UNIVERSITY OF CALIFORNIA

Los Angeles

Charge-Trap Transistors for Neuromorphic Computing

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Electrical and Computer Engineering

by

Xuefeng Gu

2018

ABSTRACT OF THE DISSERTATION

Charge-Trap Transistors for Neuromorphic Computing

by

Xuefeng Gu

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2018

Professor Subramanian Srikantes Iyer, Chair

As the demand for energy-efficient cognitive computing keeps increasing, the conventional von Neumann architecture becomes power/energy prohibitive. Brain-inspired, or neuromorphic computing has been extensively investigated in the past three decades because of its distributed memory/processors and massive connectivity, which promise low-power operation. One critical device in such a system is the synapse – a local memory which stores the connectivity between neurons. Many devices, such as resistive memory, phase-change memory, ferroelectric field-effect-transistor, and flash memory, have been suggested as candidates for analog synapses. In this work, the use of a CMOS-only and manufacturing-ready candidate – the charge-trap transistor (CTT), is investigated.

The analog programming characteristics of CTTs most pertinent to neuromorphic applications will first be investigated. In particular, the analog retention, the fine-tuning of individual CTTs, the spike-timing dependent plasticity, the weight-dependent plasticity, and the device variation

will be discussed. The implications learned from this part serve as the basic understanding for subsequent chapters using CTTs for neuromorphic applications.

Next, two algorithms for unsupervised learning, namely, winner-takes-all (WTA) clustering and temporal correlation detection, are investigated, using CTTs as the analog synapses. For each algorithm, the feasibility of hardware implementation using CTTs as the analog synapses is first studied and system performance evaluated using experimentally measured CTT characteristics. Experimental demonstration is then presented using custom-built CTT arrays in the 22 nm fully depleted silicon-on-insulator (SOI) technology.

Finally, the use of CTTs for analog synapses in an inference engine is considered. The fine-tuning of CTT weights in an array setting is first examined as it is anticipated to be different from that of discrete devices because of the half-selection and thermal disturbance by adjacent cells. The achieved standard deviation of the difference between the target and the actually programmed weight is as low as 6% of the dynamic range. The programmed CTT array is then used as a dot product engine, the key to an inference engine. Implications of the imperfect array programming in the accuracy of an inference engine are then discussed.

This dissertation of Xuefeng Gu is approved.

Jason C. S. Woo

Sudhakar Pamarti

Achuta Kadambi

Janakiraman Viraraghavan

Subramanian Srikantes Iyer, Committee Chair

University of California, Los Angeles

2018

To my wife Peiyun and daughter Xi-Mu

Table of Contents

Acknowledgements

I would first like to thank my advisor and lifetime mentor, Prof. Subramanian Iyer, for his tremendous support and guidance. This work would not have been possible without his gracious help over the years. I am grateful to the numerous discussions we had regarding research, work, and life. I am so fortunate to have this extraordinary teacher instill in me the strong work ethic, the freedom of thinking, and a way of life.

I also greatly appreciate the help and encouragement I received from my committee members, Prof. Jason Woo, Prof. Sudhakar Pamarti, Prof. Achuta Kadambi, and Prof. Janakiraman Viraraghavan. In particular, the countless discussions I had with Prof. Pamarti and Prof. Viraraghavan during the tapeout are tremendously helpful.

I have been lucky to have worked with collaborators from the industry. Their perspectives have benefited me a lot. In particular, I would like to thank Toshiaki Kirihata, Dan Berger, Norm Robson, Navneet Jain, Juhan Kim, Maciej Wiatr, and Laks Vanamurthy from GlobalFoundries, and John Barth from Invecas.

At UCLA, I enjoyed great company from all my CHIPS friends. I received a lot of help from my colleagues in the smaller NeuroCTT team: Zhe Wan, Steven Moran, Premsagar Kittur, Faraz Khan, and Jonathan Cox. Also thanks to those we have helped me one way or the other: Adeel Bajwa, Boris Vaisband, Amir Hanna, Siva Jangam, Arsalan Alam, Meng-Hsiang Liu, among many others. We are lucky to have Kyle Jung in CHIPS, who meticulously takes care of everything we need in many aspects.

I would like to thank Deeona Columbia for her great patience when helping me, and Ryo Arreola for his effort in the administrative procedures. I would also like to thank Minji Zhu for his technical support in some of the experiments.

In the seven months that I spent in IBM East Fishkill, I had the opportunity to work with and learn from many wonderful colleagues: Jang Sim, Yongchun Xin, Cheng-Yi Lin, Yang Yang,

Xiang Chen, Qintao Zhang, Geng Wang, Ravikumar Ramachandran, Sami Rosenblatt, among many others. I would also like to thank Tom Koscal from GlobalFounrdies for his help in setting up the characterization platform in the early stage of this project.

On the personal side, I enjoy great friendship with so many: Dingkun Ren, Li Du, Yuan Du, Yubo Wang, Ti-Wen Lin, Luyao Xu, Rui Zhu, among many others.

In the end, I would like to thank my family. My parents Xianghe and Ling supported me selflessly when I needed it the most. My wife Peiyun has always been by my side at *our* toughest times, for which I am forever grateful. We have accomplished this together. To my beloved daughter Xi-Mu, your coming to this world is the best gift I have ever received, and I cherish every single moment with you. For a while, it was your smile that kept me going.

List of Figures

List of Tables

VITA

September 2007 – June 2011	Bachelor of Engineering, Southeast University
September 2011 – March 2013	Master of Science, University of California, Los Angeles
May 2016 – December 2016	IBM Systems, Hopewell Junction, NY
March 2013 – December 2018	Ph.D., University of California, Los Angeles

.

1. Introduction

1.1 Background of neuromorphic computing

Over the past four decades, the size of transistors has been scaled by over three orders of magnitude. This is mainly fueled by the need to reduce cost. Interestingly, the scaling of transistors has also been the driving force of the ever-increasing system performance and functionality. However, as the transistor size has been pushed close to the manufacturing limit (production of 7 nm technology has begun by TSMC and Samsung as of the writing of this dissertation), several key challenges present themselves. First, from the device perspective, the parasitic capacitance and resistance has increased dramatically due to the ever-shrinking feature size (Fig. 1.1). In fact, some argue that the parasitic elements might be the most critical challenge for sub-10-nm nodes [1]. Second, from the system perspective, because the number of interconnects *between* chips has not scaled as much (only 3−4 times), the data-fetching between chips (say, between the CPU and the memory) using high-speed serializer-deserializer (SerDes) consumes a significant portion of the power budget and occupies a large area (Fig. 1.2). Finally, the increasing non-recurring engineering (NRE) cost has also made scaling devices alone a less attractive solution.

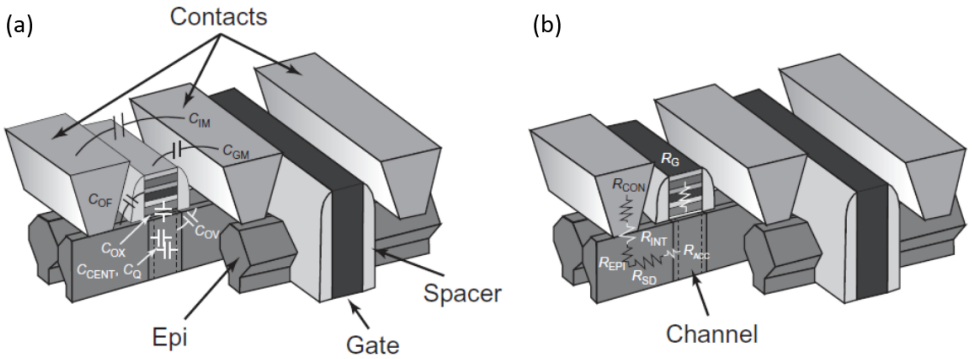

Figure 1.1 Various parasitic elements in a 3D trigate transistor with epitaxial source/drain. (a) Capacitance, (b) Resistance. Adapted from [1].

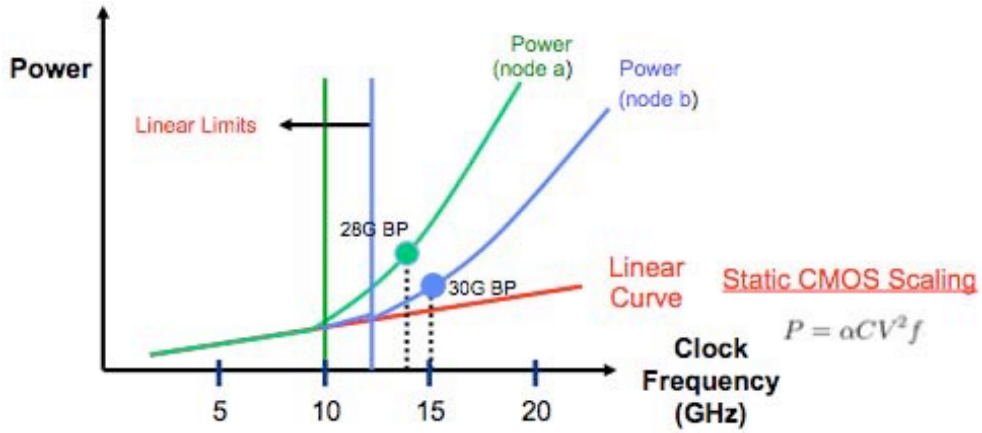

Figure 1.2 Exponential increase in power as a function of data rate. Node b is smaller than node a. Also shown are the linear power limits of static logic circuits. Adapted from [2].

An architecture change has long been considered necessary to keep improving the system performance without further shrinking the transistors. For example, three-dimensional (3D) wafer-scale integration (3D-WSI) and silicon interconnect fabric (Si-IF) targeting heterogeneous integration are being actively pursued (Fig. 1.3) [2−4]. While these approaches address the interconnect issue to some degree, the fundamental problem – the need for high-speed communication between different components of a system, remains.

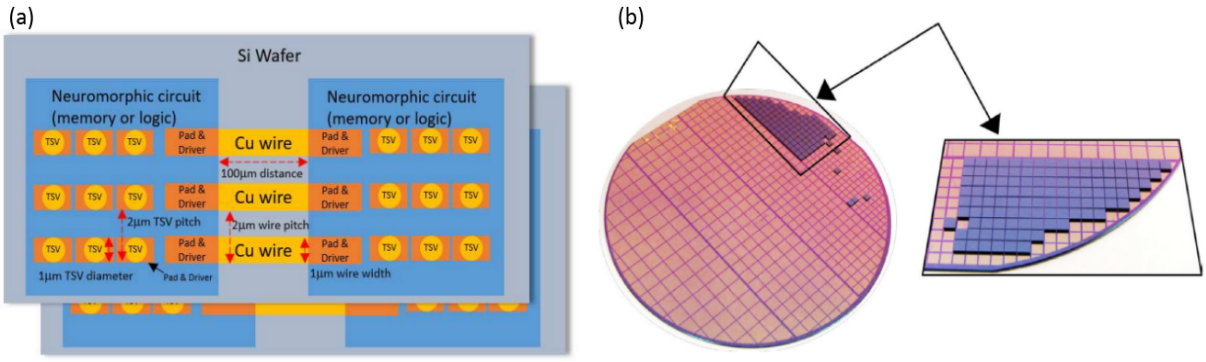

Figure 1.3 (a) The schematic of the interconnect within a 3D-WSI system. Adapted from [3]. (b) Si-dielets bonded onto a Si-IF with 100 μm inter-dielet distance. Adapted from [4].

Despite all these efforts, the state-of-the-art systems, using CPU and/or GPU, can barely compete with humans in many cognitive tasks, not to mention the tremendously more space and

power required by those systems. In this regard, the human brain is a remarkable cognitive device. While occupying a volume of only two liters and consumes merely 20 W, it performs many tasks such as image recognition and speech processing amazingly well.

Compared to the conventional von Neumann computing architecture, the main features of the human brain are its distributed memory and processor, and massive connectivity between them. In a human brain, there are ~ 10^{11} neurons (the processor) and ~ 10^{14} synapses (the connection between neurons which stores the connectivity). As shown in Fig. 1.4, each neuron receives input signals from its dendrites and sends an output signal through its axon. Between the axon of a neuron and the dendrite of the next neuron is a structure called synapse which stores the "connectivity" between the two neurons. When a neuron receives a pre-synaptic signal, its membrane potential is updated according to the synaptic strength, and when a certain threshold is reached, the neuron fires – sending a spike to the next neuron – and returns to its rest state.

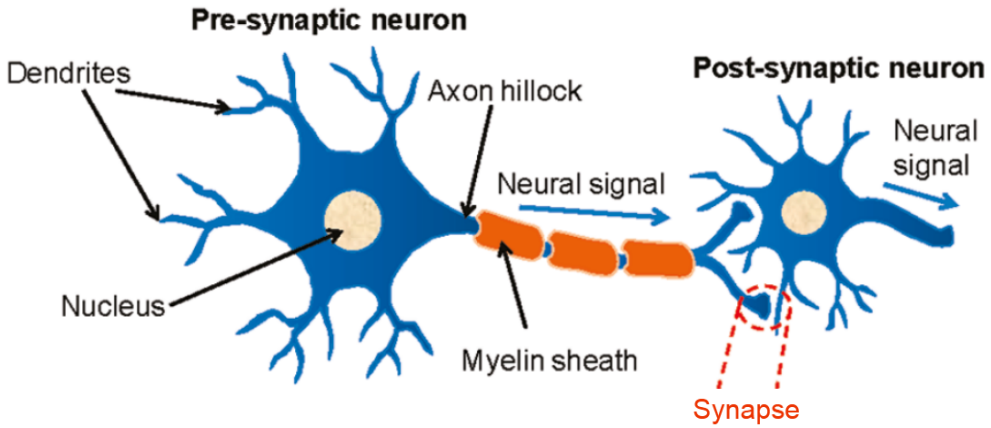

Figure 1.4 Illustration of the neuron structure and the synapse between neurons. Adapted from [42].

Unlike most computers which separate the memory from the CPU and constantly ferry data between the two, the brain stores and processes information locally, and as a result promises massively parallel processing capability with little power consumption. Therefore, brain-inspired

neuromorphic computing systems are considered a strong candidate to *complement* conventional von-Neumann computers in many cognitive tasks [5-10].

1.2 Overview of existing efforts in neuromorphic computing

The concept of neuromorphic computing can be dated back to the 1950s and gained momentum in the 1980s when a group of researchers in Caltech began looking into the possibility of building electronic circuits to mimic how various parts of the body work [11-13]. Since then, efforts have focused on building hardware inspired by the human brain, including, but not limited to, the Neurogrid [14], the HRL SyNAPSE [15], the SpiNNaker [16, 17], and the IBM TrueNorth [9]. Table 1.1 summarizes some of the key aspects of these systems and a more comprehensive survey can be found in [18].

Table 1.1 A summary of important parameters of some existing neuromorphic systems. Adapted from [18].

System	# of neurons	# synapses/neuron	Synapse implementation	Learning rule	Power
Neurogrid	1,048,576	6×10^9 total	SRAM	None	2.7 W
SpiNNaker	1,000	1,000	SRAM	Any	1 W
SyNAPSE	576	128	Memristor	STDP	130 mW
TrueNorth	1,048,576	256	SRAM	None	63 mW

These systems can be categorized based on their purposes: to simulate a large-scale neuromorphic system (ultimately at the scale of the human brain), or to solve cognitive problems more efficiently with brain-inspired architecture. The first category, which Neurogrid and SpiNNaker belong to, aims at building a platform for neuroscientists to perform biological real-

time simulations with much less power compared to conventional computers, understand how the brain works, and potentially build intelligent machines. For example, Neurogrid uses a few watts to simulate a million neurons in real time whereas a personal computer uses a few hundred watts to simulate 2.5 million neurons 9,000 times slower than real time [14].

