelism, with trade-off of the area

and leakage overhead. In the weighted sum operation, eNVM synaptic core does not gain as much benefit as the SRAM from the partitioning, due to the latency and energy consumption of multi-stage A&Rs in the finer granularity.

5 INTEGRATED DEVICE-TO-ALGORITHM SIMULATION FRAMEWORK WITH NEUROSIM

Neuromorphic hardware architectures based on synaptic memory arrays have been proposed for on-chip acceleration of weighted sum and weight update in machine/deep learning algorithms. Implementation of these architectures requires co-design of device, circuit and algorithm to achieve high learning performance while reducing the hardware cost. Many prior works [39, 59, 99] have studied the impact of several non-ideal eNVM synaptic device properties on the learning accuracy, but they could not address the impact on the circuit-level performance (e.g. area, latency, dynamic energy and leakage power) because they just incorporated the device behavioral model directly to the algorithm's code. On the other hand, some reported architectural simulator platforms (e.g. PRIME [103] and Harmonica [104]) have demonstrated powerful capability and flexibility at the system-level design, but they have limited considerations at the aforementioned non-ideal device properties (they only considered the weight precision and/or variation). MNSIM [78] is a circuit-level macro model of neuro-inspired architecture, but the accuracy in this model is the output error of weighted sum (vector-matrix multiplication), which is just one step of the algorithms thus it lacks the run-time learning accuracy of the entire algorithms. In such context, it is crucial to develop a simulation platform that is hierarchically organized from the device level, circuit level up to the algorithm level, where each level covers a wide variety of design options.

In this chapter, following the 3rd usage scenario in Section 4.1.5, we use NeuroSim as a supporting tool for a 2-layer MLP neural network with MNIST handwritten digits [105] as the training and testing dataset to implement online learning and offline classification.

The impact of the “analog” eNVM’s non-ideal device properties will be analyzed and architectures of analog and digital synapses will be benchmarked. Reliability issues due to data retention and write endurance failures will also be investigated.

5.1 Adapt MLP Network to Hardware

The network topology is 400(input layer)-100(hidden layer)-10(output layer). 400 neurons of input layer correspond to 20×20 MNIST image (edge cropped), and 10 neurons of output layer correspond to 10 classes of digits. Such simple 2-layer MLP can achieve 96~97% in the software baseline. In online learning, the MLP simulator emulates hardware to train the network with images randomly picked from the training dataset (60k images) and classify the testing dataset (10k images). In offline classification, the network is pre-trained by software, and the MLP simulator only emulates hardware to classify the testing dataset. For the hardware implementation, the MNIST input images are converted to black and white (1-bit) data to reduce the complexity of input encoding, as shown in Fig. 5.1(a). For design simplicity, the neuron node is modularized to take the weighted sum of 1-bit input data and truncate it to 1-bit output value through a low-precision activation function (Heaviside step function, e.g. a simple comparator circuit) for the input of next neuron node, as shown in Fig. 5.1(b). In this way, offline classification, which is purely feed forward (FF), can be realized in 1-bit. However, the computation on the back propagation (BP) of weight update generally needs higher precision to update the small errors.

Fig. 5.1(c) shows the circuit block diagram for hardware implementation of the 2-layer MLP network. The weighted sum operation is performed using the synaptic cores. However, the weights used in a regular synaptic array can only represent positive values

(W_H=0~1), while the weights in algorithm can be either positive or negative values (W_A=-1~1). The algorithm's weighted sum is then expressed as

$$W_A V = (2W_H - J)V = 2W_H V - JV \tag{5.1}$$

where V is the input vector and J is the matrix of all ones that has the same dimension as W_A and W_H. In this equation, $W_H V$ is the weighted sum output from the synaptic core. Therefore, we squeeze W_A from (-1~1) to the range of W_H (0~1): i.e. -1 is mapped to 0, 0 is mapped to 0.5, and 1 is mapped to 1. To reconstruct $W_A V$, we have to perform a two-step read from the array: first, we read out $W_H V$, and then multiply $W_H V$ by 2 using a 1-bit left-shift, and then subtract JV (basically the sum of vector) from $W_H V$ through the adder at the periphery. The MSB (sign bit in 2's complement notation) of the adder output will be the 1-bit output of the low-precision activation function. It should be noted that we only consider the main sub-circuit modules for the neuron periphery at current stage of this work.

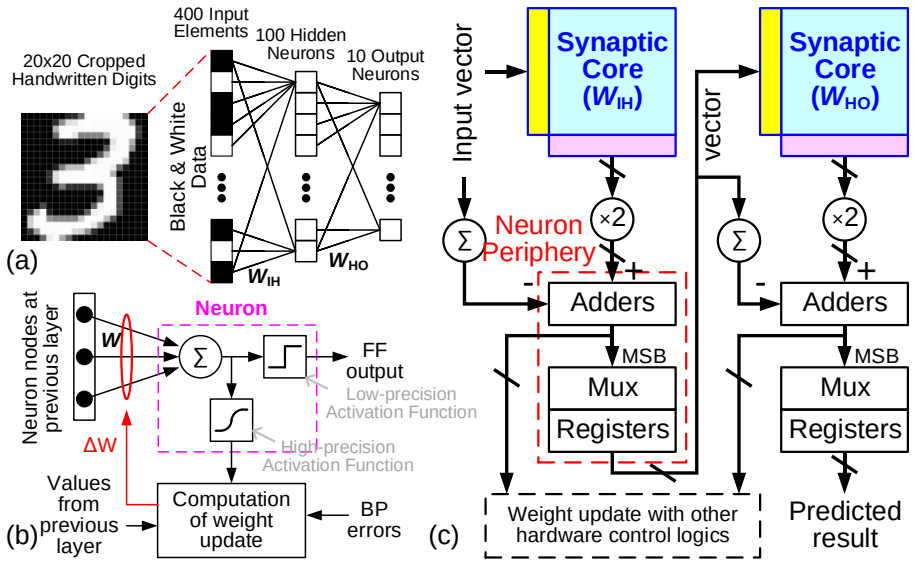


Fig. 5.1 (a) The 2-layer MLP neural network. (b) Schematic of a neuron node. (c) Circuit block diagram for hardware implementation of the 2-layer MLP network. © 2018 IEEE.

5.2 NeuroSim as a Supporting Module for MLP Simulator

The MLP simulator is shown in Fig. 5.2. It has a hierarchical organization from the algorithm level down to the device level with consideration of synaptic array and realistic device properties in detail, and it can be regarded as a standalone functional simulator that is able to evaluate the learning accuracy and the circuit-level performance for the synaptic array only during learning. To form a complete framework, NeuroSim is needed to provide circuit-level performance estimation.

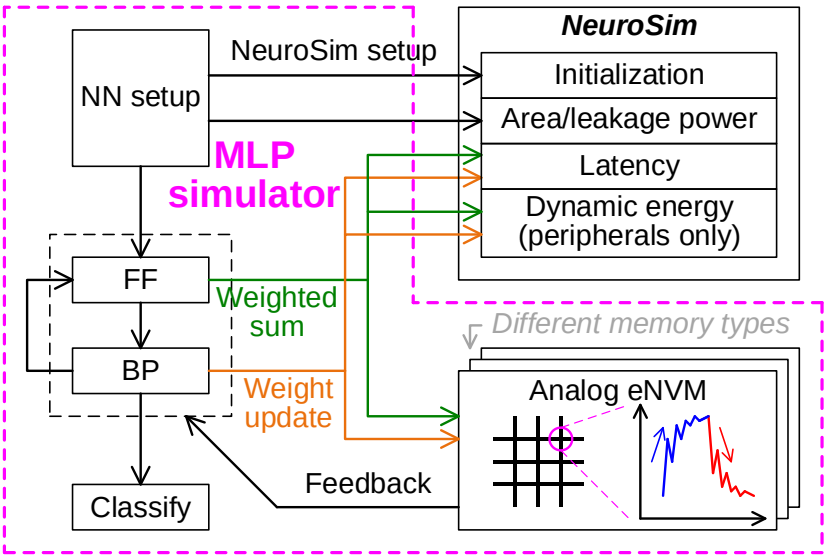


Fig. 5.2 NeuroSim as a supporting module to the MLP simulator. At the run-time of NN, the weighted sum and weight update instructions will be given to both the synaptic array/device model and NeuroSim for evaluation of computation error and circuit-level performances, respectively. © 2018 IEEE.

At the run-time of NN, the MLP simulator iteratively performs FF and BP, which contains a series of weighted sum and weight update operations, respectively. Whenever a weighted sum or weight update instruction is given, the instruction will be passed to the synaptic array and device behavioral model for calculation of computation error, as well as passed to NeuroSim for evaluation of circuit-level performances. As mentioned in the 3rd

usage scenario in Section 4.1.5, NeuroSim can be just responsible for the dynamic energy calculation of the array peripherals because the MLP simulator can better handle that of the synaptic array by itself.