What is more relevant to this dissertation is the second category, which tries to solve real-world cognitive tasks with brain-inspired neural networks. HRL SyNAPSE and IBM TrueNorth belong to this category. The HRL SyNAPSE adopts the CMOS technology to emulate all neural and synaptic computations and memristor technology for high-density analog synapse storage [15]. The IBM TrueNorth, on the other hand, exclusively relies on the CMOS technology. Built on Samsung's 28-nm technology, and fully capable of running convolutional neural networks, the chip has 5.4 billion transistors with 4096 neuro-synaptic cores interconnected via an intrachip network that integrates 1 million programmable spiking neurons and 256 million configurable synapses. Although the sizes of these systems are not comparable to human brains, it is interesting to note that they can already perform some cognitive tasks at biological real time and a much lower power compared to conventional von Neumann architectures. For example, when neurons fire on average at 20 Hz and have 128 active synapses, the total measured power of TrueNorth was 72 mW, corresponding to 26 pJ per synaptic event (considering total energy). This is 176,000 times more energy efficient compared with an optimized simulator running the exact same network on a modern general-purpose microprocessor, and 769 times more energy efficient compared with a state-of-the-art multiprocessor neuromorphic approach running a similar network [9].

Note that the IBM TrueNorth utilizes ternary synaptic weights (0 and ±1) stored in SRAM. This has at least two drawbacks. First, due to the coarse precision of synaptic weights, a larger-scale network is required for competitive accuracy. Second, since it requires six transistors to store one bit of information in SRAM, the synapses end up occupying ~ 30% area of the chip. These

problems can be addressed by using analog memory for the synapses, as is the case in HRL SyNAPSE where memristive devices are used.

Indeed, since the publication of "*The missing memristor found*" in 2008 [19], the interest in neuromorphic, or brain-inspired computing, has skyrocketed (Fig. 1.5). A lot of research has focused on the characterization of various analog synaptic devices including resistive memory, phase-change memory, magnetic memory, ferroelectric FETs, flash memory, etc., and their applications in neuromorphic computing [20−24]. In the next Section, we will review the representative device characteristics of some of these analog memory devices, which leads to the discussion of a new device (in Sections 1.4 and 1.5) that will be studied in this dissertation: the charge-trap transistors (CTTs).

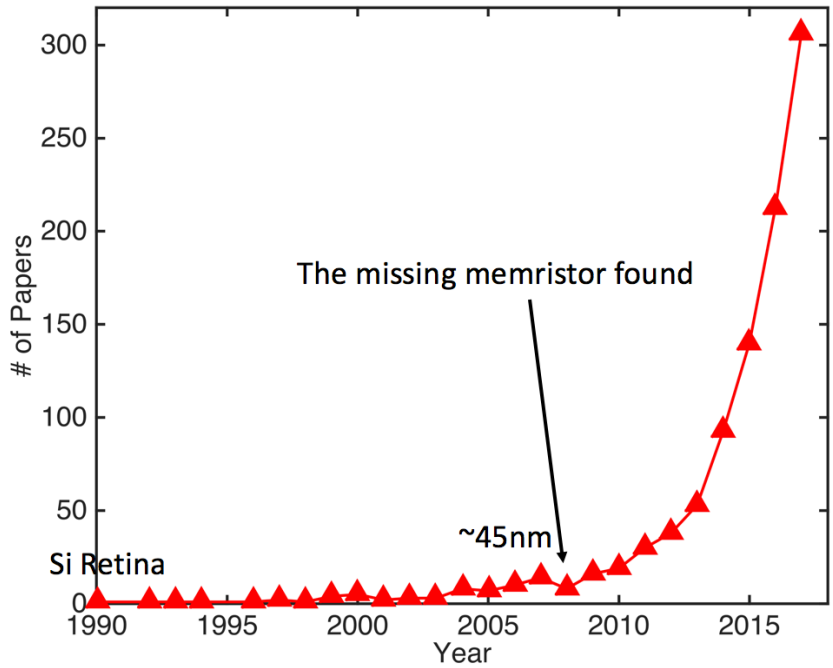

Figure 1.5 Interest in neuromorphic or brain-inspired computing in the past three decades. Web of Science search criteria: Neuromorphic Computing OR Brain-inspired Computing.

1.3 Existing analog synapses

A memristor is the fourth fundamental passive circuit element (besides the resistor, the capacitor, and the inductor) predicted by Leon Chua from symmetry arguments in 1971 [25]. The behavior of a memristor can be described with two simple equations [19]:

$$v = R(w,i)i \tag{1.1}$$

$$\mathrm{d}w/\mathrm{d}t = f(w,i) \tag{1.2}$$

where w is a set of state variables, and f and R are functions of time. Note from Eq. (1.2) that, the state variable w depends on the history of the current flowing through the device, making the resistance R also dependent on the current's history. Although such nonlinear device characteristics hold great promise to provide very valuable circuit functionalities–for example, high-density electronic resistance switches–no direct connection was found between the mathematics and the physical properties of any practical system for almost four decades. This gap was closed by a group of researchers from HP Labs in 2008 [19]. Using a simple analytical example, the authors showed that, memristance arises naturally in nanoscale systems in which solid-state electronic and ionic transport are coupled under an external bias voltage, paving the path for extensive memristor research in the past decade [26–35].

Fig. 1.6 shows the typical switching characteristics of a bipolar TiN/HfOx/AlOx/Pt memristor cell. In Fig. 1.6(a), the device first goes through a SET process where the voltage gradually increases from 0 to 2 V, with a current compliance of 100 μA enforced. The current starts to increase very rapidly at about 1.6 V, and the device is programmed to a low resistance state. The device then goes through a RESET process (with a voltage of as high as –3.3 V) to return to its original state. The compliance current during the SET process can be utilized to control the state

the memristor is programmed to. In Fig. 1.6(b), compliance currents of 1–200 μA are used in different SET processes, resulting in seven intermediate resistance states.

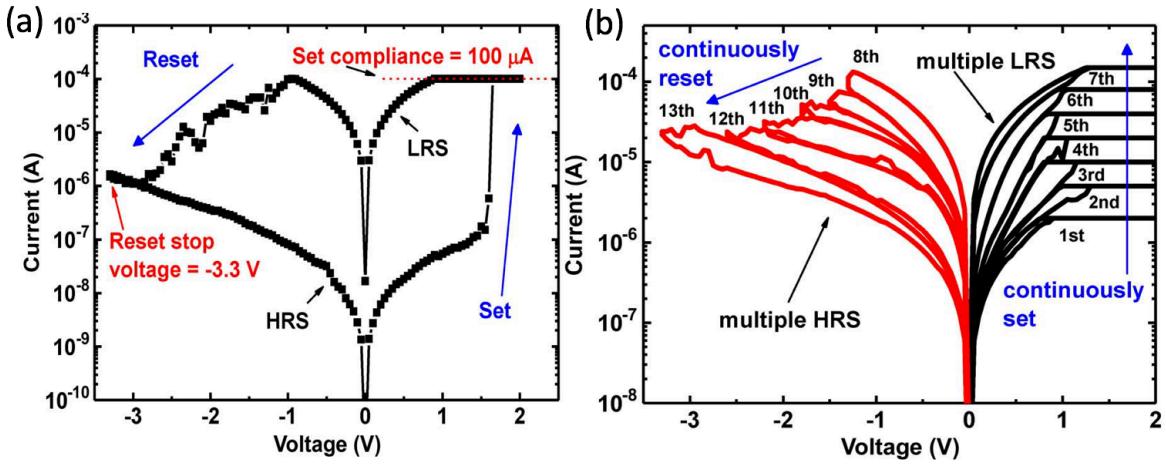

Figure 1.6 (a) The typical switching characteristics of a bipolar TiN/HfOx/AlOx/Pt memristor cell showing binary memristance states. (b) Different memristance states can be obtained by using different set compliance current. Adapted from [36].

Phase-change memory (PCM), which exploits the resistance difference between the crystalline and amorphous phases, is another memristive device [37, 38]. As shown in Fig. 1.7, to RESET the device, a high enough current is necessary to melt the material and a rapid quenching is required to keep the material in the amorphous state; to SET the device, a lower current is used for a longer time to allow time for recrystallization [37].

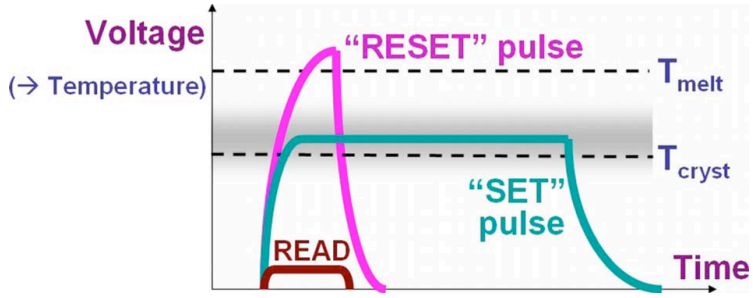

Figure 1.7 Operating mechanism of a phase change memory [37].

Although resistive memristors and PCM have the potential to be used for high-density analog synapses, they have a main drawback: the added material and process complexities (Table 1.2).

This adds a significant cost especially for small-scale memory. Furthermore, the implications of adding these materials into a conventional CMOS process is yet to be fully understood.

Table 1.2 Examples of material systems used for resistive and phase-change memory. Adapted with modification from [44].

Synapse Type	Material System
Resistive memory	TiO_x/HfO_x [36]
	PCMO [39]
	TiO_x [40]
	WO_x [41]
Phase-change memory	$Ge_2Sb_2Te_5$ [Kuzum 42]
	$Si_{16.4}Sb_{32.5}Te_{51.1}$ [43]

Besides two-terminal memristors, floating-gate transistors have received increasing interest recently. For example, using a commercial 180-nm NOR flash memory technology, Guo *et al*. reported a prototype inference engine which can classify a Modified National Institute of Standards and Technology (MNIST) handwritten digit in less than 1 μs with an energy consumption of ~ 20 nJ. Both numbers are more than three orders of magnitude better than those of the 28-nm IBM TrueNorth digital chip for the same task at a similar fidelity [45]. There are also a few startup companies which are promoting floating-gate transistors for low-power artificial intelligence (AI) applications [46, 47]. However, the downsides of flash-memory-based analog synapses are also evident: additional processes are required, and the device are not logic-voltage compatible. For example, in [45], about 30% of the total active area is occupied by level shifters because of the high operating voltage.

We have in this Section briefly reviewed a few popular candidates for analog synapses. The goal is to understand the limitations of these approaches, which are summarized below:

- CMOS process compatibility: new materials and processes are most of the time inevitable, regardless of the technology.
- CMOS logic-voltage compatibility: high voltage is typically required for the flash memory.

If a CMOS-compatible (or ideally, CMOS-only) analog memory device exists, and the device can be programmed with logic-compatible voltages, these problems can all be addressed. Fortunately, such a device does exist – any logic transistor with a high-k-metal-gate in advanced-node technologies will suffice. The charge-trapping phenomenon in the high-k gate dielectric can be exploited for memory applications. The transistors to be used this way are called charge-trap transistors (CTTs). In the next two Sections, we will first describe the operating mechanism of CTTs, and then discuss the value propositions of CTTs for analog synapse applications.

1.4 CTT basics

Around 2000s, the relentless device scaling finally led to a very thin gate oxide in transistors, resulting in unacceptably high gate leakage through SiO_2. High-dielectric-constant materials, i.e. high-k materials, were therefore proposed to provide the transistors with the same ON-current while not significantly reducing the gate-oxide thickness [48−50]. An undesirable feature of these high-k dielectrics however, is that there are inherent oxygen vacancies in the material, acting as the charge-trapping centers. When devices are biased at a high gate voltage for a prolonged time, these oxygen vacancies will trap electrons in them and gradually increase the threshold voltage (V_T) of the device (Fig. 1.8). In other words, the V_T changes with time and is unstable. This effect is traditionally known as the bias temperature instability (BTI), a nuisance that is to be eliminated for an ideal process [51−56]. Moreover, because it is a universal feature of high-k gate dielectrics, virtually all advanced-node technologies beyond 32 nm have this problem.

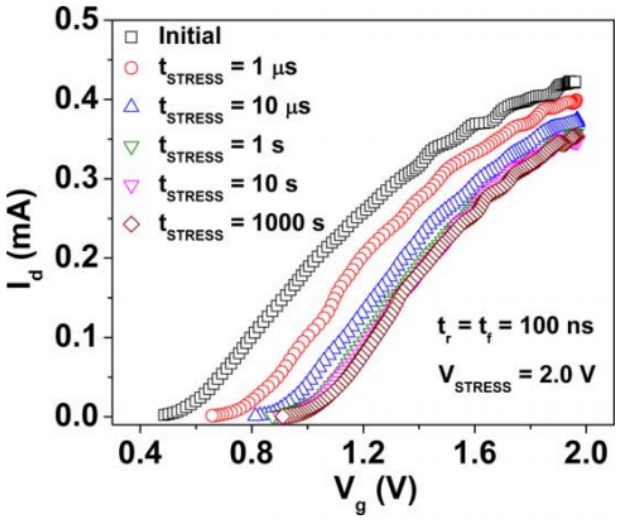

Figure 1.8 Single-pulse transfer characteristics measured after different gate stress conditions. Adapted from [56].

However, it has recently been demonstrated that, the charge-trapping phenomenon by the oxygen vacancies in HfO_2 gate dielectric, when utilized properly, can be used as a mechanism for nonvolatile memory applications [57–60]. It is found that, if a drain bias is present when the gate pulse is being applied, the charge-trapping behavior can be stabilized. Fig. 1.9 shows the V_T change in a pulsed voltage ramp sweep (PVRS) measurement for different drain biases, where 10-ms gate programming pulses with increasing amplitude are applied [57]. In Fig. 1.9(a), we can observe that the maximum ΔV_T one can obtain first increases with an increasing V_D and then decreases after a certain V_D (~ 1.3 V in this case) because the device is easier to breakdown [61]. Furthermore, Fig. 1.9(b) shows the improved retention at a higher V_D during programming. The extrapolated charge loss after ten years when the device is baked at 65 °C is less than 10% at a programming V_D of 1.3 V, whereas it is as much as over 30% at a programming V_D of 0.5 V. This behavior is drastically different from BTI, where the ΔV_T is a few tens of mV and vanishes when the gate bias is removed. This behavior has been carefully studied and attributed to the high channel temperature during programming, which significantly reduces the capture/emission times of deep traps. After the

programming pulses are removed, the device rapidly cools down, leaving those filled deep traps a long time to de-trap, and hence resulting in a long retention time.

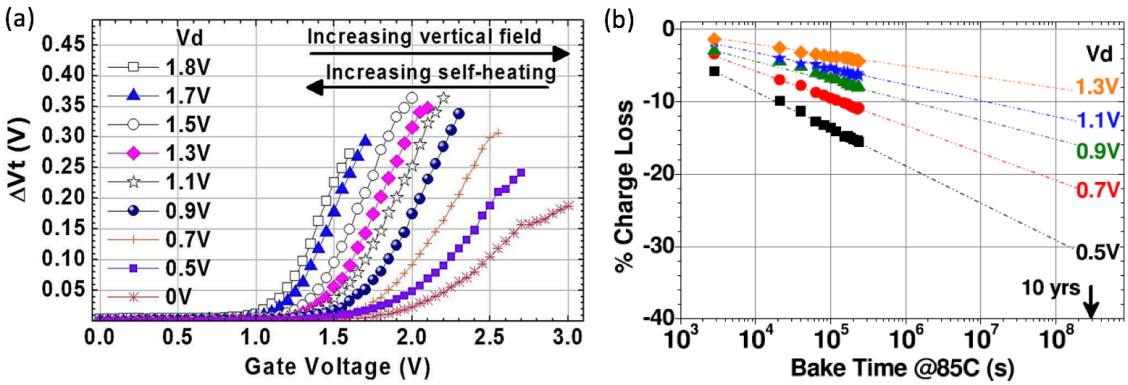

Figure 1.9 High-k charge-trapping phenomenon with a drain bias. (a) PVRS measurement for different drain biases. (b) Higher drain biases during programming not only increases ΔV_T, but also improves retention. Adapted from [57].