5.3 Impact of Synaptic Device Properties on Accuracy

To quantify the impact of the aforementioned non-ideal device properties in Section 2.3, we performed sensitivity analyses in online learning and offline classification. Fig. 5.3(a) shows the requirement of weight precision. Because the memory resources are limited on-chip, we have to truncate the synapse weights into finite precisions. The result suggests that 6-bit weight is required for online learning, while 2-bit weight is needed for offline classification (at least for MNIST dataset) and 1-bit weight introduces slight degradation. Fig. 5.3(b) shows the learning accuracy with different conductance ON/OFF ratios. Limited ON/OFF ratio<50 will degrade the accuracy of offline classification. The network may adapt itself to this limited ON/OFF ratio during learning thus the online learning can tolerate more (ON/OFF ratio>10 is needed). However, the accuracy drop in online learning is sharper, which is probably because the network will deviate more from its correct form with both erroneous weighted sum and weight update results. Fig. 5.3(c) shows the impact of nonlinearity with different polarities of nonlinearity for the potentiation (P) and depression (D). The result shows that high nonlinearity can be tolerated if P/D has the same polarity. However, for common situations where P/D is positive/negative, the impact of nonlinearity on the online learning accuracy is very critical. High accuracy can only be achieved with small nonlinearity (<1). For offline classification, there is no nonlinearity issue as the cell conductance can be iteratively programmed to the desired value [106].

Variation sensitivity analyses are performed with different nonlinearities (P/D: positive/negative) in online learning. Fig. 5.3(d) shows the impact of conductance variation on the learning accuracy. We added the variation (with standard deviation (σ) in terms of percentage) on the highest conductance state (ON state) as it changes the conductance range most. The result shows that the conductance variation does not degrade the learning accuracy. Instead, it remedies the accuracy loss due to high nonlinearity. However, an opposite trend can be observed for the device-to-device variation, as shown in Fig. 5.3(e). The amount of device-to-device variation is defined as the nonlinearity baseline's standard deviation (σ) respect to 1 step of 6 steps, which is similar to the definition in Section 2.3.2. At low nonlinearity (<1), the accuracy slightly decreases with larger variation. For the nonlinearity>1, the impact becomes much more prominent. On the other hand, the amount of cycle-to-cycle variation (σ) is expressed in terms of the percentage of entire weight range, which is also similar to the definition in Section 2.3.2. As shown in Fig. 5.3(f), small cycle-to-cycle variation (<2%) can alleviate the degradation of learning accuracy by high nonlinearity. The reason may be attributed to the random disturbance that aids convergence of the weights to an optimal weight pattern (i.e. to help the system jump out of local minima). Thus, synaptic devices with nonlinear weight update behavior may perform better than expected if they exhibit a little noisy weight update. However, too large variation (>2%) overwhelms the deterministic weight update amount defined by the algorithm thus is harmful to the learning accuracy.

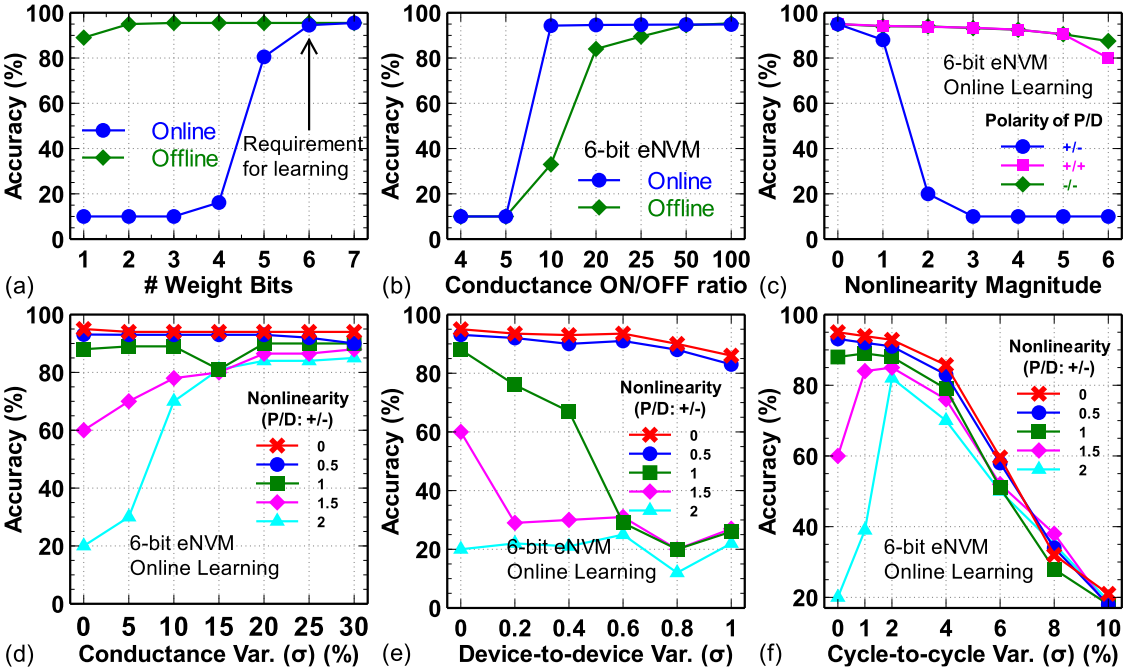


Fig. 5.3 The impact of (a) weight precision, (b) conductance ON/OFF ratio, (c) weight update nonlinearity, (d) conductance variation, (e) device-to-device variation and (f) cycle-to-cycle variation in online learning and/or offline classification.

5.4 Benchmark Results and Discussions

Table 5.1 and Table 5.2 survey representative analog eNVM and FeFET devices in literature with extracted aforementioned device properties such as number of conductance states, weight update nonlinearity, ON-state resistance (R_{ON}), ON/OFF ratio, programing pulse condition, and weight update variation, etc. Based on these parameters, NeuroSim was used to evaluate the system-level performance metrics such as learning accuracy, area, latency, energy and leakage power for online learning with 1 million MNIST images being trained. The benchmark results show that all analog eNVM devices fail to achieve a good accuracy>90%. The cause of degradation can be largely attributed to the devices' poor ON/OFF ratio. It is observed that for ON/OFF ratio<10, the devices cannot perform well in the learning no matter how good other parameters are. This agrees with the results in

Fig. 5.3(b). The second critical parameter is the nonlinearity. Even the PCMO device has slightly better ON/OFF ratio than the AlO_x/HfO_2 one, its high nonlinearity restrains itself from converging to the desired conductance during weight update, leading to a poor accuracy of 10%. In contrast, the learning accuracy of both FeFET devices is much better (~90%), owing to their large ON/OFF ratio. Even though their nonlinearities are not small, the degradation can be less critical if both potentiation and depression have the same nonlinearity polarity, as observed in Fig. 5.3(c).

Benchmark results of digital synapses are also included in Table 5.2 for comparison, where a digital eNVM with R_{ON}/R_{OFF}=200kΩ/10MΩ and 2.5V/10ns programming pulse is assumed. It can be observed that SRAM is better than digital eNVM in the latency and energy efficiency, but much worse in the area and leakage power. Despite that both these digital synapses can achieve better accuracy (~94%) than all analog synapses, they typically require 2.5X-10X area and >30X leakage power consumption (if SRAM). However, some analog synapses such as AlO_x/HfO_2 and GST PCM have less advantage in area due to their small R_{ON}, where the transistor W/L in peripheral circuits (such as Mux and switch matrixes) needs to be larger to prevent noticeable IR drop. On the other hand, it is found in analog synapses that most of the latency and energy are dominated by the weight update, and they are far too large compared to those in SRAM, making analog synapses not favorable for the online learning [66]. This is because we have used a naïve scheme for the weight update, where all cells in each operation need to go through the full number of pulse cycles (essentially the worst case) no matter the cells have to be updated (have a ΔW) or not. To optimize this scheme, we propose to use the maximum ΔW's number of cycles in each weight update operation. If all the cells in an operation do not

need an update (ΔW=0), this operation can even be skipped. Table 5.1 and Table 5.2 show the latency and energy with both the naïve and optimized schemes. In the optimized scheme, the latency in analog synapses are significantly reduced, indicating ΔW are often small or zero. In TaO_x/TiO_2 (Type B) and PCMO devices, the reduction ratios are extremely large because these devices basically learn nothing (almost no ΔW). Similarly, the energy can also be greatly reduced in the optimized scheme because skipping an operation saves the dynamic energy in charging the array wires and circuits. The only exceptions are AlO_x/HfO_2 and GST PCM. Their energy reduction is much less because their R_{ON} is small thus the array static energy (consumed by cells) dominates rather than the dynamic energy. All in all, if the programming pulse is further reduced (<20 ns), and if the peripheral circuit design can be made simpler for generating non-identical programming pulses, the analog synapses can be superior to digital synapses in nearly every aspect of the circuit-level performance with the optimized weight update scheme, as observed from the results of HfZrO (HZO) based FeFET.