Taking advantage of this self-heating-enhanced charge-trapping behavior in HfO_2, GlobalFoundries has recently commercialized CTTs for digital multi-time programmable memory. Table 1.3 summarizes some of the products that have been prototyped in different technology nodes [60]. Note that the CTT technology has been demonstrated in both planar and FinFET nodes, bulk and SOI substrates alike.

Table 1.3 A summary of prototype CTT-based multi-time programmable memory (MTPM). Adapted from [60].

Technology	32nm SOI	22nm SOI	14nm FinFET bulk
Cell	$0.109\mu m^2$ with 1.2nm Gox NMOS	$0.144\mu m^2$ with 1nm Gox NMOS	$0.1411\mu m^2$ with FIN NMOS
Macro density	80Kb	64Kb	40Kb
Density/mm^2	~2Mb/mm^2	~2.5Mb/mm^2	~1.3Mb/mm^2
Activation energy	~1.35eV	~2.4eV	~1.6eV
Design-assist circuits	Yes	No	Yes
Number of cells / BL	256	256	128
Sacrificial WLs and BLs	Yes	No	Yes
Redundant WLs and BLs	Yes	No	Yes

1.5 Value propositions of CTT: Motivation

It should be clear by now that, the CTT is a good candidate and worth exploring for analog synapses in various neuromorphic computing systems targeting fast and low power cognitive tasks. Compared to other analog synapse alternatives, CTTs possess the following advantages:

- It is CMOS-only. Unlike resistive/phase-change RAM or flash technology, no extra materials or processes are necessary. This has an obvious advantage: the development cost of the technology is very low – any advanced-node logic transistors can be used as CTTs straight out of the fab, and existing IP designs in an older node can be easily ported to a new node.
- It is a three-terminal device, providing flexibility in the control of the device and avoiding the use of access transistors for two-terminal emerging memory devices.
- It only uses logic compatible voltage below 2.5 V. This is a significant advantage especially compared to flash-based analog synapses.

However, one thing to keep in mind is that the CTT is not a high-density memory device because a relatively wide device (> ~ 200 nm) is required for efficient charge-trapping to take place. From the cost perspective, this limits the memory size to a few tens of MBs (see Rows 1–3 in Table 1.3 for examples), which is enough to implement most of the neural networks. Therefore, it is practical to build very useful hardware using CTT-based analog synapses.

1.6 Dissertation organization

In this Chapter, we have provided a brief overview on the background of neuromorphic computing, pointed out that there are different types of neuromorphic systems based on their purposes: to simulate the brain or to perform fast and low power cognitive tasks, and revealed that analog synapses are essential for the second-type neuromorphic system because of their capability of enabling in-memory computation and relatively high density compared to SRAM. Various analog synaptic devices, including resistive and phase-change RAM, as well as flash memory, are discussed. CTTs are then introduced for their apparent advantages of being CMOS-only and logic-voltage-compatible. The operating mechanism of CTTs, namely, the charge-trapping phenomenon

enhanced and stabilized by a drain bias during programming, is described. CTT value propositions and the motivation of this dissertation are discussed.

In Chapter 2, we will discuss the analog programming characteristics of CTTs most pertinent to neuromorphic applications. In particular, the analog retention, fine-tuning of individual CTTs, the spike-timing dependent plasticity, and the weight-dependent plasticity are discussed. Variation of CTTs is also characterized using a custom-built CTT array.

In Chapter 3, two algorithms for unsupervised learning, namely, winner-takes-all (WTA) clustering and temporal correlation detection, are investigated, using CTTs as the analog synapses. For each algorithm, the background is first reviewed, and implementations using other analog synapse alternatives are discussed. The implementation using CTTs as the analog synapses is then studied and system performance evaluated using experimentally measured CTT characteristics. Experimental demonstrations of both algorithms are then presented using custom-built CTT arrays.

In Chapter 4, the use of CTTs as analog synapses in an inference engine is studied. We first consider the configuration of using CTTs for a fully connected neural network. The fine-tuning of CTT weights in an array setting is studied since it is anticipated to be different from that of discrete, standalone devices because of the half-selection, thermal disturbance by adjacent cells, etc. The robustness of the inference engine to weight variations is also discussed. In Section 4.5, the comparison between CTT and other analog memory devices is discussed.

Finally, the dissertation is concluded with an outlook of the future of CTT-based neuromorphic systems.

2. Characterization of CTT for Analog Synapses

2.1 Use of CTTs as Analog Synapses

Besides being used as a multi-time-programmable memory where only two states are utilized, CTTs hold great potential for analog memory as well. This is primarily for two reasons: 1) The amount of trapped electrons in the high-k gate dielectric can be finely modified by applying lower and shorter programming pulses (compared to digital applications); and 2) CTT-based synapses can be operated in the subthreshold region, providing a large dynamic range with a relatively small change to ΔV_T. For example, ΔV_T of 250 mV can be readily achieved by applying a 10-ms trapping pulse with $V_G = 2.8V$ and $V_D = 1.5$ V, providing a subthreshold current change of more than 1000× (Fig. 2.1).

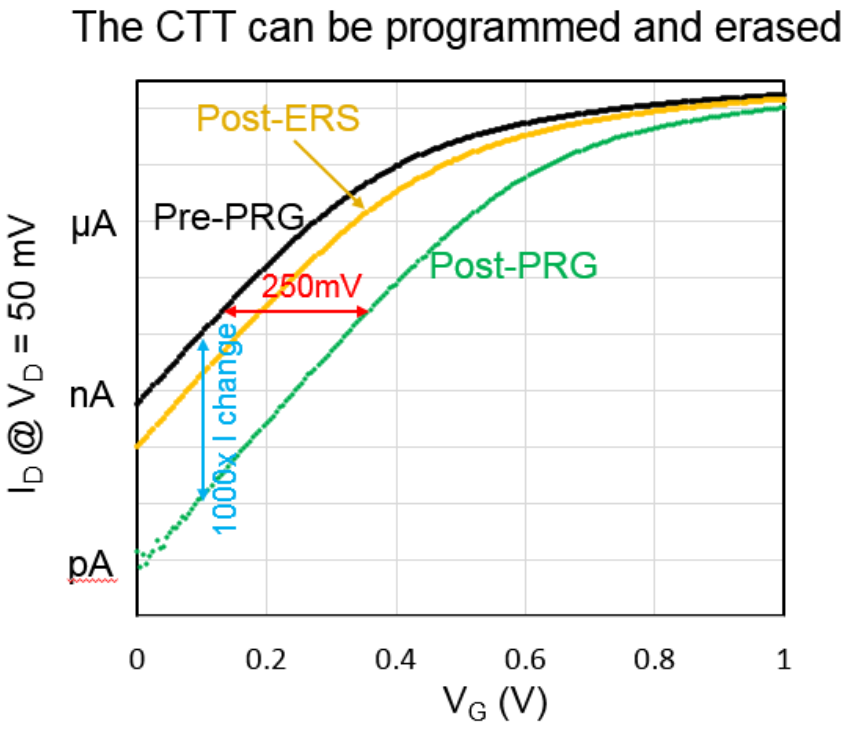

Figure 2.1 Transfer characteristics of a CTT before and after programming, and also after erase. ΔV_T of 250 mV and a corresponding subthreshold current change of 1000× can be achieved.

After being programmed, a CTT can also be erased by a negative gate-to-source and gate-to-drain voltage (Fig. 2.1), although not completely. This provides an extra degree of controllability in accurate programming of the device. In addition, this ability to be erased is desired for applications such as winner-takes-all (WTA) clustering which will be discussed in Chapter 3.

As shown in Fig. 2.2, when a weight is to be decreased, trapping pulses are applied to the gate of the CTT, increasing its V_T and therefore decreasing the weight; when a weight is to be increased, de-trapping pulses are applied, reducing its V_T and therefore increasing the weight.

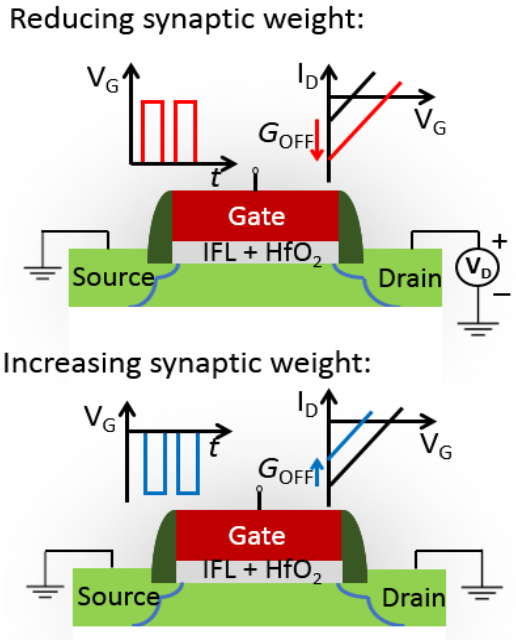

Figure 2.2 Illustration of the use of subthreshold current as the synaptic weight and the pulsing schemes to reduce or increase the weight.

Using the CTTs this way as an analog synapse finds various applications in neuromorphic computing. In this Chapter, we will examine the spike-timing dependent plasticity and weight-dependent plasticity, which are required characteristic for adaptive learning and WTA clustering, respectively. The application of this configuration for an inference engine will be discussed in Chapter 4.

2.2 Device characteristics in 22nm fully depleted SOI

In this Section, CTT behavior is characterized in the 22nm fully depleted silicon-on-insulator (22FDX) technology. Specifically, super-low-V_T transistors (a standard logic device offering) are characterized. The same device is used later in the experimental demonstration of unsupervised learning as well as a dot product engine for inference engines.

The device structure is schematically depicted in Fig. 2.3, where the gate-oxide is 1.25-nm-thick, SOI is 6-nm-thick, and the BOX is 20-nm-thick. One notable feature is the N-well (NW), which provides an extra degree of tunability of V_T. In practice, NW is shared by an array of devices and is contacted by a via through locally etched BOX.

Caution must be taken when programming pulses are sent from the semiconductor parameter analyzer to the device. Because the cabling has a capacitance as large as 1 nF, and the wiring resistance from the pads to the device terminal can be as large as 100 Ω, ultrashort pulses should be avoided. Here, the shortest pulse is 20-μs long, ensuring a high fidelity of the actual voltage that is applied to the device terminals.

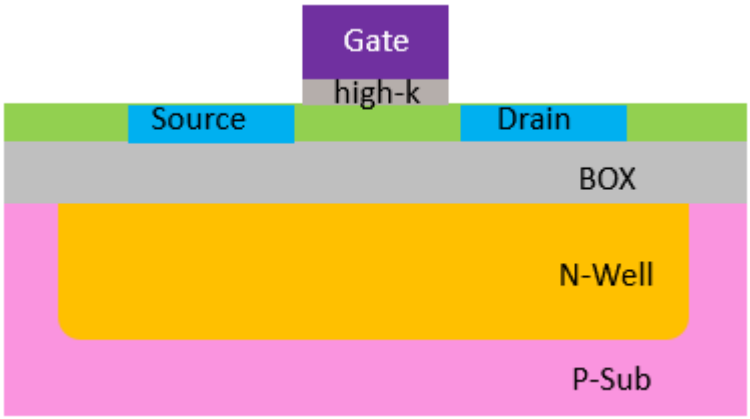

Figure 2.3 Structure of the super-low-V_T CTT under characterization.

2.2.1 Programming and erase behavior

Fig. 2.4 shows the I_D-V_G curves of a CTT before programming, after programming, and after erase. During programming, a 10-ms, 2.6 V pulse is applied to the gate with V_D = 1.3 V, resulting in 103 mV V_T shift (ΔV_T); during erase, a 30-ms, −2.8 V pulse is applied to the gate with both source and drain grounded. An important feature to note here is that, there is a ~10 mV ΔV_T residual after erase. More negative V_G and longer erase time are necessary to reduce the V_T further. Measurements with source/drain reversed (dashed curves) do not show significant differences, indicating that the charge-trapping is not highly asymmetric along the channel direction.

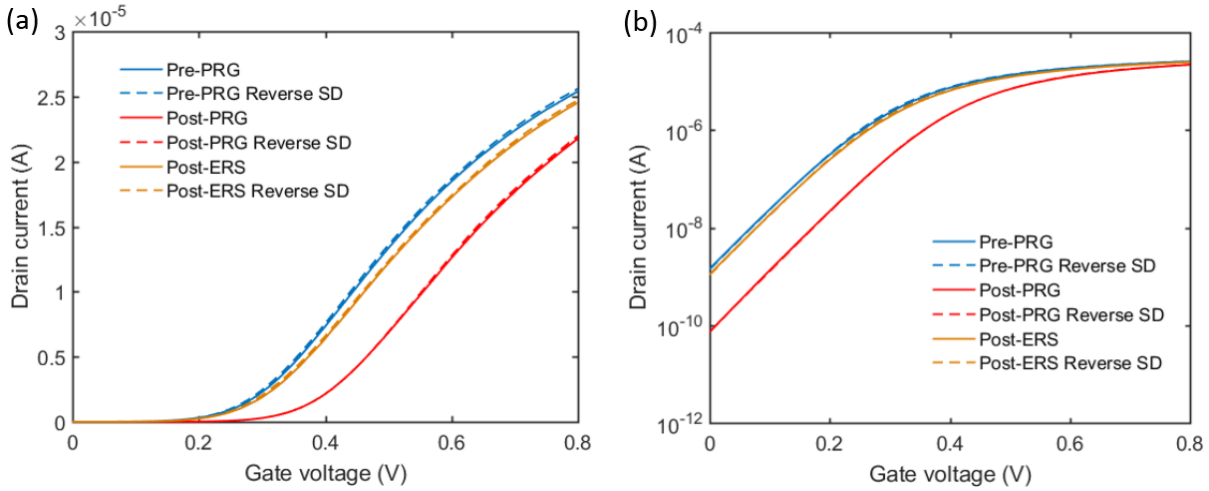

Figure 2.4 Typical I_D-V_G curves for unprogrammed (blue), programmed (red), and erased (yellow) CTT. Dashed curves are measured with source and drain flipped. W/L=170nm/20nm.

Fig. 2.5 shows the I_D-V_G curves of a CTT programmed to intermediate states. These states are achieved by applying lower and shorter gate voltage during programming. It is worth noting that, as the device is programmed, the subthreshold swing does not degrade (Fig. 2.6).