Table 5.1 Specs and Online Learning Performance of Different Analog eNVM Synapses

	Analog eNVM synapses				
Device type	Ag:a-Si [32]	TaO_x/TiO_2 (Type B) [34]	PCMO [35]	AlO_x/HfO_2 [36]	GST PCM [40]
# of conductance states	97	102	50	40	100-120
Nonlinearity (weight increase/decrease)	2.4/-4.88	1.85/-1.79	3.68/-6.76	1.94/-0.61	0.105/2.4
R_{ON} (ON-state resistance)	26 MΩ	5 MΩ	23 MΩ	16.9 kΩ	4.71 kΩ
ON/OFF ratio	12.5	2	6.84	4.43	19.8
Weight increase pulse	3.2V/300µs	3V/40ms	-2V/1ms	0.9V/100µs	0.7V (avg.)/ 6µs
Weight decrease pulse	-2.8V/300µs	-3V/10ms	2V/1ms	-1V/100µs	3V (avg.)/ 125ns
Cycle-to-cycle variation (σ)	3.5%	<1%	<1%	5%	1.5%
Online learning accuracy	~73%	~10%	10%	~41%	~87%
Area	1072.0 μm^2	1071.3 μm^2	1071.3 μm^2	3657.2 μm^2	7233.0 μm^2
Latency (naïve)	4.20E8 s	3.57E10 s	7.00E8 s	5.60E7 s	4.39E6 s
Energy (naïve)	87.94 mJ	65.86 mJ	29.4 mJ	150 mJ	1.52 J
Latency (optimized)	64200 s	0.2845 s	5.2507 s	4439.8 s	413.0 s
Energy (optimized)	14.81 mJ	0.17 mJ	0.17 mJ	146.19 mJ	1.34 J
Leakage power	35.29 µW	35.29 µW	35.29 µW	35.29 µW	35.29 µW

Table 5.2 Specs and Online Learning Performance of Different Analog FeFET and Digital Synapses

	Analog FeFET synapses		Digital synapses	
Device type	HZO FeFET [53]	HZO FeFET [54]	6-bit SRAM	6-bit digital (binary) eNVM
# of conductance states	32	32	--	2
Nonlinearity (weight increase/decrease)	2.53/1.83	1.545/1.755	--	--
R_{ON} (ON-state resistance)	559.28 kΩ	500 kΩ	--	200 kΩ
ON/OFF ratio	45	~1300	--	50
Weight increase pulse	3.65V (avg.)/ 75ns	2.17V (avg.)/ 50µs	--	2.5V/10 ns
Weight decrease pulse	-2.95V (avg.)/ 75ns	-1.62V (avg.)/ 50µs	--	-2.5V/10 ns
Cycle-to-cycle variation (σ)	<1%	<1%	--	--
Online learning accuracy	~90%	~90%	~94%	~94%
Area	1190.4 μm^2	1193.5 μm^2	10311 μm^2	2681.9 μm^2
Latency (naïve)	3.36E4 s	2.24E7 s	7.76 s	162.3 s
Energy (naïve)	98.01 mJ	38.39 mJ	6.98 mJ	47.7 mJ
Latency (optimized)	1.2924 s	479.6 s	0.5217 s	1.8677 s
Energy (optimized)	0.28 mJ	0.21 mJ	2.2 mJ	1.6 mJ
Leakage power	35.29 µW	35.29 µW	1.1 mW	25.17 µW

For offline classification, accuracy>93% can be achieved using either 2-bit SRAM or digital eNVM (equivalently Fig. 5.3(a)) or 2-bit analog eNVM with sufficiently large ON/OFF ratio=50. Table 5.3 shows the circuit-level performance benchmark results of SRAM, digital and analog eNVM based architectures for offline classification on the entire testing dataset of 10k images. Without any training process, the analog eNVM based architecture can be superior to the other two designs in terms of latency and energy.

Table 5.3 Benchmark of Architecture with SRAM, Digital and Analog eNVM Based Synaptic Core for Offline Classification. © 2018 IEEE.

	2-bit SRAM	**2-bit digital eNVM**	**2-bit analog eNVM**
Area	4450.8 μm^2	1071.2 μm^2	1247.3 μm^2
Latency	32.997 ms	10.39 ms	0.25 ms
Dynamic Energy	16.939 μJ	7.30 μJ	3.38 μJ
Leakage Power	475.67 μW	22.89 μW	35.29 μW

5.5 Reliability Analysis

Besides the non-ideal device properties studied in the previous section, reliability issues such as data retention and write endurance could also be harmful to the learning performance of neural networks. In this section, we investigate the impact of data retention and write endurance with generic assumptions of all possible failure mechanisms by incorporating the retention and endurance models into the MLP simulator. Since the emphasis is on the reliability, we set the synaptic weight to be 6-bit (64 levels) and assumes linear conductance tuning without variation in all the simulations.

5.5.1 Data Retention

Data retention refers to the ability of memory device to retain its programmed state over a long period of time. Typical retention specification for NVM in memory application

is more than 10 years at 85^{o}C. Many binary eNVM devices have been able to meet this requirement. However, there are no reported data for analog eNVM that shows such retention, which can be attributed to the instability of intermediate conductance states [107]. To be general, we consider four scenarios of conductance drift for the retention analysis. As shown in Fig. 5.4(a)-(c), the conductance can either drift toward its maximum, minimum or intermediate states. These three scenarios have ever been reported in the retention measurement of binary eNVMs [108-110]. In addition, we also consider random conductance drift towards its maximum or minimum state with equal probability, as shown in Fig. 5.4(d). The formula for modeling the conductance drift behavior is assumed to follow the one that is widely used in PCM [111, 112], which can be described as

$$G=G_0\left(\frac{t}{t_0}\right)^{v} \tag{5.2}$$

where G_0 is the initial conductance, t is the retention time, v is the drift coefficient and t_0 is the time constant which is assumed to be 1 second in this work. In the retention analyses, the offline classification is used with the conductance ON/OFF ratio set to be 50, which is a sufficiently large ratio, in order to still capture the conductance drift at the lowest conductance state.

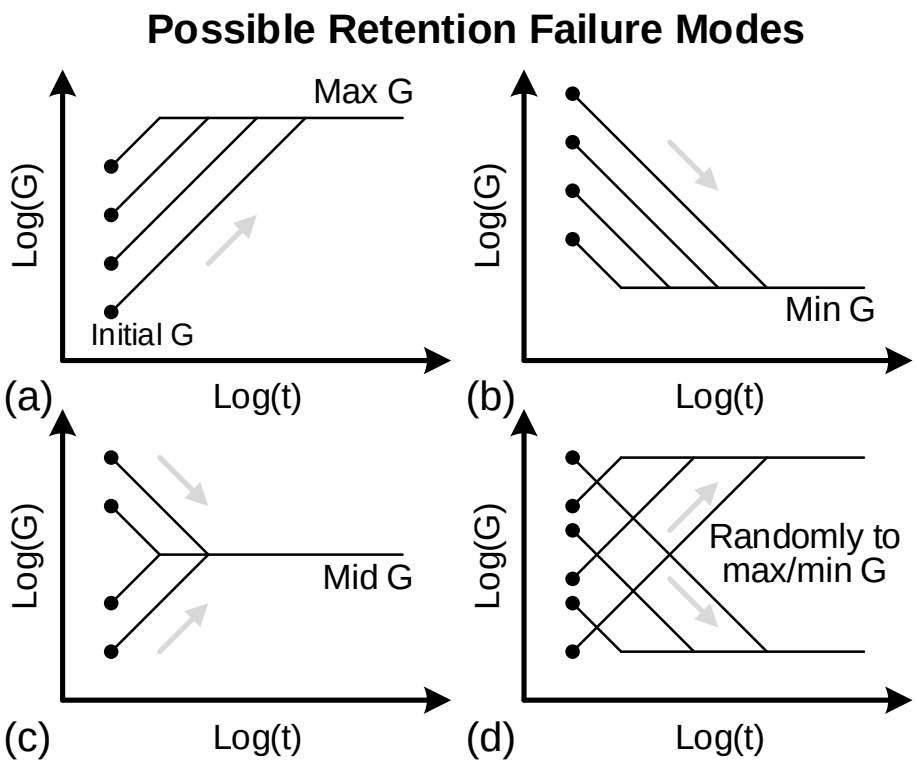


Fig. 5.4 General assumptions of retention failure modes: conductance drifting towards its (a) maximum state, (b) minimum state, (c) intermediate state, or (d) maximum/minimum state with randomness. © 2018 IEEE.

Fig. 5.5(a) shows the degradation of classification accuracy over retention time at a fixed drift coefficient of 0.01 with different final weight states that the conductance drifts to. It can be simply calculated that the conductance change is ~20% over 10 years under such drift coefficient, and it leads to degradation of accuracy <90% for all final weight states. On the other hand, the result suggests that the final state either be at the maximum or minimum conductance has the poorest accuracy. To have a quantitative comparison between different final weight states, we measure the maximum drift coefficient of all states that still give an accuracy >90% at a retention time of 10 years. As shown in Fig.

5.5(b), the final weight at 0.6 can tolerate up to a maximum drift coefficient of ~0.012, which corresponds to ~25% of the conductance change at 10 years.