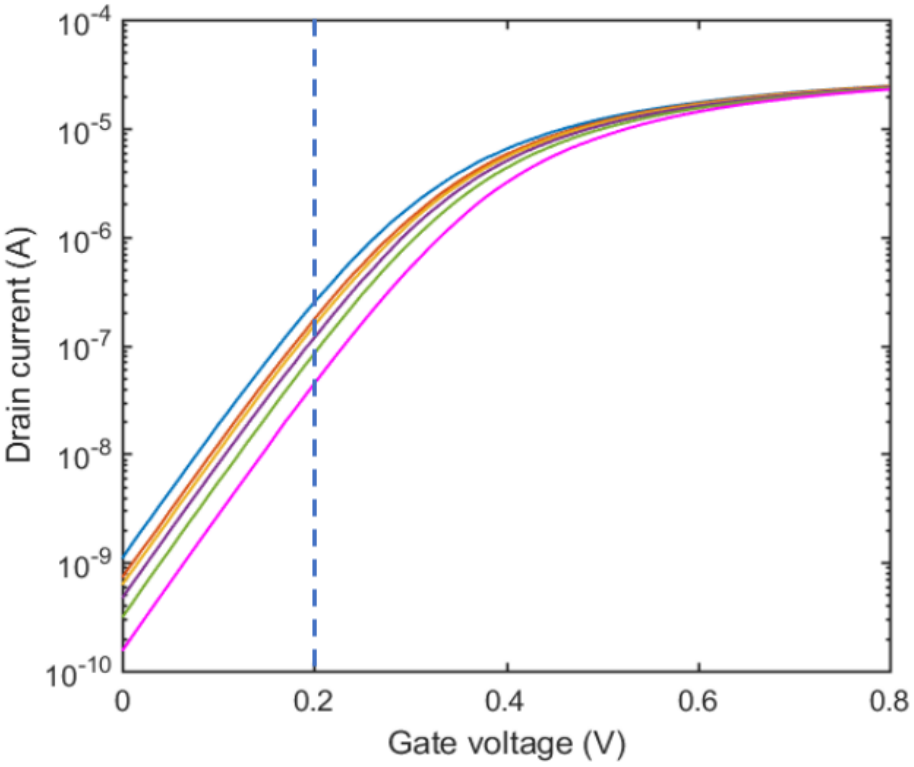

Figure 2.5 I_D-V_G curves of CTT at multiple intermediate states.

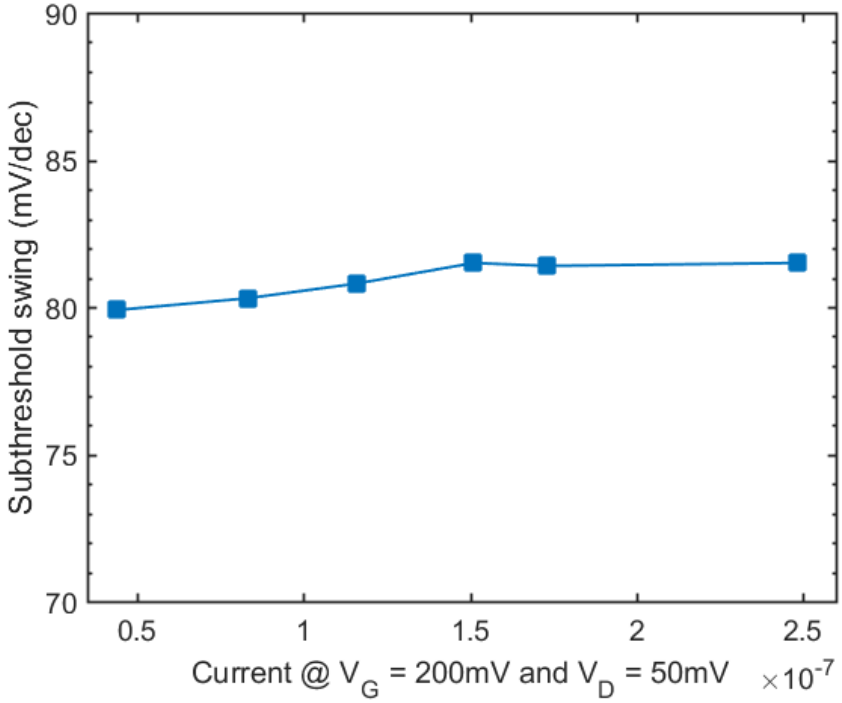

Figure 2.6 Subthreshold swing of CTT as the device is programmed.

2.2.2 Tunability offered by the N-well

As mentioned above, the super-low V_T CTT being studied here is a flip-well device, with an NW underneath the BOX. The device characteristics can be modified by the NW bias (Fig. 2.7) with a sensitivity of $\Delta V_T/\Delta V_{NW} = -88$ mV/V. With regards to the application of CTT as an analog synapse, the NW bias can serve two purposes: 1) During programming, a positive bias reduces the V_T and therefore increases the channel current, improving the programming efficiency without raising the gate voltage; 2) After programming, NW bias can be used to collectively increase or decrease the synaptic weight without added complexity in the circuitry. We will focus on the first point here.

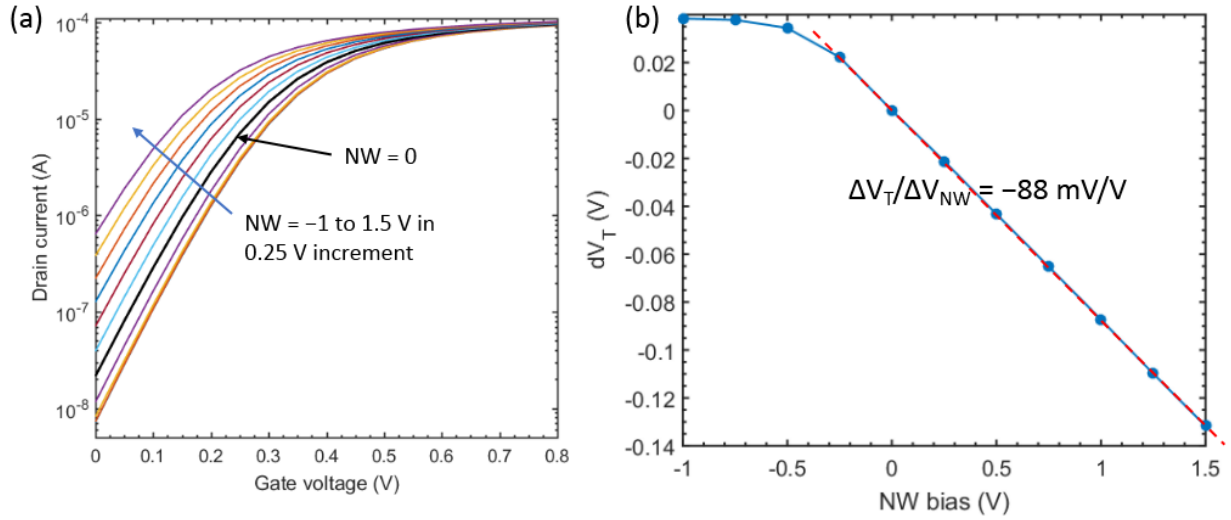

Figure 2.7 Effect of NW on device V_T. (a) I_D-V_G curves for different NW biases. (b) ΔV_T as a function of V_{NW}.

Fig. 2.8 shows the effect of NW bias during programming. It is observed that the V_T increases more with a larger NW bias during programming (up to 12 mV, or 13%), and this improvement saturates for NW bias larger than 0.5 V.

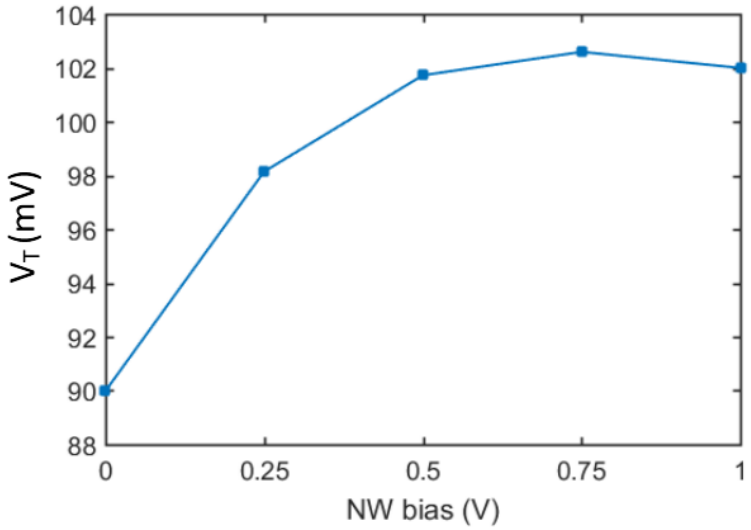

Figure 2.8 Effect of NW bias during programming. ~13% ΔV_T improvement can be achieved with a 0.5 V NW bias.

2.2.3 Analog retention and fine-tuning

CTTs have been qualified as multi-time-programmable digital memory for 10 years at 105−125 °C. The relatively large V_T difference (therefore current difference) is sensed by a slew sense amplifier. In the analog regime, however, the states are ideally continuous, making the quantification of retention more challenging.

There have been a few reports on the analog retention behavior of other analog memory devices in the literature. Fig. 2.9 shows two examples of TiN/HfO_x/AlO_x/Pt and WO_x, respectively. In Fig. 2.9(a), the device resistance can vary for more than 100% [36]; in Fig. 2.9(b), the device current decays significantly within a period of three minutes [63].

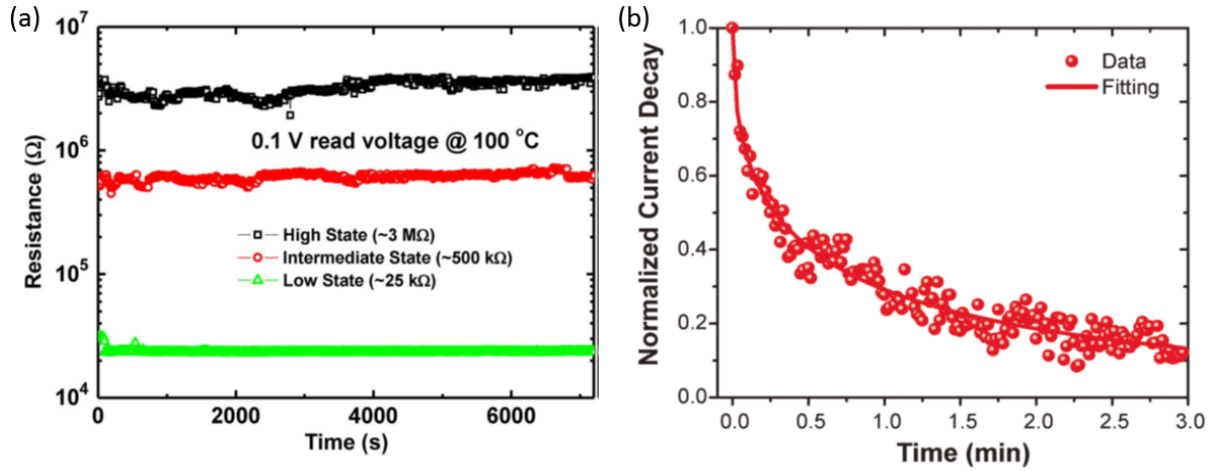

Figure 2.9 Analog retention characteristic for two memristors. (a) TiN/HfO_x/AlO_x/Pt cell [36]. (b) WO_x cell [63].

We show in Fig. 2.10 the retention of multiple analog states in CTT. Here, the device is programmed with a train of 315 50-µs pulses. In these pulses, V_G increases from 1 V to 2.57 V in 5 mV increment while V_D is held at 1 V. The pulses are applied in 21 segments (each with 15 pulses) and the inference current at V_G = 200 mV and V_D = 50 mV is measured for one hour after each segment. Two features can be observed: First, after each segment of programming pulses, the inference current gradually increases within one hour, apparently due to de-trapping of trapped electrons; second, not all current increase resemble the shape of an exponential function, indicating that there is no single time constant.

A detailed look at Fig. 2.10 reveals that, the lower current (or the higher V_T) the CTT is programmed to, the more up-drift there is after programming. For example, when the CTT is programmed with 21 gate pulses from 1 V to 1.1 V (Fig. 2.11(a)), the current increase in one hour is only 0.87%; when the current is programmed to 330 nA using 315 gate pulses (Fig. 2.11(c)), the current increase in one hour is 64 nA, or 19.3%, corresponding to a V_T recovery of approximately 7 mV. Although the 7 mV V_T recovery is usually not an issue for digital memory applications, it may be of concern to analog memory since the current is changed significantly.

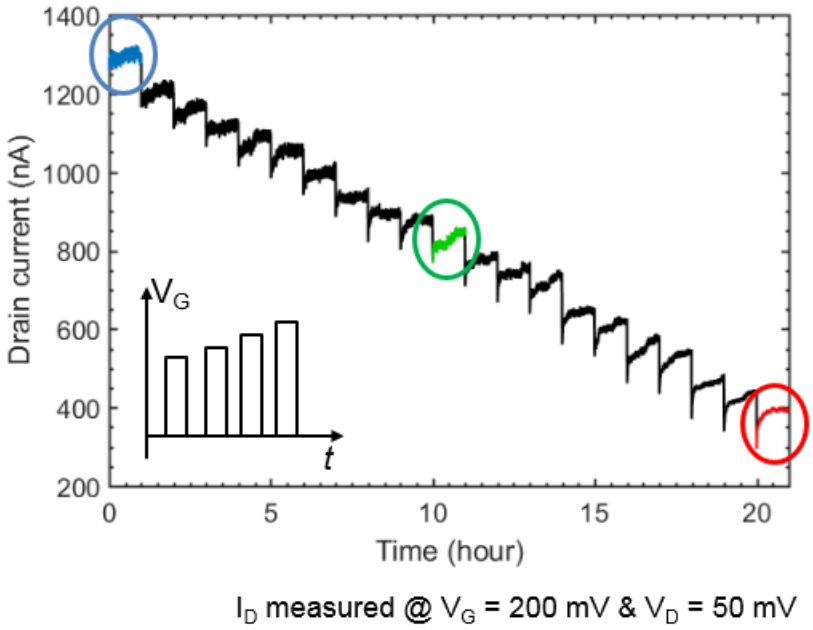

Figure 2.10 Analog retention characteristic of CTT. The CTT is programmed by 315 50-µs gate pulses with the voltage increasing from 1 V to 2.57 V in 5 mV increment. The drain current is measured at V_G = 200 mV and V_D = 50 mV for one hour after each 15-pulse segment. The zoom-in of three circles is shown in Fig. 2.11.

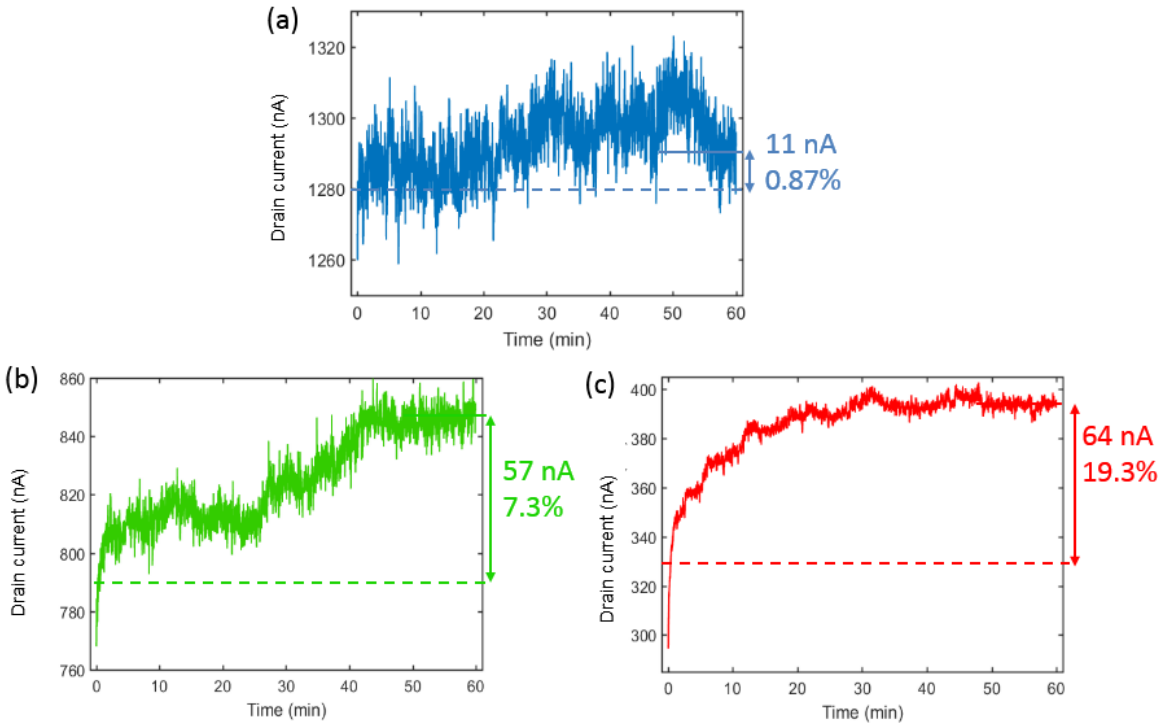

Figure 2.11 The current recovery for 1 hour after programming for different current levels. (a) the 1st, (b) the 11th, and (c) the 21st segment in Fig. 2.10, corresponding the different colors.