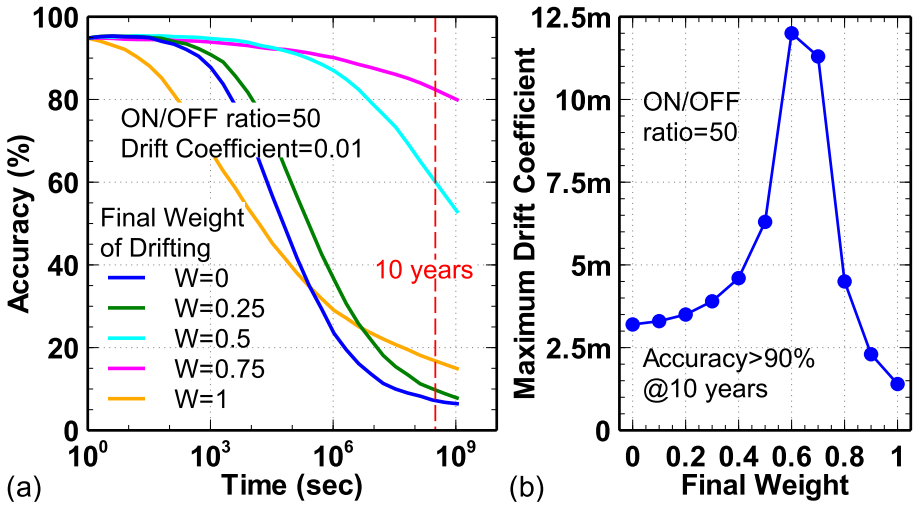


Fig. 5.5 (a) Classification accuracy as a function of retention time with conductance drifting toward different final weight states. (b) The maximum drift coefficient as a function of final weights for achieving >90% accuracy at 10 years. © 2018 IEEE.

The reason why intermediate final weight states (Fig. 5.4(c)) have less accuracy degradation than either the maximum or minimum ones (Fig. 5.4(a)-(b)) can be attributed to the deviation of weighted sum after retention degradation. This can be easily observed from the distribution of the absolute difference of column conductance sum before and after retention degradation, as shown in Fig. 5.6 for the first and second layer of MLP NN. The difference (ΔW) is measured between the array conductance patterns before and after a retention of 10 years, and a small drift coefficient of 0.001 is used to ensure that most of the conductance have not reached their final states at 10 years. As all the conductance will drift in the same direction to the maximum or minimum final weight state, a larger deviation of weighted sum is expected, and the high inverse correlation between Fig. 5.6

and Fig. 5.5(b) confirms that the accuracy degradation is strongly affected by the amount of weighted sum deviation.

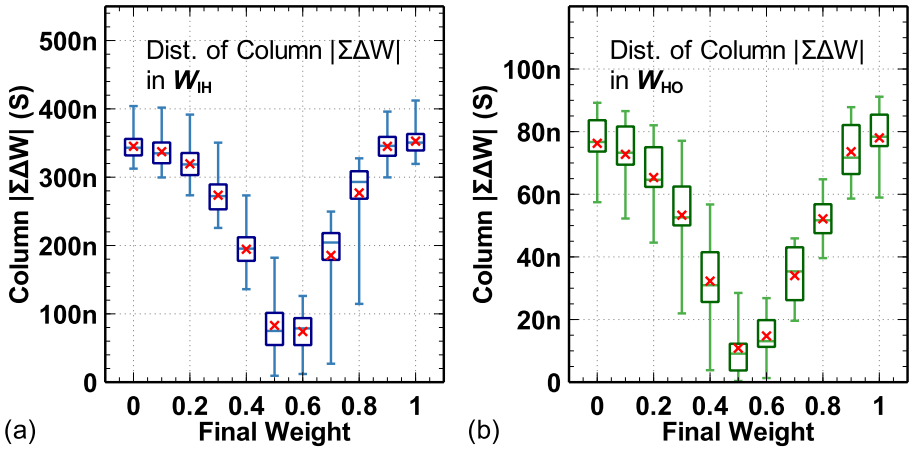


Fig. 5.6 Distribution of the absolute difference of column conductance sum before and after 10 years (drift coefficient=0.001) in the (a) first and (b) second layer of MLP NN. Both results are highly correlated with Fig. 5.5(b). © 2018 IEEE.

The above argument can be further substantiated by the analysis of random conductance drift in Fig. 5.4(d), where its impact on the classification accuracy is shown in Fig. 5.7. With the same drift coefficient of 0.01, the accuracy degradation is much less severe than the ones in other drift scenarios (Fig. 5.5(a)), even we select the worst result in Fig. 5.7 for comparison. The reason is because the weighted sum deviation will be averaged out by this randomness. It can be expected that if either drifting towards maximum or minimum conductance is much more probable, the accuracy degradation will be as severe as that of W=0 or W=1 in Fig. 5.5(a).

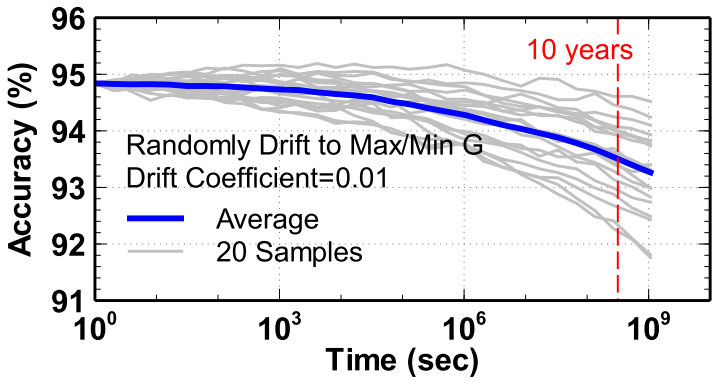


Fig. 5.7 Monte Carlo simulation on the accuracy with conductance randomly drifting toward its maximum or minimum states. Under the same drift coefficient, the randomness behavior does not lead to radical change in weighted sum thus the impact on the accuracy is much smaller compared to other conductance drift scenarios. © 2018 IEEE.

In fact, the only experimental work so far that reported the retention properties in analog RRAM suggests that its behavior can be due to multiple hops of oxygen vacancies over long retention time [107], which is analogous to Brownian Motion. It also shows that the read current distribution of each conductance level follows a normal distribution, where its standard deviation (σ) increases with retention time. In other words, the retention behavior can be modeled as an increasing conductance variation over time, which is illustrated in Fig. 5.8(a). From [107], its σ is described as

$$\sigma=\lambda\sqrt{t}+\theta \tag{5.3}$$

where λ and θ are fitting parameters. Since these fitting parameters can vary in different devices, conductance states and even temperatures, we rather evaluate the impact of this retention behavior based on σ. As shown in Fig. 5.8(b), a σ of ~0.2 will lead to a significant degradation on the accuracy. It can be calculated that given θ=0, λ should be smaller than ~7e-6 for the accuracy to remain >90% at 10 years.

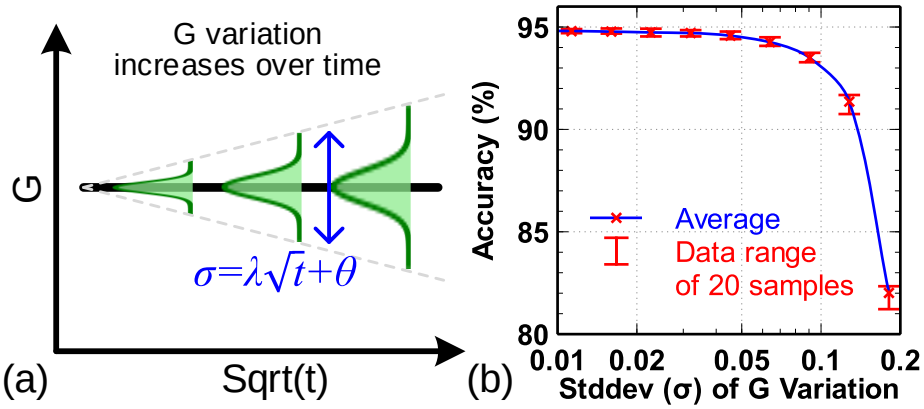


Fig. 5.8 (a) The retention model proposed in [107] suggesting the an increasing conductance variation over time. (b) The impact of conductance variation on the classification accuracy. © 2018 IEEE.

5.5.2 Write Endurance

In memory application, the write endurance specifies the number of times that a memory device can be programmed (written) before the write failure occurs. Typical binary eNVM devices can achieve $>10^6$ write cycles (between the highest and lowest conductance states). However, the analog eNVM endurance definition should be different as it has only incremental conductance change by each write pulse. So far, there is no prior work discussing the endurance behavior of analog eNVM for neuromorphic computing. To study the endurance effect in this work, we assume that the strength of conductance tuning (ΔG) decreases over write pulse cycles, which is expressed as

$$\Delta G=\Delta G_0(1-r)^{(\#pulses)} \quad (5.4)$$

where ΔG_0 is the ideal conductance change without considering endurance degradation, r is the reduction ratio, #pulses means the cumulative number of pulses that has been applied to the device. As illustrated in Fig. 5.9(a), the conductance will eventually be unchangeable

after an excessive number of cycles. To analyze its impact, we apply the endurance property in the online learning of the MLP NN. As shown in Fig. 5.9(b), the learning accuracy degradation begins to be noticeable as we gradually increase r to be >0.01. We also apply variations of 10% and 20% on the ratio, and it does not really either significantly alleviate or worsen the degradation.