Fig. 2.12 shows the relationship between the current increase in 1 hour and the current immediately after programming. An approximate linear relationship is fitted to be

$$\Delta I = -0.075 \times I_{RA} + 114.5 \text{ (nA)} \tag{2.1}$$

where ΔI is the current increase and I_{RA} is the current immediately after programming.

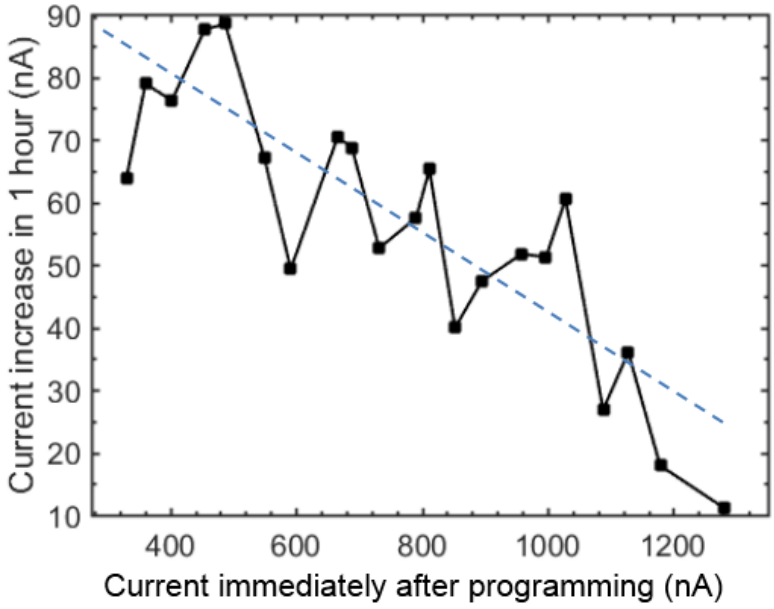

Figure 2.12 The relationship between the current increase in 1 hour and the current immediately after programming.

The implications of such analog retention behavior in accurately programming the analog memory are two-fold: First, the CTT needs be over-programmed; second, a block write-verify algorithm needs to be adopted: the CTTs in an array should be programmed one block at a time, and the verification of the inference current should take place after the block-write to allow for the stabilization of the current.

Using the fitted behavior in Eq. (2.1), a CTT is programmed to discrete target current levels from 1.2 μA to 350 nA. For each target current, the device is over-programmed by the amount predicted by Eq. (2.1) and the current is measured after 1 hour. The relationship between the programmed current after 1 hour and the target current is shown in Fig. 2.13(a). It is clear that the

dots almost lie on the ideal $y = x$ line, indicating a very good programming result. Indeed, the standard deviation of the difference between the programmed and the target current is less than 1.6% of the range of the target current.

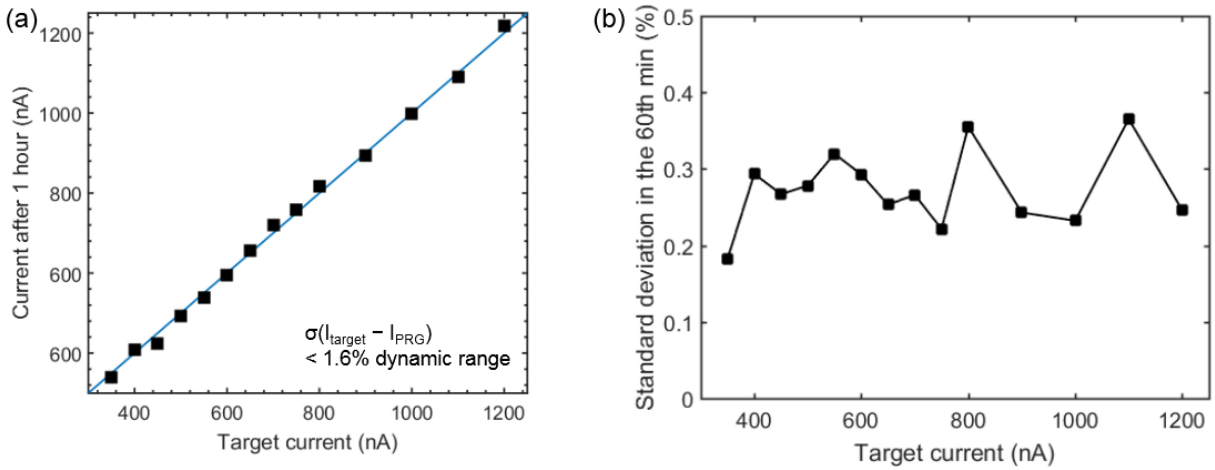

Figure 2.13 (a) The programmed current (after 1 hour) vs. the target current, with over-programming of the CTT according to Eq. (2.1). (b) The standard deviation of the current measured in the last minute of 1 hour vs. the target current. It can be seen that the current is very stable after 1 hour.

Fig. 2.13(b) shows the standard deviation of the current measured in the last minute of 1 hour as a function of the target current. The current is stable regardless of the target current. In Section 4.3, the stability of the current after programming will be revisited in the array context.

2.3 Spike-timing dependent plasticity (STDP)

The CTT can be employed to realize plastic synapses that possess spike-timing dependent plasticity (STDP) (Fig. 2.14) [64]. STDP is a key memory and learning mechanism in biological synapses: if the pre-synaptic neuron repeatedly fires right before the post-synaptic neuron does, the connectivity between the two neurons is strengthened (long-term potentiation, LTP); on the contrary, if the pre-synaptic neuron repeatedly fires right after the post-synaptic neuron does, the connectivity is weakened (long-term depression, LTD). In other words, the change in the connectivity between neurons depends on the timing-difference between pre- and post-synaptic

neurons. Since the V_T change after programming depends on the amplitude of the gate voltage, if a pulsing scheme can translate the timing difference to the amplitude, it should be able to demonstrate STDP in CTT. For example, in [36] and [42], discrete pulses are applied to the device to mimic the STDP characteristic. The idea is that, no pre- or post-synaptic pulse alone can significantly modify the state of the device, and only when the two pulses are simultaneously present can the device state be changed.

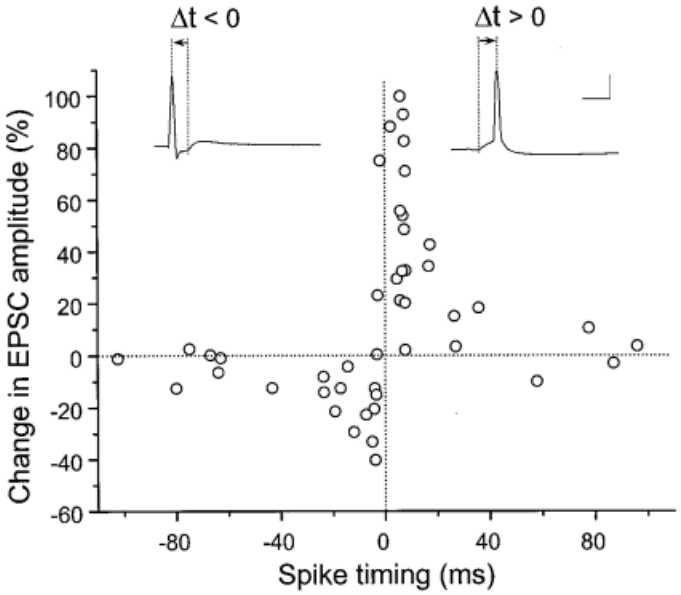

Figure 2.14 Typical spike-timing dependent plasticity exhibited by synapses [64].

A similar pulsing scheme depicted in Fig. 2.15(b) is adopted here but the pulses are continuous instead of a train of discrete pulses in [36] and [42]. This makes circuit design much simpler. With this scheme, the pre-synaptic signal is connected to the source of the CTT, the post-synaptic signal is connected to the gate of the CTT, and the drain is biased at a constant voltage (Fig. 2.15(a)). The pre- and post-synaptic spikes take the same form: The voltage first drops to 0 from V_1 for a period of t_1, returns to V_p and then gradually decays to V_1 again with a slope of S (V/s).

Fig. 2.16 illustrates in detail how the change in the CTT channel conductance is modulated by the timing difference ($\Delta t = t_{pre} - t_{post}$) between the two pulses, mimicking the STDP behavior. In

both cases ($\Delta t > 0$ and $\Delta t < 0$), the time segment 3 is the main programming pulse. For example, during time segment 3 in the LTD case ($\Delta t = t_{pre} - t_{post} > 0$, Fig. 2.16(a)), the source of the CTT (pre-synaptic signal) is grounded and a high voltage is applied to its gate (post-synaptic signal, whose amplitude decreases as Δt increases) while a drain voltage is being applied. The CTT is programmed to have a higher V_T and therefore a smaller synaptic weight. Note that the amplitude of the gate pulse is $V_p - S\Delta t$, which, as Δt increases, decreases and results in a smaller weight change.

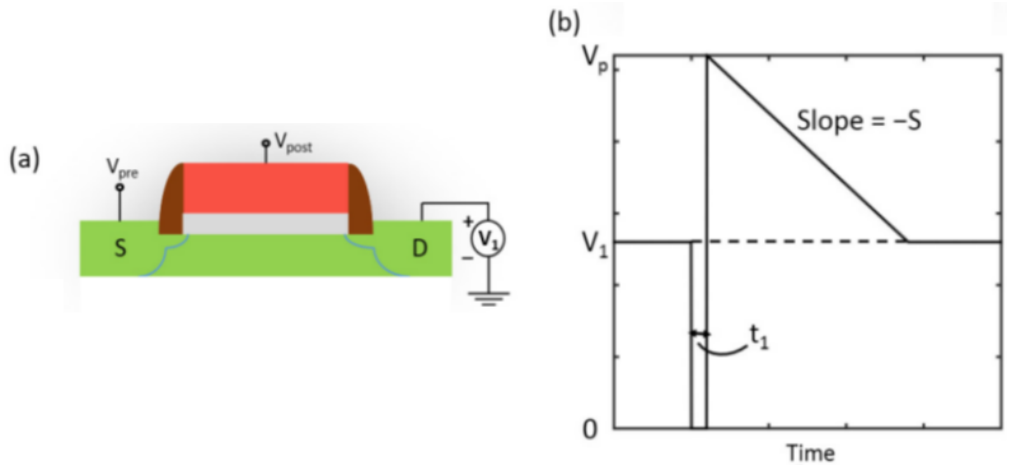

Figure 2.15 Pulsing scheme to demonstrate STDP in CTT. (a) The connection: the pre-synaptic signal is connected to the source of the CTT, the post-synaptic signal is connected to the gate of the CTT, and the drain is biased at a constant voltage. (b) The pulsing scheme: The voltage first drops to 0 from V_1 for a period of t_1, returns to V_p and then gradually decays to V_1 again.

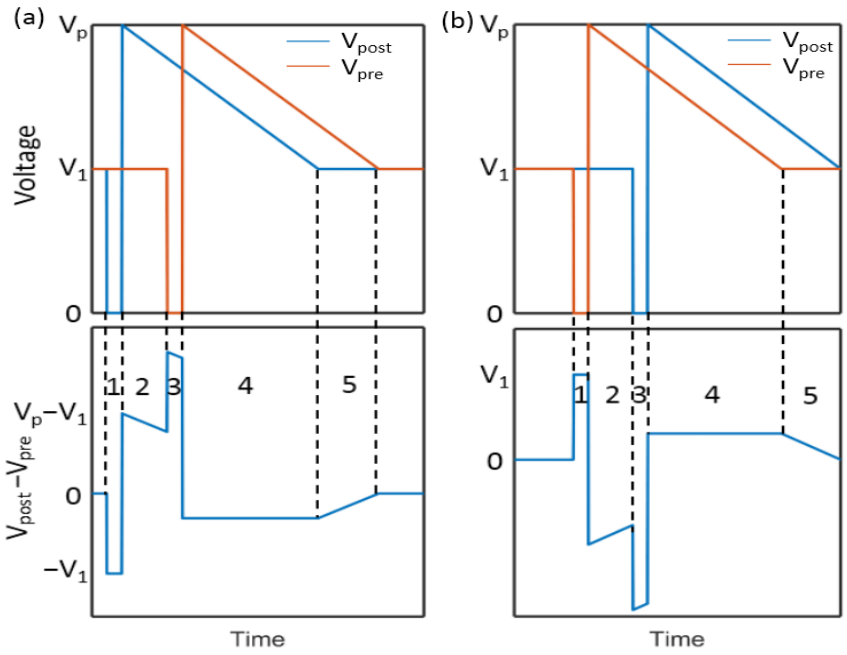

Figure 2.16 Detailed breakdown of the applied pre- and post-synaptic pulses when (a) the post-synaptic neuron fires before the pre-synaptic one ($\Delta t = t_{pre} - t_{post} > 0$), and (b) the post-synaptic neuron fires after the pre-synaptic one ($\Delta t < 0$).

In the experiments, the CTT is first initialized with a program-erase cycle, and then incrementally programmed to an intermediate inference current of approximately 400 nA / 0.3 µm before going through LTD and LTP cycles. The synaptic pulses as shown in Fig. 2.15(b) with V_1= 1 V, t_1 = 1 ms, S = 1.3 V / 15 ms, and varying values of V_p are generated by the waveform generator fast measurement unit (WGFMU) and applied to the CTT. For each timing-difference, the experiment is repeated 10 times to obtain the statistical behavior.

The boxplot of conductance change as a function of the spike timing-difference is shown in Fig. 2.17 for V_p = 2.3 V. It is clear that there is cycle-to-cycle variation, partly due to the physical change in the CTT after each trial. It is also important to note that, as the pre- and post-synaptic pulses are separated by more than ~ 6 ms, the pulses have very small effect on the state of the CTT.

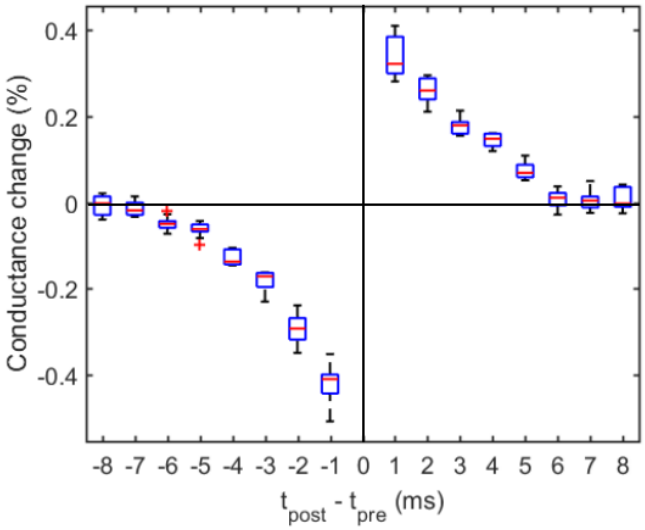

Figure 2.17 Boxplot of the STDP behavior when V_p = 2.3 V.

The average of the conductance change as a function of the timing difference is shown in Fig. 2.18 for different values of V_p (2.3 V, 2.1 V, 1.9 V). It is observed that the conductance change at a given timing-difference, both in the LTP and LTD regimes, increases with the highest programming voltage V_p. A maximum conductance of 34% and – 42% is obtained in the LTP and LTD regimes, respectively, when V_p = 2.3V. To obtain a larger conductance change, higher V_p or longer t_1 is required.

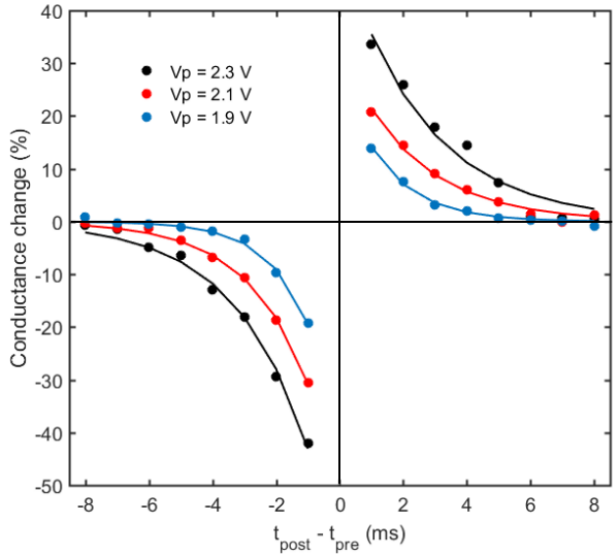

Figure 2.18 Spike-timing dependent plasticity exhibited by CTT. Solid lines are exponential curves fitted to Eq. (2.2).

Fig. 2.18 also demonstrates the tunability of the STDP behavior through changing pulse parameters. The conductance change in both LTP and LTD can be fitted as:

$$\Delta G/G = A_{LTP,\,LTD} \times \exp\left(-|\Delta t|/\tau_{LTP,\,LTD}\right) \tag{2.2}$$

and the values of the fitting parameters $A_{LTP,\,LTD}$ and $\tau_{LTP,\,LTD}$ for different values of V_p are listed in Table 2.1. Both A and τ increase as V_p increases. In practice, V_1, t_1, and S can all be modified to obtain the desired STDP behavior.

Table 2.1 Parameters from the fitted Equation (2.2).

V_p (V)	LTP		LTD	
	A (%)	τ (ms)	A (%)	τ (ms)
2.3	52.25	2.588	−66.47	2.306
2.1	33.21	2.258	−52.01	1.904
1.9	28.26	1.437	−41.77	1.304

2.4 Weight-dependent plasticity

For many cases of unsupervised learning, for example winner-takes-all (WTA) clustering and temporal correlation detection, as will be discussed in Chapter 3, a device characteristic – the weight-dependent plasticity, is desired. It requires that the change of inference current (or channel conductance) be dependent on the conductance itself.

The subthreshold OFF-conductance (G_{OFF}) of the CTT at V_{DS} = 50 mV and V_{GS} = 0 is used as the synaptic weight. In the LTD regime where the weight is to be decreased, a positive gate pulse is applied and electrons are trapped into $HfSiO_x$ through the interfacial layer (IFL), increasing V_T and decreasing G_{OFF} (Fig. 2.19(a)); in the LTP regime, a negative gate pulse is applied and trapped electrons tunnel back into the channel, decreasing V_T and increasing G_{OFF} (Fig. 2.19(b)). In the experiments, a CTT is first pre-programmed to an intermediate starting state by applying a gate

pulse of 2.5 V for 60 μ s with V_D = 1.3 V. The device subsequently goes through four cycles: two LTD and two LTP cycles, with 256 trapping or detrapping pulses in each cycle. In the LTD cycle, G_{OFF} is decreased by a 20-μs, 2.5 V gate pulse with V_D = 1.3 V; in the LTP cycle, G_{OFF} is increased by a 50-μs, – 2.6 V gate pulse with zero drain bias. The resulting G_{OFF} is shown in Fig. 2.19(c) where a reversible and reproducible modification of synaptic weights can be observed. Over 200 levels are achieved for both LTP and LTD regimes with a fine resolution of less than 1 nS for LTP and 0.25 nS for LTD. As we will show later in Section 3.1.2, although the LTD has a smaller dynamic range, it will not affect the convergence of the learning algorithm.

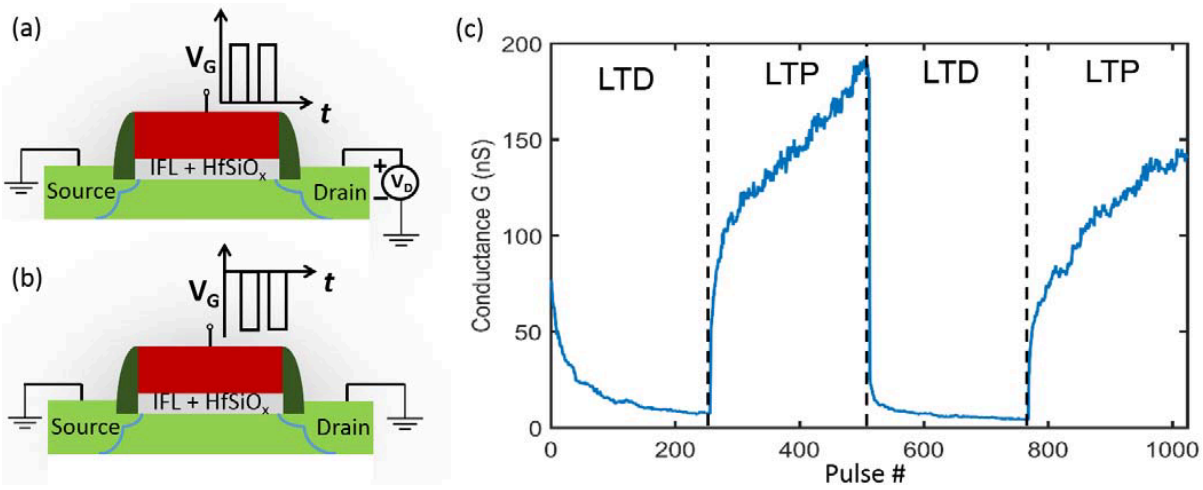

Figure 2.19 Configurations of the CTT in the (a) LTD and (b) LTP regimes. (c) Reversible and reproducible device conductance change through four cycles.

We show in Fig. 2.20(a) the relative G_{OFF} change as a function of G_{OFF} itself when five trapping and detrapping pulses as specified above are applied. It is observed that, in the LTP regime, the relative G_{OFF} increase is smaller when the initial G_{OFF} is larger; on the contrary, in the LTD regime, the relative G_{OFF} reduction is larger when the initial G_{OFF} is larger. The curves corresponding to the LTP and LTD regimes are fitted to exponential and sigmoid functions, respectively, for different programming times (Fig. 2.20(b)). As expected, a longer programming time consistently leads to a larger G_{OFF} change because of the larger V_T change caused by more trapped/detrapped charge [58].

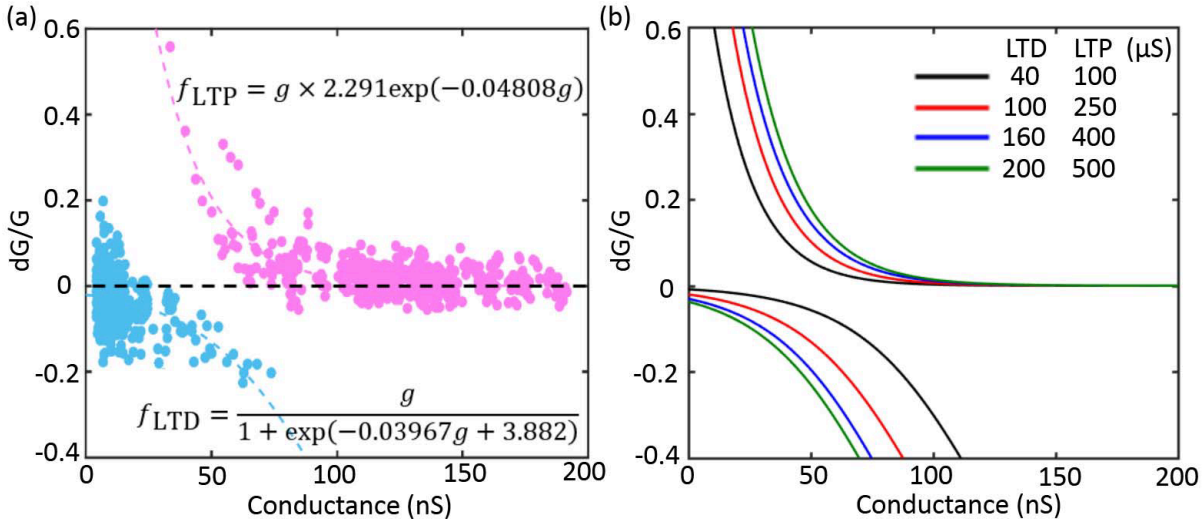

Figure 2.20 (a) The weight-dependent plasticity when five trapping/detrapping pulses are applied in the LTD/LTP regimes, respectively. (b) Fitted curves when pulses of different widths are applied.

2.5 Variation of CTT

We have seen from Section 2.2.3 that the inference current of a CTT gradually increases after the device is programmed, and the increase is larger for a smaller target current. This makes the block write-verify process necessary when programming a CTT array to accurate weight values. In this Section, we show statistical results on the programming behavior of a custom-built twin-cell CTT array with 10 word lines (WLs) and 8 bitline (BLs) to further establish the necessity of the block write-verify approach.

In one programming cycle, 500-μs programming pulses with V_D = 1.2 V and varying V_G (2.1 V, 2.4 V, and 2.7 V) are applied sequentially to the CTTs in the array. After each programming cycle, the inference current I_{inf} of all CTTs in the array is measured at V_G = 200 mV and V_D = 50 mV. To suppress the channel current from other CTTs in the same column, −0.3 V is applied to their gates. The evolution of average V_T and I_{inf} with the number of programming pulses is shown in Fig. 2.21. As expected, both the V_T increase and the I_{inf} reduction are higher for a higher gate

programming voltage. The highest achievable ΔV_T is 115 mV when the device is programmed for a total of 30 ms at V_G = 2.7 V.

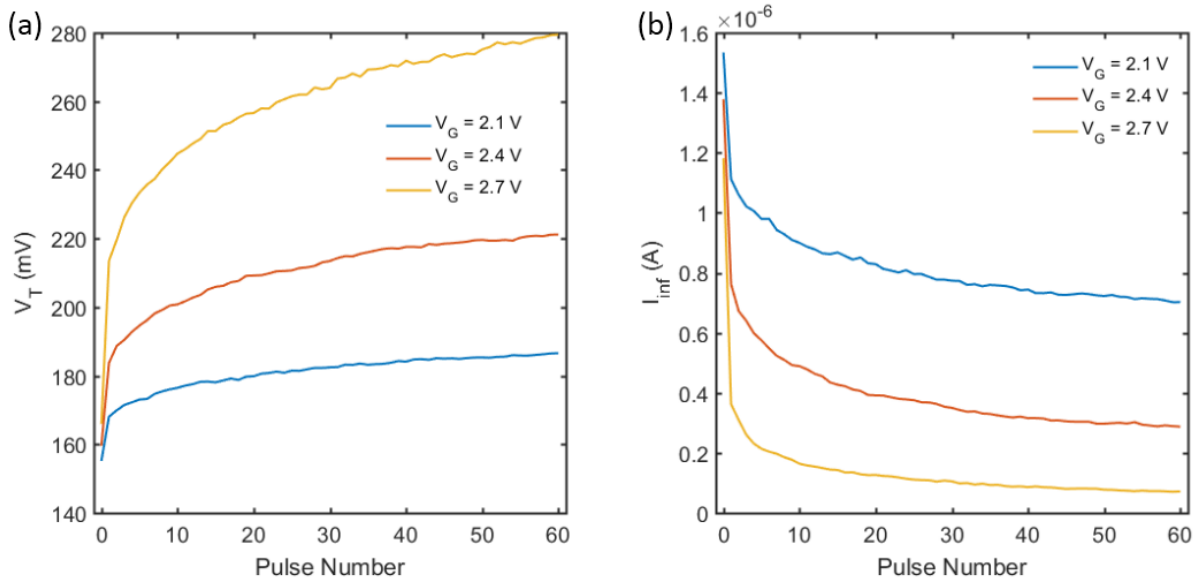

Figure 2.21 The evolution of average (a) V_T and (b) I_{inf} with the number of programming pulses, for different gate programming voltage. Each programming pulse is 500-µs long and the total programming time is 30 ms.

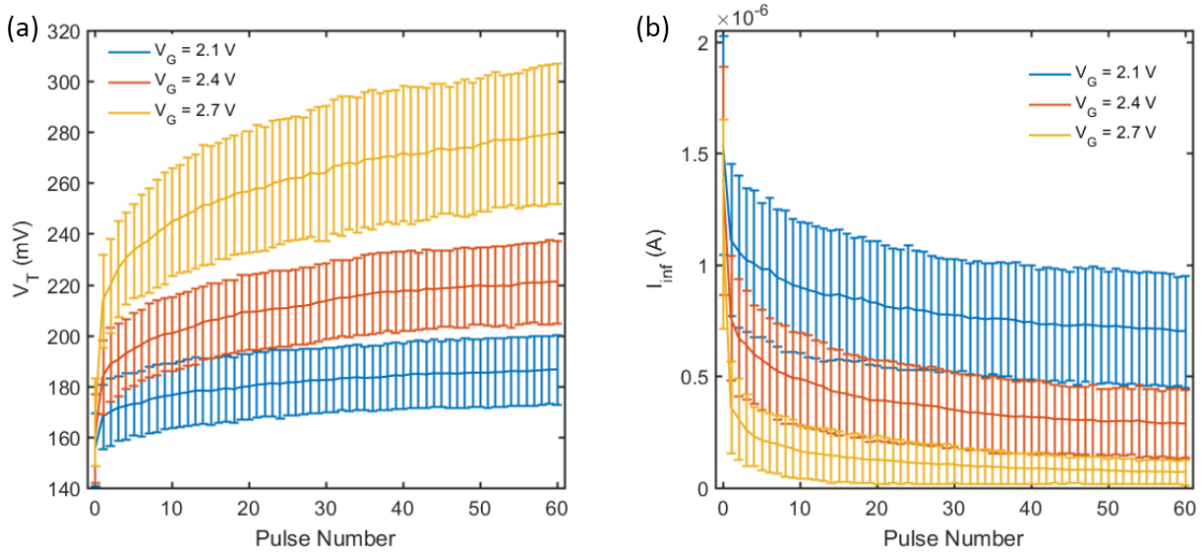

Figure 2.22 Statistics of (a) V_T and (b) I_{inf} with the number of programming pulses, for different gate programming voltage. Each programming pulse is 500-µs long and the total programming time is 30 ms.

Fig. 2.22 shows the average as well as the standard deviation of the V_T and I_{inf}. It is clear that there exists substantial device variation. When the I_{inf} of a device needs to be programmed to a specific value, for example when CTTs are used as synapses in an inference engine described in Chapter 4, although first-order estimate can be made to determine approximately how long a programming pulse is necessary, the I_{inf} needs to be verified after the pulse and a decision to apply or not apply another pulse is made.

3. CTTs in Unsupervised Learning

As discussed in Section 1.1, by conducting in-memory computation, analog-synapse-based neuromorphic systems do not need to constantly fetch data between the memory and the CPU, therefore significantly reducing the power and time required. Various analog synapses have been explored in neuromorphic systems for both unsupervised learning and supervised learning.