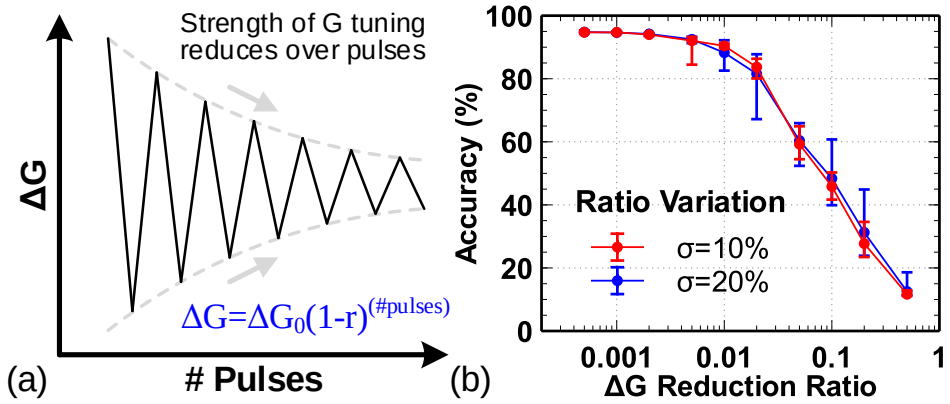


Fig. 5.9 (a) Endurance degradation in weight update of synaptic devices. Strength of conductance tuning decreases over pulse cycles. (b) The impact of ΔG reduction ratio (with 10% and 20% variation) on the learning accuracy. 10 device samples are measured for each data point. © 2018 IEEE.

In the endurance analysis, we assume the maximum conductance of the device is 100 nS. It can be calculated that the required cumulative number of pulses to reduce the strength of conductance tuning by 50% and 90% are ~70 and ~230 under r=0.01, respectively. Fig. 5.10(a)-(b) shows the distribution of the sum of absolute conductance change in the first and second layer of MLP NN without endurance effect to achieve the targeted learning accuracy. The conductance changes with 70 and 230 write pulses are also labeled. Given only the results of Fig. 5.10(a)-(b), we may easily conclude that r=0.01 is too large thus there will be a significant accuracy degradation, because most of the devices require far

more pulses than these two numbers to achieve >90% accuracy. However, the accuracy with r=0.01 in Fig. 5.9(b) disproves this argument.

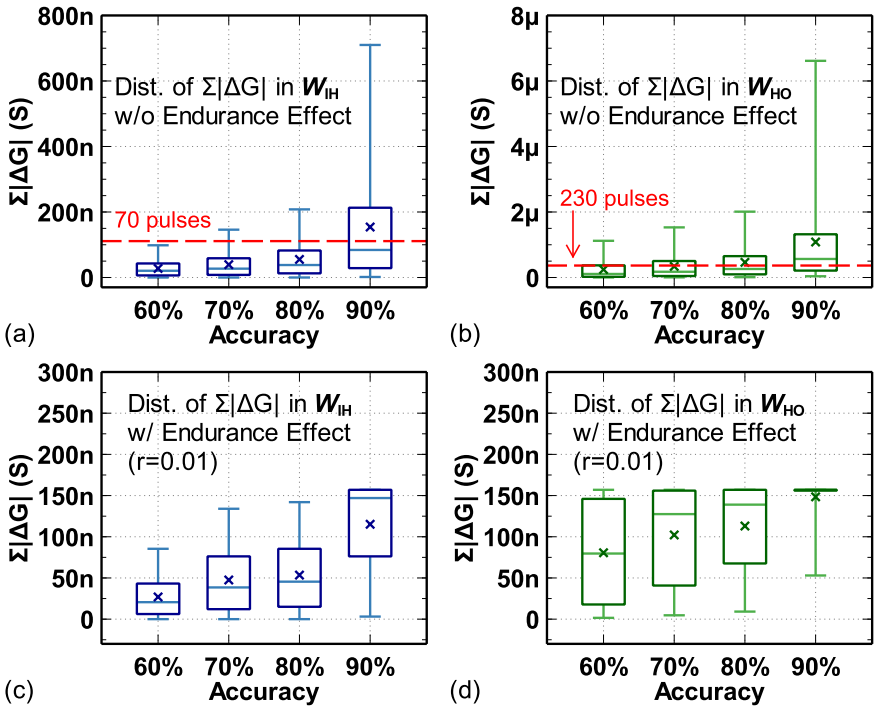


Fig. 5.10 Distribution of the sum of absolute conductance change in the (a) first and (b) second layer without endurance effect, and (c) first and (d) second layer of MLP NN with endurance effect (r=0.01). The network can adapt itself to this endurance degradation by activating other synaptic devices whose conductance are still tunable. © 2018 IEEE.

In fact, the network has the ability to adapt itself to this endurance degradation by relying on other devices whose conductance is still tunable. As shown in Fig. 5.10(c)-(d), the conductance cannot be further tuned beyond a certain amount of total conductance change (~150 nS), and the network will keep activating other inactive devices to take over the responsibility of learning during the entire learning process. To see this effect more clearly, 2D color maps of the total absolute conductance change in the first and second layer are shown in Fig. 5.11 and Fig. 5.12, respectively. Without endurance degradation, the training in the network only relies on the conductance change in some of the active

eNVM devices to achieve the 90% accuracy. With endurance degradation, most of the devices have to participate in the training to achieve the 90% accuracy, thus it can be observed that the entire color map almost ends up being filled with the same color (which means the conductance tuning limit). Besides the network's ability to adapt the endurance degradation from algorithm's point of view, analog eNVM devices were also demonstrated to have $>10^3$ write pulses of conductance tuning [34, 40]. Therefore, the endurance issue may not be as critical as estimated.

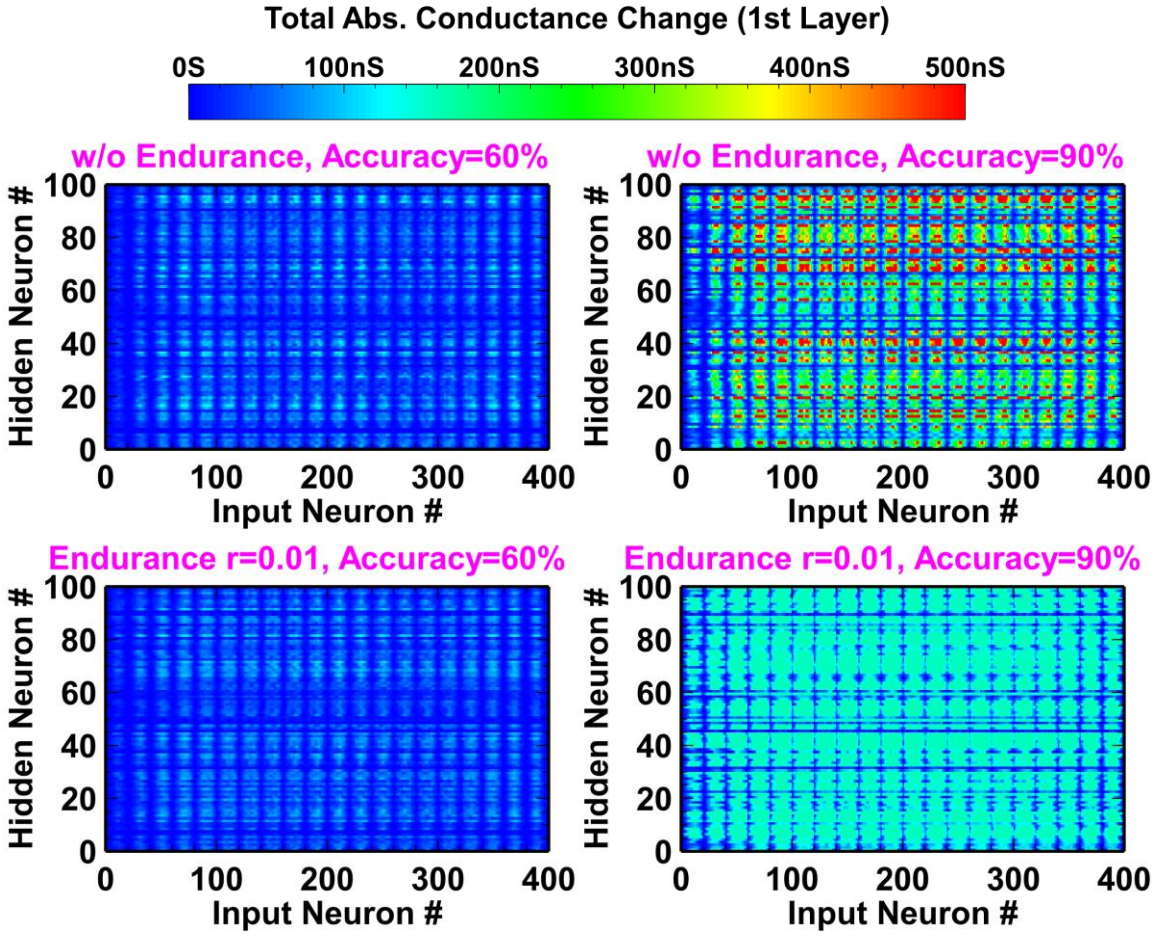


Fig. 5.11 2D color map of the total absolute conductance change in the first layer.