In this chapter, two algorithms for unsupervised learning, namely, winner-takes-all (WTA) clustering and temporal correlation detection, are investigated, using CTTs as the analog synapses. It is shown that the algorithms can indeed be implemented with CTTs and are robust to device variations. Experimental demonstration of these algorithms using the custom-built CTT arrays is also discussed.

3.1 Winner-takes-all network

3.1.1 Background

In this era of mass connectivity (social media, internet of things, etc.), a vast amount of data is being generated every day. However, 80% of these data is unstructured, making extracting valuable information out of them a very challenging task. One of the many ways to analyze the data is *clustering* – to categorize data into subgroups such that the data points within each group are "similar" to each other. Note that there is no universal definition of "similar", and the division of data might not be unique. A good example of clustering can be found in social media. People can be categorized into different "professions" according to who they interact with, what their topics of interest are, and even what they eat.

Various algorithms in software have been developed for clustering purposes, including winner-takes-all (WTA) and k-means clustering. These algorithms can also be implemented in hardware [66, 67]. For example, Serb *et al*., in [66] reported a WTA network with TiO_2-based memristors. In their work, a one-layer WTA network with four input neurons and two output neurons is constructed to cluster two simple four-bit binary patterns: 1001 and 0110, and noisy versions of them. The four synapses corresponding to one output neuron are implemented in software and the other four are implemented with memristors (Fig. 3.1(a)). Specialization functions are used to monitor the quality of clustering. As can be seen in Fig. 3.1(b), the two patterns are well separated after ~ 100 pattern presentations, even though there is significant device variation (Fig. 3.1(d)).

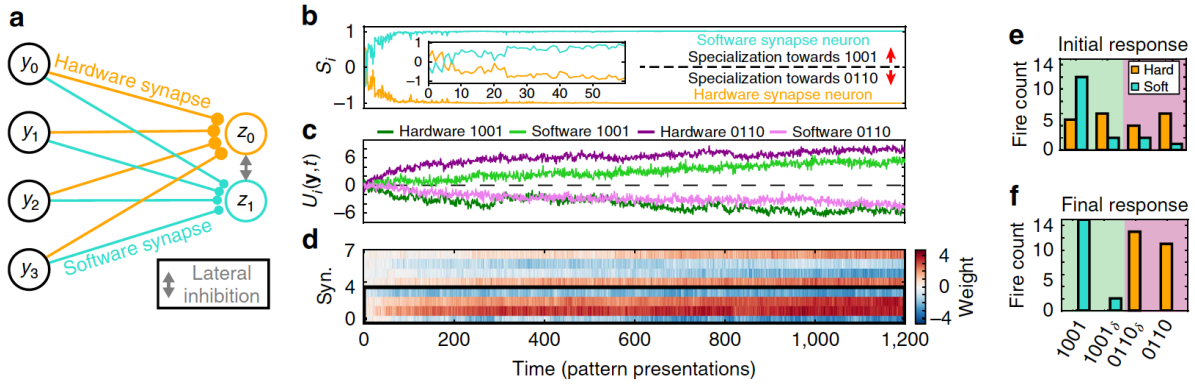

Figure 3.1 Results of a simple 4×2 WTA clustering network implemented by memristors and software [66]. (a) The structure of the WTA network. Four input neurons correspond to the four bits in a pattern, and two output neurons correspond to two clusters. (b) Evolution of the "specialization" function which indicates the quality of the clustering. (c) The evolution of membrane potentials as the network is trained. (d) Evolution of the synaptic weights. (e) and (f) Clustering results before (e) and after (f) training. "δ" indicates a one-bit-flipped version of the pattern.

Kim *et al*. in [67] reported a similar simulation study in 2018. They fabricated a thin-film transistor (TFT)-type NOR flash memory cell with a half-covered floating gate and experimentally demonstrated the LTP/LTD characteristics in it (Fig. 3.2). Using experimental data in the simulation, they then investigated the learning of a single pattern from the MNIST dataset, as well as the learning of ten different patterns.

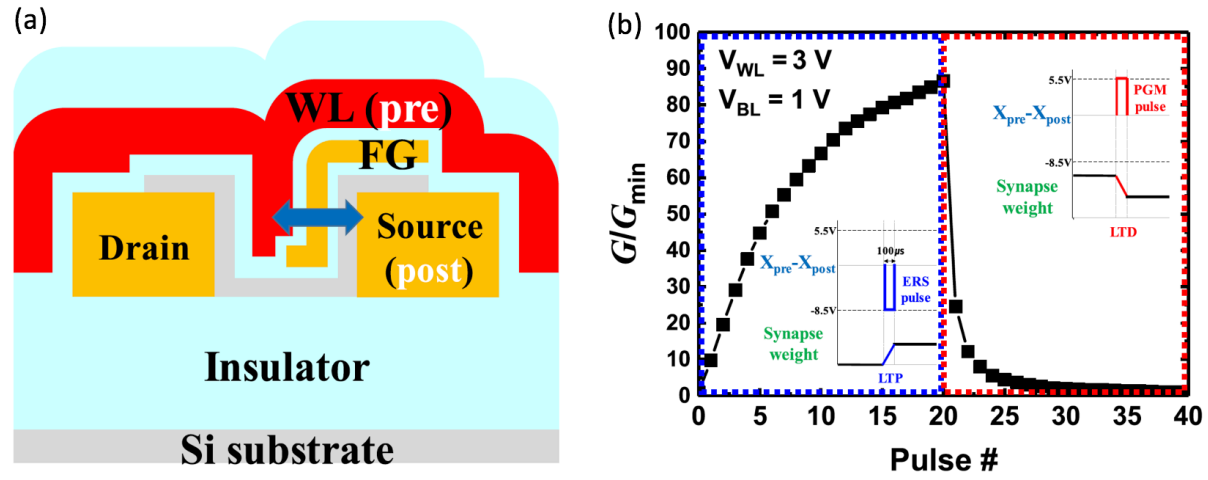

Figure 3.2 Device structure and LTP/LTD characteristics exhibited by a TFT-like NOR flash cell [67]. (a) Schematic of the device structure and the connection of pre- and post-synaptic signals that cause a weight update. (b) LTP/LTD characteristics.

The key feature of a device that enables its use for a WTA network is the weight-dependent plasticity discussed in Section 2.4. We will first use experimentally measured CTT characteristics to study the feasibility of CTT for use in a WTA network (Section 3.1.2), followed by the experimental demonstration in Section 3.3.2.

3.1.2 Simulation results with CTT characteristics

A one-layer WTA network aiming at classifying stylized letters z, v, n, and one-bit-flipped noisy versions of them is studied here using CTTs as the synaptic devices (Fig. 3.3(a)). Prezioso *et al*. in [68] used the same problem for back-propagation training with memristors, but here it is used for unsupervised learning instead. The input layer of the network has nine neurons corresponding to nine pixels of the pattern and the output layer has three neurons corresponding to the three categories: z, v, and n, respectively (Fig. 3.3(b)).

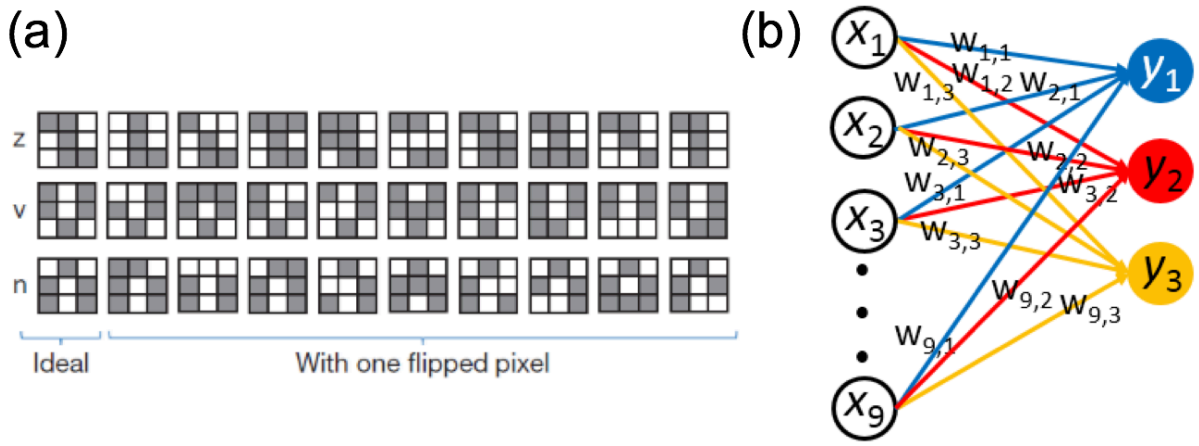

Figure 3.3 (a) The patterns to be clustered: stylized letters z, v, n and noisy versions of them. Reproduced from [68]. (b) The 9×3 WTA network. Output neurons of different colors correspond to different categories. Adapted from [65].

For each output neuron *j* (1, 2, or 3), its output is determined by $y_j = \sum_{i=1}^{9} x_i G_{i,j}$, where $G_{i,j}$ is the channel conductance of the CTT between the input neuron *i* and the output neuron *j*, measured at $V_{GS} = 0$ and $V_{DS} = 50$ mV, and x_i is the input which is 50 mV when the *i* th pixel is black (firing) or 0 when the *i* th pixel is white (not firing). For each presentation of a pattern, the neuron with the largest output fires and claims the pattern, and only the 9 synaptic weights associated with this neuron are updated (hence, winner takes all) and the other synapses remain unchanged. Specifically, only when the output neuron *j* has the largest output and fires (wins) are $G_{i,j}$ (i = 1 − 9) updated. In theory, it has been shown that an optimal weight update follows the weight-dependent plasticity rule [69]:

$$\Delta G \propto 1 - f(G) \text{ if the weight is to be increased, and}$$

$$\Delta G \propto -f(G) \text{ if the weight is to be decreased.}$$

where $f(G)$ is a function of the conductance G. The weight-dependent plasticity behavior in CTT resembles this rule very well, indicating that CTTs might be used as synapses in a WTA network for clustering.

When the output neuron *j* fires, $G_{i,j}$ is increased by detrapping pulses if the input neuron *i* also fires or decreased by trapping pulses if the input neuron *i* does not fire. As shown in Fig. 3.4, in

the simulation, we start from CTTs with random conductances ranging from 50−150 nS. Training of the neural network starts with randomly selecting a pattern from z, v, or n with equal probability and presenting it to the network. Then a random bit of the pattern is flipped and the noisy version is presented to the network again. After each pattern presentation, formulas fitted from experimental data in Section 2.4 is used to update the synaptic weights. The entire process is free of any intervention.

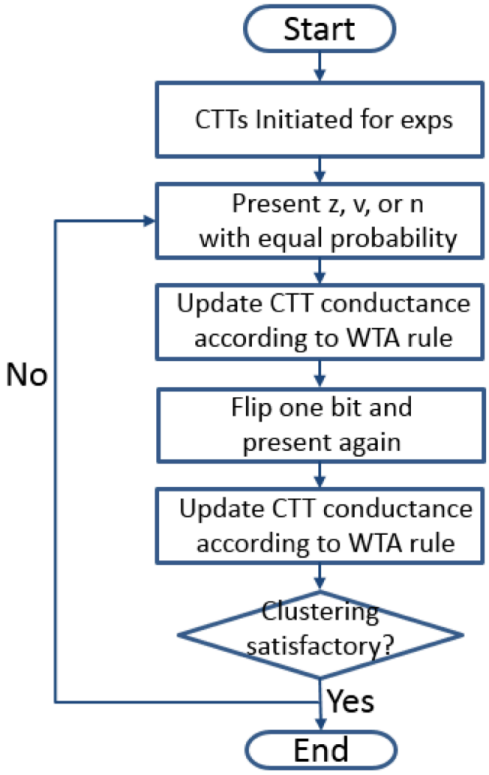

Figure 3.4 Flow chart for WTA network training.

In the simulation, a total of 1000 patterns are presented to the neural network with 500 correct ones and 500 noisy ones. Two trapping (20-μs, 2.5 V gate pulse with V_D = 1.3 V) and detrapping pulses (50-μ s, − 2.6 V gate pulse with zero drain bias) are applied to decrease or increase $G_{i,j}$, respectively. Figs. 3.5(a) and (b) show the clustering results for the first and the last 100 presentations, respectively. It is observed that a substantial number of misclassifications occur in the first 100 cycles, while all patterns are correctly classified for the last 100 cycles. To better

understand the convergence behavior of the algorithm, a specialization function, S_i, is defined for each output neuron i, as the pattern x (z, v, or n) which yields the largest output y_i for the neuron. Perfect clustering is achieved when the neuron specializations remain constant and correspond to three different patterns as the neural network is trained. Fig. 3.5(c) shows the specializations of the output neurons as the network is trained. In fact, perfect clustering is achieved after only 82 training cycles, after which Neurons 1, 2, and 3 correspond to patterns n, v, z, respectively. Between points A and B, even though the specializations of Neurons 2 and 3 stay constant, the algorithm is not convergent since both neurons claim the letter v. It should be further noted that this example is only to illustrate the evolution of specializations and does not represent a typical case. It is verified through 10,000 simulation runs that, the average number of cycles after which perfect clustering is achieved is only 24, well within the demonstrated endurance of over 1,000 for CTT-based non-volatile memory [58].

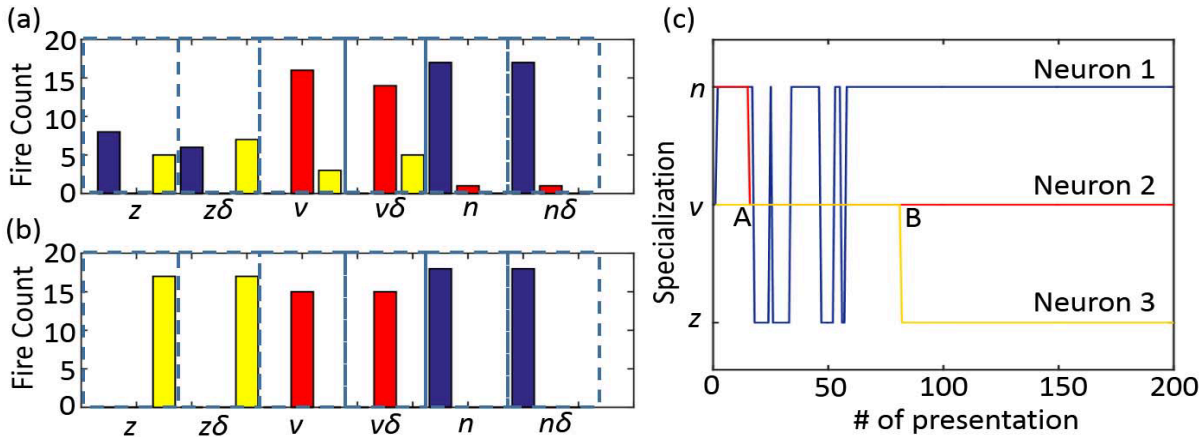

Figure 3.5 Fire counts from three output neurons (a) before and (b) after training. Blue, red, yellow: output neurons 1, 2, and 3. "" denotes a noisy version. (c) The evolution of the output neuron specializations as the network is trained. Adapted from [65].

Fig. 3.6 depicts an example of the evolution of the synaptic weights $G_{1,1}$ and $G_{2,1}$. It is observed that, the sharp decreases in $G_{2,1}$ are larger than the sharp increases in $G_{1,1}$. This is caused by the asymmetry between LTP and LTD found in Fig. 2.19(c). It is also observed that, the weights, starting from random values, eventually reach a steady state after which each weight only varies

around a certain value. In this example, the steady-state is 23.8 nS for $G_{1,1}$ and 93.2 nS for $G_{2,1}$ for the last 100 cycles when two trapping/detrapping pulses are applied in the LTD/LTP regimes. These two values, representing respectively "low" and "high" weights after training, vary with the applied programming conditions. For instance, when five trapping/detrapping pulses are applied, a "low" of 15.2 nS and a "high" of 95.8 nS are obtained. When a longer programming pulse is applied, larger conductance change is induced in each update step, leading to higher "high" and lower "low" eventual weights. Larger weight changes also result in faster convergence and a smaller noise margin. It is anticipated that the amplitudes of the trapping/detrapping pulses will have similar effects on the convergence behavior.