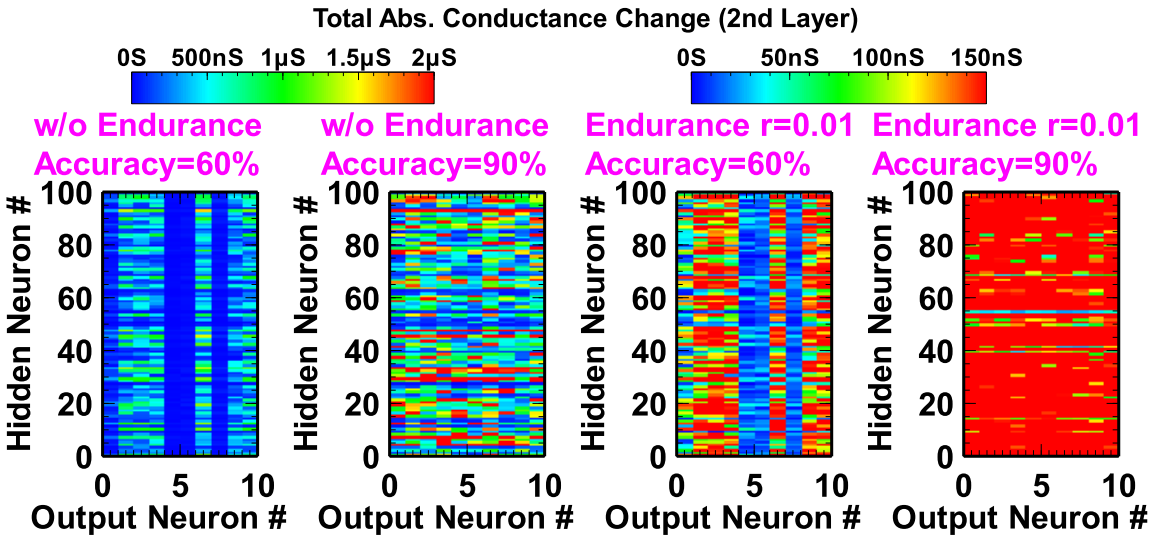


Fig. 5.12 2D color map of the total absolute conductance change in the second layer.

5.6 Summary

We have developed an integrated device-to-algorithm framework that connects circuit-level macro model NeuroSim to NNs to evaluate the learning performance of neuro-inspired architectures. We have used this framework to analyze the impact of non-ideal device properties and benchmark several representative analog synapses in a 2-layer MLP NN. The results suggest that degradation of learning accuracy is mainly due to small ON/OFF ratio and large nonlinearity with different polarities in potentiation and depression. The optimized weight update scheme is also proposed to minimize the latency and energy overhead by skipping redundant pulse cycles and even operations during training. With this scheme, analog synapses can be potentially better in hardware performance than SRAM synapses, while achieving >90% online learning accuracy. For read-intensive applications such as the offline classification, analog eNVM is the most suitable synaptic device due to its capability of parallel weighted sum operation.

Impact of important reliability properties for synaptic devices such as data retention and write endurance are also investigated. It is observed that there is a strong correlation between the degradation of classification accuracy and the weighted sum deviation, thus retention behaviors which causes less deviation will have smaller impact on the accuracy. The analysis also includes the existing retention model based on conductance variation, enabling estimation of the model parameters based on targeted performance. In contrast, the endurance issue defined in this work is considered to be less critical than estimated because the network is able to alleviate it by making use of other devices whose conductance are still tunable.

6 CONCLUSION

The crossbar array architecture with resistive synaptic devices has been proposed for on-chip implementation of weighted sum and weight update operations in the neuromorphic learning algorithms. It is crucial to explore the design methodologies for practical hardware implementation of the resistive cross-point array architecture, where we have recognized possible non-ideal device properties that are detrimental to the learning performance. Using sparse coding algorithm as a benchmark platform, we developed design strategies at both circuit and device levels to mitigate the impact of these non-ideal properties. By applying these strategies with tolerable trade-offs on chip area, latency and energy, it is shown that the synaptic behavior is greatly improved and the recognition accuracy can return from ~30% to ~95%.

Array design for performance improvement is also proposed. The 1S1R array architecture can reduce the weight update energy consumption compared to the crossbar array architecture. Alternatively, the "pseudo-crossbar" array architecture is even better in terms of the write disturbance and energy efficiency in weight update, which directly turns off the unselected rows. Besides the array, the peripheral circuits in crossbar and pseudo-crossbar are also discussed. As the read circuit is complex and not area-efficient, the MIT device is introduced as the oscillation neuron to replace the entire read circuit. To address the interference issue of oscillation between columns in simple crossbar array, the 2T1R array architecture is proposed at negligible increase in array area. In circuit-level benchmark, it is shown that the oscillation neuron not only saves a lot of area on chip, but also improves the latency and energy due to less sharing of neuron peripheral circuits by array columns.

As today's neuromorphic computing system is at early design stage, there can be a variety of design choices from the algorithm level to the device level, such as neural network topology, eNVM array architecture, peripheral circuit design and eNVM device engineering. Therefore, we developed NeuroSim platform that is beneficial in exploring the design space of neuro-inspired architectures at such early design stage. In the case study of array partitioning problem, we have demonstrated that NeuroSim alone can provide circuit-level performance estimation of neuro-inspired architectures thus the partition strategy can be simply envisioned.

For a more complex case, the role of NeuroSim can be a supporting tool. To explore the feasibility of different synaptic devices for neuromorphic computing, we integrated NeuroSim with a 2-layer multilayer perceptron (MLP) neural network to build an integrated device-to-algorithm simulation framework. The framework has shown its power in evaluating the performance of learning as well as other hardware metrics such as area, latency, dynamic energy and leakage for neuromorphic architectures. We believe that MLP+NeuroSim framework can be a handy and flexible tool to perform design optimization for on-chip implementation of learning with various mainstream and emerging synaptic device technologies. The source code of MLP+NeuroSim framework is publically available at [113] for other researchers to download. To support the more advanced learning algorithms such as convolutional neural network (CNN), recurrent neural network (RNN) and/or spiking neural network (SNN), which could be the future work for extension.

REFERENCES

[1] Y. LeCun, and C. Cortes, "The MNIST database of handwritten digits," 1998.

[2] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1701-1708, 2014.

[3] Amazon Alexa, http://alexa.amazon.com/

[4] Apple Siri personal assistant, http://www.apple.com/ios/siri/

[5] Microsoft Cortana personal assistant, http://www.microsoft.com/en-us/mobile/experiences/cortana/

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, pp. 5, 2009.

[7] C. Urmson, J. A. Bagnell, C. R. Baker, M. Hebert, A. Kelly, R. Rajkumar, P. E. Rybski, S. Scherer, R. Simmons, and S. Singh, "Tartan racing: A multi-modal approach to the DARPA Urban Challenge," *Technical report, Carnegie Mellon University*, 2007.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484-489, 2016.

[9] M. Emilio, M. Moises, R. Gustavo, and S. Yago, "Pac-mAnt: Optimization based on ant colonies applied to developing an agent for Ms. Pac-Man," *IEEE Symposium on Computational Intelligence and Games (CIG)*, pp. 458-464, 2010.

[10] The CIFAR-10 and CIFAR-100 dataset, https://www.cs.toronto.edu/~kriz/cifar.html

[11] D. Steinkraus, I. Buck, and P. Simard, "Using GPUs for machine learning algorithms," *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1115-1120, 2005.

[12] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," *International Conference on Machine Learning (ICML)*, pp. 873-880, 2009.

[13] J. Von Neumann, "The principles of large-scale computing machines," *Annals of the History of Computing*, vol. 3, no. 3, pp. 263-273, 1981.

[14] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE,* vol. 78, no. 10, pp. 1629-1636, 1990.

[15] D. B. Strukov, "Nanotechnology: smart connections," *Nature,* vol. 476, no. 7361, pp. 403-405, 2011.

[16] B. Belhadj, A. Joubert, Z. Li, R. Héliot, and O. Temam, "Continuous real-world inputs can open up alternative accelerator designs," *International Symposium on Computer Architecture (ISCA),* pp. 1-12, 2013.

[17] X. Jin, M. Lujan, L. A. Plana, S. Davies, S. Temple, and S. Furber, "Modeling spiking neural networks on SpiNNaker," *Computing in Science & Engineering,* vol. 12, no. 5, pp. 91-97, 2010.

[18] S. Kumar, "Introducing qualcomm zeroth processors: Brain-inspired computing," 2013.

[19] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE,* vol. 102, no. 5, pp. 699-716, 2014.

[20] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, and Y. Nakamura, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science,* vol. 345, no. 6197, pp. 668-673, 2014.

[21] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "CNP: An fpga-based processor for convolutional networks," *International Conference on Field Programmable Logic and Applications (FPL),* pp. 32-37, 2009.

[22] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "NeuFlow: A runtime reconfigurable dataflow processor for vision," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* pp. 109-116, 2011.

[23] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 G-ops/s mobile coprocessor for deep neural networks," *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),* pp. 682-687, 2014.

[24] I. Kuon, and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* vol. 26, no. 2, pp. 203-215, 2007.

[25] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning,"

International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 269-284, 2014.

[26] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, and N. Sun, "DaDianNao: A machine-learning supercomputer," *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 609-622, 2014.

[27] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen, "PuDianNao: A polyvalent machine learning accelerator," *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 369-381, 2015.

[28] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting vision processing closer to the sensor," *International Symposium on Computer Architecture (ISCA)*, pp. 92-104, 2015.

[29] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, and A. Borchers, "In-datacenter performance analysis of a tensor processing unit," *International Symposium on Computer Architecture (ISCA)*, pp. 1-12, 2017.

[30] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, pp. 382001, 2013.

[31] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, "A learning-enabled neuron array IC based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71-81, 2013.

[32] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297-1301, 2010.

[33] I.-T. Wang, Y.-C. Lin, Y.-F. Wang, C.-W. Hsu, and T.-H. Hou, "3D synaptic architecture with ultralow sub-10 fJ energy per spike for neuromorphic computation," *IEEE International Electron Devices Meeting (IEDM)*, pp. 665-668, 2014.

[34] L. Gao, I.-T. Wang, P.-Y. Chen, S. Vrudhula, J. Seo, Y. Cao, T.-H. Hou, and S. Yu, "Fully parallel write/read in resistive synaptic array for accelerating on-chip learning," *Nanotechnology*, vol. 26, no. 45, pp. 455204, 2015.

[35] S. Park, A. Sheri, J. Kim, J. Noh, J. Jang, M. Jeon, B. Lee, B. R. Lee, B. H. Lee, and H. Hwang, "Neuromorphic speech systems using advanced ReRAM-based synapse," *IEEE International Electron Devices Meeting (IEDM)*, pp. 625-628, 2013.

[36] J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, and H. Hwang, "Improved synaptic behavior under identical pulses using AlO_x/HfO_2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Letters,* vol. 37, no. 8, pp. 994-997, 2016.

[37] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H. S. P. Wong, "A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation," *Advanced Materials,* vol. 25, no. 12, pp. 1774-1779, 2013.

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

[39] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, and E. U. Giacometti, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 Synapses) using phase-change memory as the synaptic weight element," *IEEE Transactions on Electron Devices,* vol. 62, no. 11, pp. 3498-3507, 2015.

[40] D. Kuzum, R. G. Jeyasingh, B. Lee, and H.-S. P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano letters,* vol. 12, no. 5, pp. 2179-2186, 2011.

[41] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," *IEEE International Electron Devices Meeting (IEDM),* pp. 79-82, 2011.

[42] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, and J. Sunwoo, "A 20nm 1.8 V 8Gb PRAM with 40MB/s program bandwidth," *International Solid-State Circuits Conference (ISSCC),* pp. 46-48, 2012.

[43] R. Fackenthal, M. Kitagawa, W. Otsuka, K. Prall, D. Mills, K. Tsutsui, J. Javanifard, K. Tedrow, T. Tsushima, Y. Shibahara, and G. Hush, "A 16Gb ReRAM with 200MB/s write and 1GB/s read in 27nm technology," *International Solid-State Circuits Conference (ISSCC),* pp. 338-339, 2014.

[44] H. Y. Lee, P. S. Chen, T. Y. Wu, Y. S. Chen, C. C. Wang, P. J. Tzeng, C. H. Lin, F. Chen, C. H. Lien, and M.-J. Tsai, "Low power and high speed bipolar switching with a thin reactive Ti buffer layer in robust HfO_2 based RRAM," *IEEE International Electron Devices Meeting (IEDM),* pp. 1-4, 2008.

[45] Y. Wu, B. Lee, and H.-S. P. Wong, "Ultra-low power Al_2O_3-based RRAM with 1μA reset current," *International Symposium on VLSI Technology Systems and Applications (VLSI-TSA),* pp. 136-137, 2010.

[46] C. Ho, E. Lai, M. Lee, C. Pan, Y. Yao, K. Hsieh, R. Liu, and C.-Y. Lu, "A highly reliable self-aligned graded oxide WO_x resistance memory: conduction mechanisms and reliability," *IEEE Symposium on VLSI Technology (VLSI-T)*, pp. 228-229, 2007.

[47] Z. Wei, Y. Kanzawa, K. Arita, Y. Katoh, K. Kawai, S. Muraoka, S. Mitani, S. Fujii, K. Katayama, and M. Iijima, "Highly reliable TaO_x ReRAM and direct evidence of redox reaction mechanism," *IEEE International Electron Devices Meeting (IEDM)*, pp. 1-4, 2008.

[48] C. Rohde, B. J. Choi, D. S. Jeong, S. Choi, J.-S. Zhao, and C. S. Hwang, "Identification of a determining parameter for resistive switching of TiO_2 thin films," *Applied Physics Letters*, vol. 86, no. 26, pp. 262907, 2005.

[49] W. Wu, H. Wu, B. Gao, N. Deng, S. Yu, and H. Qian, "Improving Analog Switching in HfO_x-Based Resistive Memory With a Thermal Enhanced Layer," *IEEE Electron Device Letters*, vol. 38, no. 8, pp. 1019-1022, 2017.

[50] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges," *Advanced materials*, vol. 21, no. 25-26, pp. 2632-2663, 2009.

[51] S. Park, H. Kim, M. Choo, J. Noh, A. Sheri, S. Jung, K. Seo, J. Park, S. Kim, and W. Lee, "RRAM-based synapse for neuromorphic system with pattern recognition function," *IEEE International Electron Devices Meeting (IEDM)*, pp. 231-234, 2012.

[52] R. Pandian, B. J. Kooi, J. L. Oosthoek, P. van den Dool, G. Palasantzas, and A. Pauza, "Polarity-dependent resistance switching in GeSbTe phase-change thin films: The importance of excess Sb in filament formation," *Applied Physics Letters*, vol. 95, no. 25, pp. 252109, 2009.

[53] M. Jerry, P.-Y. Chen, J. Zhang, P. Sharma, K. Ni, S. Yu, and S. Datta, "Ferroelectric FET analog synapse for acceleration of deep neural network training," *IEEE International Electron Devices Meeting (IEDM)*, pp. 139-142, 2017.

[54] S. Oh, T. Kim, M. Kwak, J. Song, J. Woo, S. Jeon, I. K. Yoo, and H. Hwang, "$HfZrO_x$-Based Ferroelectric Synapse Device With 32 Levels of Conductance States for Neuromorphic Applications," *IEEE Electron Device Letters*, vol. 38, no. 6, pp. 732-735, 2017.

[55] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of BSB recall function using memristor crossbar arrays," *Design Automation Conference (DAC)*, pp. 498-503, 2012.

[56] P.-Y. Chen, D. Kadetotad, Z. Xu, A. Mohanty, B. Lin, J. Ye, S. Vrudhula, J. Seo, Y. Cao, and S. Yu, "Technology-design co-optimization of resistive cross-point

array for accelerating learning algorithms on chip," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 854-859, 2015.

[57] J. Liang, and H.-S. P. Wong, "Cross-point memory array without cell selectors - device characteristics and data storage pattern dependencies," *IEEE Transactions on Electron Devices*, vol. 57, no. 10, pp. 2531-2538, 2010.

[58] P.-Y. Chen, L. Gao, and S. Yu, "Design of Resistive Synaptic Array for Implementing On-Chip Sparse Learning," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 4, pp. 257-264, 2016.

[59] P.-Y. Chen, B. Lin, I. Wang, T.-H. Hou, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, and S. Yu, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," *ACM/IEEE International Conference on Computer-Aided Design (ICCAD)*, pp. 194-199, 2015.

[60] S. Yu, P.-Y. Chen, Y. Cao, L. Xia, Y. Wang, and H. Wu, "Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect," *IEEE International Electron Devices Meeting (IEDM)*, pp. 451-454, 2015.

[61] D. Kadetotad, Z. Xu, A. Mohanty, P.-Y. Chen, B. Lin, J. Ye, S. Vrudhula, S. Yu, Y. Cao, and J.-s. Seo, "Parallel architecture with resistive crosspoint array for dictionary learning acceleration," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, vol. 5, no. 2, pp. 194-204, 2015.

[62] P.-Y. Chen, J. Seo, Y. Cao, and S. Yu, "Compact oscillation neuron exploiting metal-insulator-transition for neuromorphic computing," *International Conference on Computer-Aided Design (ICCAD)*, 2016.

[63] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[64] P.-Y. Chen, and S. Yu, "Partition SRAM and RRAM based synaptic arrays for neuro-inspired computing," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2310-2313, 2016.

[65] P.-Y. Chen, X. Peng, and S. Yu, "System-level benchmark of synaptic device characteristics for neuro-inspired computing," *IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pp. 1-2, 2017.

[66] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: an integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," *IEEE International Electron Devices Meeting (IEDM)*, pp. 135-138, 2017.

[67] P.-Y. Chen, and S. Yu, "Reliability perspective of resistive synaptic devices on the neuromorphic system performance," *IEEE International Reliability Physics Symposium (IRPS)*, 2018.

[68] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," *Advances in neural information processing systems*, pp. 801-808, 2006.

[69] D. H. Hubel, and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106-154, 1962.

[70] B. A. Olshausen, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607-609, 1996.

[71] B. Lin, Q. Li, Q. Sun, M.-J. Lai, I. Davidson, W. Fan, and J. Ye, "Stochastic coordinate coding and its application for drosophila gene expression pattern annotation," *CoRR abs/1407.8147*, 2014.

[72] C.-C. Chang, and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2011.

[73] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, "Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers in Neuroscience*, vol. 9, no. 222, 2015.

[74] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv preprint arXiv:1602.02830v3*, 2016.

[75] International Technology Roadmap for Semiconductors (ITRS), http://www.itrs2.net/

[76] P. Narayanan, G. Burr, R. Shenoy, K. Virwani, and B. Kurdi, "Circuit-level benchmarking of access devices for resistive nonvolatile memory arrays," *IEEE International Electron Devices Meeting (IEDM)*, pp. 705-708, 2014.

[77] A. Chen, "Comprehensive methodology for the design and assessment of crossbar memory array with nonlinear and asymmetric selector devices," *IEEE International Electron Devices Meeting (IEDM)*, pp. 746-749, 2013.

[78] L. Xia, B. Li, T. Tang, P. Gu, X. Yin, W. Huangfu, P.-Y. Chen, S. Yu, Y. Cao, and Y. Wang, "MNSIM: Simulation platform for memristor-based neuromorphic computing system," *ACM/IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 469-474, 2016.

[79] B. Li, Y. Wang, Y. Wang, Y. Chen, and H. Yang, "Training itself: Mixed-signal training acceleration for memristor-based neural network," *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 361-366, 2014.

[80] D. Chabi, Z. Wang, W. Zhao, and J.-O. Klein, "On-chip supervised learning rule for ultra high density neural crossbar using memristor for synapse and neuron," *International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 7-12, 2014.

[81] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 10, pp. 1864-1878, 2014.

[82] K. Moon, E. Cha, J. Park, S. Gi, M. Chu, K. Baek, B. Lee, S. Oh, and H. Hwang, "High density neuromorphic system with $Mo/Pr_{0.7}Ca_{0.3}MnO_3$ synapse and NbO_2 IMT oscillator neuron," *IEEE International Electron Devices Meeting (IEDM)*, pp. 464-466, 2015.

[83] Y. Zhou, and S. Ramanathan, "Mott memory and neuromorphic devices," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1289-1310, 2015.

[84] Z. Yang, C. Ko, and S. Ramanathan, "Oxide electronics utilizing ultrafast metal-insulator transitions," *Annual Review of Materials Research*, vol. 41, pp. 337-367, 2011.

[85] S. G. Kim, T. J. Ha, S. Kim, J. Y. Lee, K. W. Kim, J. H. Shin, Y. T. Park, S. P. Song, B. Y. Kim, and W. G. Kim, "Improvement of characteristics of NbO_2 selector and full integration of $4F^2$ 2x-nm tech 1S1R ReRAM," *IEEE International Electron Devices Meeting (IEDM)*, pp. 249-252, 2015.

[86] Y. Zhou, and S. Ramanathan, "Correlated electron materials and field effect transistors for logic: a review," *Critical Reviews in Solid State and Materials Sciences*, vol. 38, no. 4, pp. 286-317, 2013.

[87] T. Wang, and J. Roychowdhury, "Design tools for oscillator-based computing systems," *Design Automation Conference (DAC)*, pp. 1-6, 2015.

[88] N. Shukla, A. Parihar, E. Freeman, H. Paik, G. Stone, V. Narayanan, H. Wen, Z. Cai, V. Gopalan, and R. Engel-Herbert, "Synchronized charge oscillations in correlated electron systems," *Scientific reports*, vol. 4, 2014.

[89] S. P. Levitan, Y. Fang, J. A. Carpenter, C. N. Gnegy, N. S. Janosik, S. Awosika-Olumo, D. M. Chiarulli, G. Csaba, and W. Porod, "Associative processing with coupled oscillators," *International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 235-235, 2013.

[90] Y. W. Lee, B.-J. Kim, J.-W. Lim, S. J. Yun, S. Choi, B.-G. Chae, G. Kim, and H.-T. Kim, "Metal-insulator transition-induced electrical oscillation in vanadium dioxide thin film," *Applied Physics Letters,* vol. 92, no. 16, 2008.

[91] M. D. Pickett, and R. S. Williams, "Sub-100 fJ and sub-nanosecond thermally driven threshold switching in niobium oxide crosspoint nanodevices," *Nanotechnology,* vol. 23, no. 21, pp. 215202, 2012.

[92] M. D. Pickett, G. Medeiros-Ribeiro, and R. S. Williams, "A scalable neuristor built with Mott memristors," *Nature Materials,* vol. 12, no. 2, pp. 114-117, 2013.

[93] A. Sharma, T. Jackson, M. Schulaker, C. Kuo, C. Augustine, J. Bain, H.-S. Wong, S. Mitra, L. Pileggi, and J. Weldon, "High performance, integrated 1T1R oxide-based oscillator: Stack engineering for low-power operation in neural network applications," *IEEE Symposium on VLSI Technology (VLSI-T),* pp. T186-T187, 2015.

[94] Y. Koo, K. Baek, and H. Hwang, "Te-based amorphous binary OTS device with excellent selector characteristics for x-point memory applications," *IEEE Symposium on VLSI Technology (VLSI-T),* pp. 1-2, 2016.

[95] S. Li, X. Liu, S. K. Nandi, D. K. Venkatachalam, and R. G. Elliman, "High-endurance megahertz electrical self-oscillation in Ti/NbO_x bilayer structures," *Applied Physics Letters,* vol. 106, no. 21, pp. 212902, 2015.

[96] S. J. Wilton, and N. P. Jouppi, "CACTI: An enhanced cache access and cycle time model," *IEEE Journal of Solid-State Circuits,* vol. 31, no. 5, pp. 677-688, 1996.

[97] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 31, no. 7, pp. 994-1007, 2012.

[98] Predictive Technology Model (PTM), http://ptm.asu.edu/

[99] S. Agarwal, S. J. Plimpton, D. R. Hughart, A. H. Hsia, I. Richter, J. A. Cox, C. D. James, and M. J. Marinella, "Resistive memory device requirements for a neural algorithm accelerator," *International Joint Conference on Neural Networks (IJCNN),* pp. 929-938, 2016.

[100] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Computer Architecture Letters,* vol. 10, no. 1, pp. 16-19, 2011.

[101] FreePDK45, https://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[102] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems (NIPS)*, pp. 1097-1105, 2012.

[103] P. Chi, S. Li, Z. Qi, P. Gu, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A Novel Processing-In-Memory Architecture for Neural Network Computation in ReRAM-based Main Memory," *ACM/IEEE International Symposium on Computer Architecture (ISCA)*, pp. 27-39, 2016.

[104] X. Liu, M. Mao, B. Liu, B. Li, Y. Wang, H. Jiang, M. Barnell, Q. Wu, J. Yang, H. Li, and Y. Chen, "Harmonica: A Framework of Heterogeneous Computing Systems With Memristor-Based Neuromorphic Computing Accelerators," *IEEE Transactions on Circuits and Systems I (TCAS-I)*, vol. 63, no. 5, pp. 617-628, 2016.

[105] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[106] L. Gao, P.-Y. Chen, and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," *IEEE Electron Device Letters*, vol. 36, no. 11, pp. 1157-1159, 2015.

[107] M. Zhao, H. Wu, B. Gao, Q. Zhang, W. Wu, S. Wang, Y. Xi, D. Wu, N. Deng, S. Yu, H.-Y. Chen, and H. Qian, "Investigation of statistical retention of filamentary analog RRAM for neuromophic computing," *IEEE International Electron Devices Meeting (IEDM)*, pp. 872-875, 2017.

[108] S. H. Jo, K.-H. Kim, T. Chang, S. Gaba, and W. Lu, "Si memristive devices applied to memory and neuromorphic circuits," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 13-16, 2010.

[109] Y. Y. Chen, M. Komura, R. Degraeve, B. Govoreanu, L. Goux, A. Fantini, N. Raghavan, S. Clima, L. Zhang, and A. Belmonte, "Improvement of data retention in HfO_2/Hf 1T1R RRAM cell under low operating current," *IEEE International Electron Devices Meeting (IEDM)*, pp. 252-255, 2013.

[110] A. Prakash, D. Jana, and S. Maikap, "TaO_x-based resistive switching memories: prospective and challenges," *Nanoscale research letters*, vol. 8, no. 1, pp. 418, 2013.

[111] R. A. Cobley, C. D. Wright, and J. A. V. Diosdado, "A model for multilevel phase-change memories incorporating resistance drift effects," *IEEE Journal of the Electron Devices Society*, vol. 3, no. 1, pp. 15-23, 2015.

[112] S. Kim, B. Lee, M. Asheghi, F. Hurkx, J. P. Reifenberg, K. E. Goodson, and H.-S. P. Wong, "Resistance and threshold switching voltage drift behavior in phase-change memory and their temperature dependence at microsecond time scales

studied using a micro-thermal stage," *IEEE Transactions on Electron Devices*, vol. 58, no. 3, pp. 584-592, 2011.

[113] MLP simlator (+NeuroSim) version 1.0, https://github.com/neurosim/MLP_NeuroSim