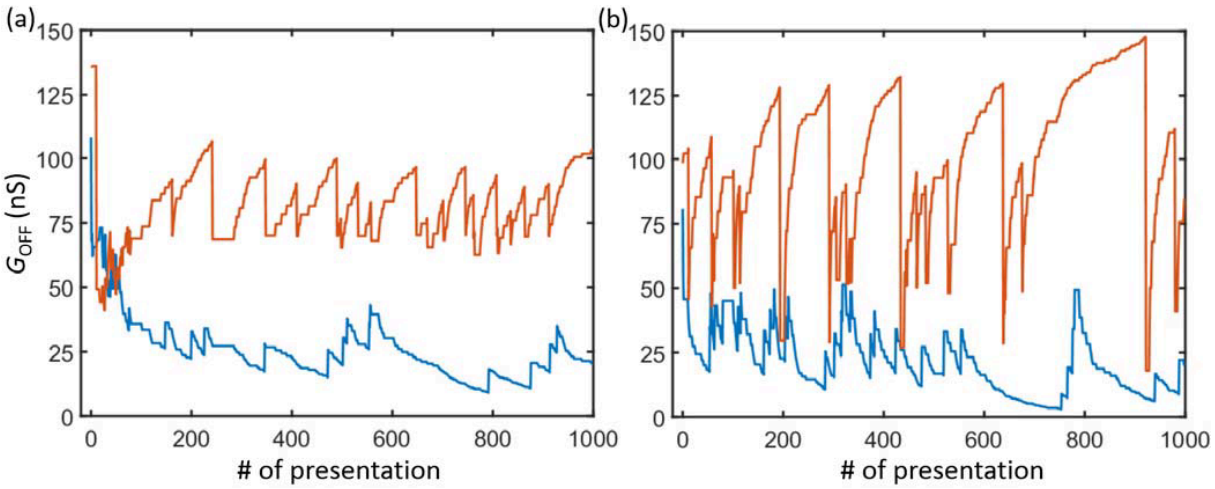

Figure 3.6 An example of the evolution of synaptic weights $G_{1,1}$ (blue) and $G_{2,1}$ (red) for different programming times: (a) Two pulses are applied for LTD/LTP, and (b) Five pulses are applied for LTD/ LTP. Adapted from [65].

In practice, when actual CTTs are used to construct the neural networks, the effect of device variation on the robustness of the algorithm needs to be evaluated. We illustrate here the example where two trapping and detrapping pulses are used to update the weights (Fig. 3.7(a)). An empirically determined variation of Gaussian distribution with 3σ of $f_{10\text{pulse}} - f_{2\text{pulse}}$ is added to the conductance change calculated from fitted equations, where $f_{10\text{pulse}}$ denotes the fitted conductance change when ten pulses are used to update the weights and $f_{2\text{pulse}}$ denotes the fitted conductance

change when two pulses are used to update the weights. More variation is introduced when $G > 60$ nS in the LTP regime and when $G < 40$ nS in the LTD regime to better approximate the experimental data. With this variation, the simulation was performed for 10,000 times and a 100% perfect clustering rate was achieved. Fig. 3.7(b) depicts an example of ΔG as a function of the conductance G itself from one of these simulation runs. It is indeed observed that the conductance change with the empirically introduced variation is comparable to the experimental data. With this methodology, it is also found that a longer programming time leads to a less robust algorithm: perfect clustering cannot be achieved when five LTP/LTD pulses are applied. It means that the effects of the variation are smaller when the programming time is shorter. This is because a shorter programming time corresponds to a smaller ΔG in each update step.

The energy consumption in the LTP regime is minimal since it is only due to electrons being detrapped from the high-k layer. In the LTD regime, the energy dissipation is mainly through the channel current because of the drain bias; it is given by $E = V_D \int I_D \mathrm{d}t$ where I_D is the channel current. For a device with a W/L = 20 nm / 20 nm and programming conditions specified above, E is estimated to be 0.5 nJ. This is a reasonable value compared to the range of pJ to hundreds of nJ reported for many other synapse candidates [44]. This makes the CTT a promising candidate to be used as efficient synapses in low-power online learning networks.

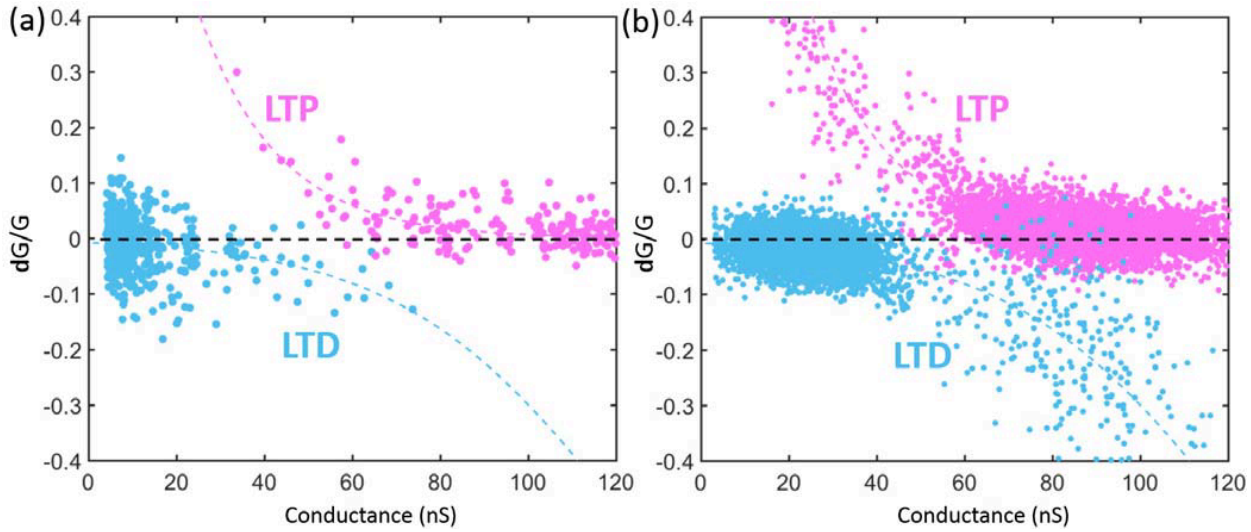

Figure 3.7 (a) Experimentally measured and (b) Empirically determined relative conductance change as a function of the conductance itself in the LTP and LTD regimes. The algorithm converges with the variation shown in Fig. (b). Adapted from [65].

3.1.3 Experimental demonstration

Array-level demonstration is next presented using custom-built standalone arrays. The CTT array has 10 WLs and 8 BLs, with individual and direct access to the terminals of each CTT via two rows of 25×1 scribe line monitor (SLM) pads. The layout of the array together with the SLM pads are shown in Fig. 3.8. During testing, these pads are directly contacted simultaneously by a custom-built probe card. Custom applications are compiled in EasyEXPERT to program and measure the CTT devices.

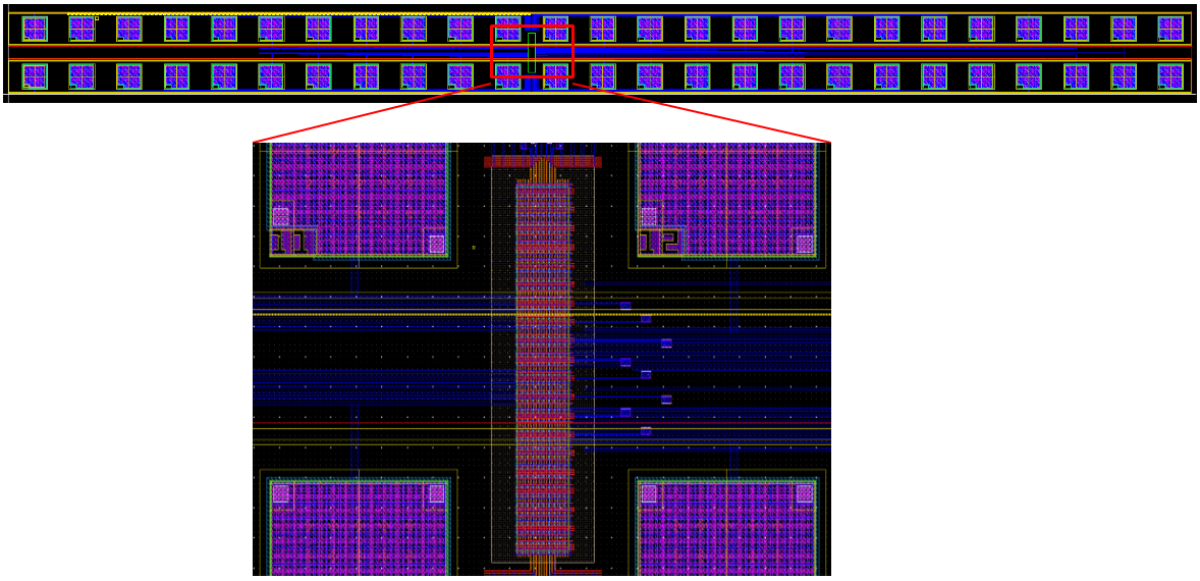

Figure 3.8 The standalone CTT array and two rows of 25×1 scribe line monitor (SLM) pads used to directly access all terminals of each CTT individually.

The clustering of stylized letters z, v, and n is demonstrated here in hardware. Nine rows and three columns are selected to represent the 9×3 CTT array (Fig. 3.9). When a CTT is to be programmed according to the WTA rule, its BL is grounded, and the drain and gate programming pulses are applied to its SL and WL, respectively. All other WLs are grounded to mimic what happens in an integrated chip. On the contrary, when a CTT is to be erased according to the WTA rule, a −1.3 V gate pulse and 1.2 V drain pulse are applied to its WL and SL, respectively, while all other WLs are grounded and SLs floating. A negative gate pulse is necessary to avoid high drain-to-gate voltage on the devices in the same column, which can cause unintended erasure.

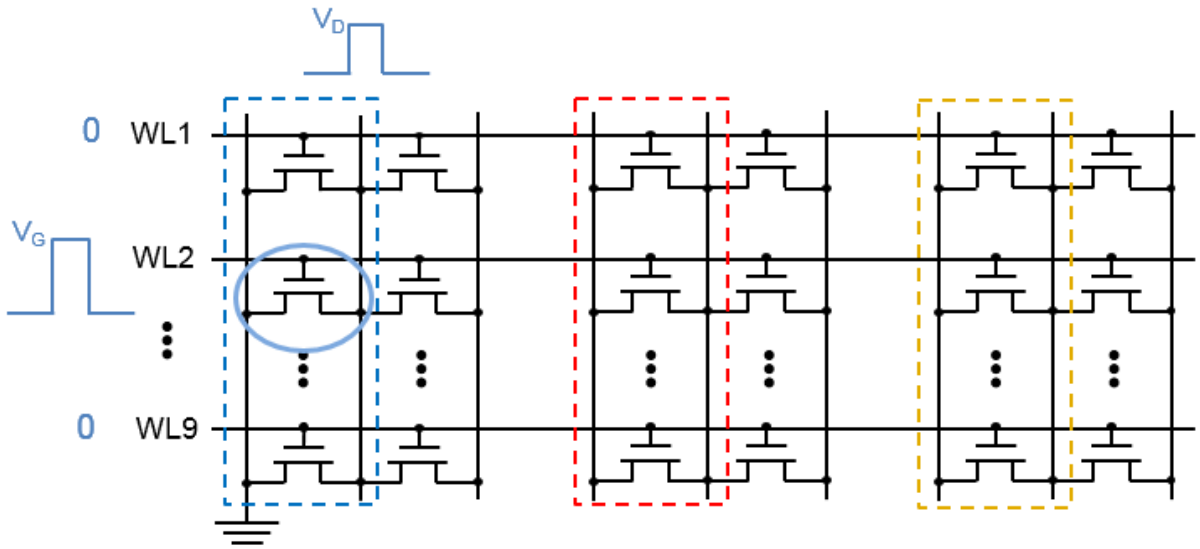

Figure 3.9 Use of the standalone array for demonstration of the WTA network. Three columns are selected corresponding to three neurons of different colors. Also shown is the pulse configuration when a CTT (Row 2, Colum 1) is programmed.

Before the experiments, all CTTs are initialized by a programming pulse to an intermediate state, to allow for subsequent trapping and de-trapping. Using the same array, three trials were performed sequentially. After each trial, the array is erased and initialized again to prepare for the next one. In each run, 40 patterns are presented (the stylized letters z, v, n, and noisy versions of them). The clustering results are examined thereafter and summarized in Fig. 3.10.

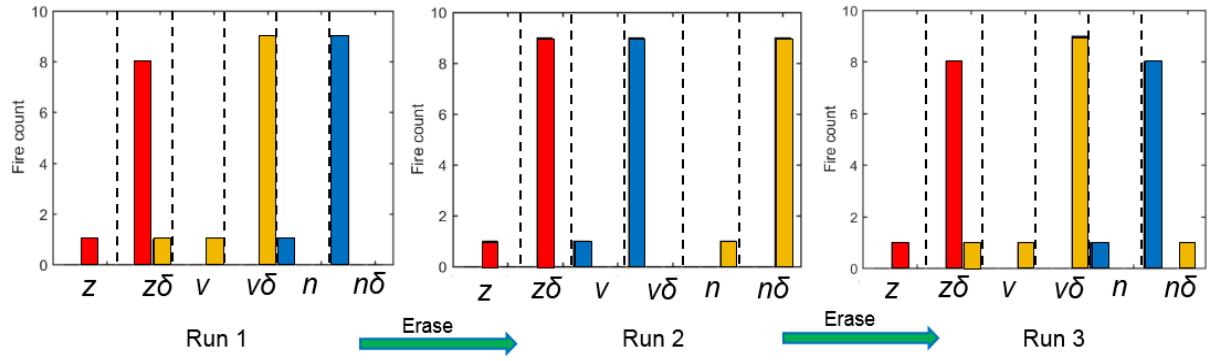

Figure 3.10 Clustering results of the three trial runs.

As shown in Fig. 3.10, the three runs have a clustering error of 1, 0, and 2 counts, respectively, averaging to an accuracy rate of 96.7%. This is lower than the perfect clustering as predicted in

Section 3.1.2, due to the combined effects of inter-device and cycle-to-cycle variations, and also the half selection issue in an array setting. Nevertheless, this demonstration remains, to the best of our knowledge, the largest array-level WTA network that has been implemented using analog memory. For reference, in [66], only four synapses were implemented in hardware, and [67] provided only simulation, using device data from a single thin-film transistor. This work also demonstrates that the half selection issue is not detrimental to the network operation.

We show in Fig. 3.11 the evolution of the 27 synaptic weights during the course of the pattern presentations in Run 2. In Fig. 3.11(a), no apparent patterns exist in the current map before any presentations. After 30 presentations, different patterns emerge in the three neurons, corresponding to the letters v, z, and n, respectively. Note that, in Run 2 of Fig. 3.10, the three neurons also specialize to patterns v, z, and n, respectively. This specialization of neurons to the corresponding patterns is a general feature in WTA networks, as shown in [66] and [67], and simulations in Section 3.1.2.

Shown in Fig. 3.11(b) is an example of the current evolution of two CTTs as circled in green in Fig. 3.11(a). It is important to note that the current does not stay constant even if the CTT is not programmed, an effect of the half selection.

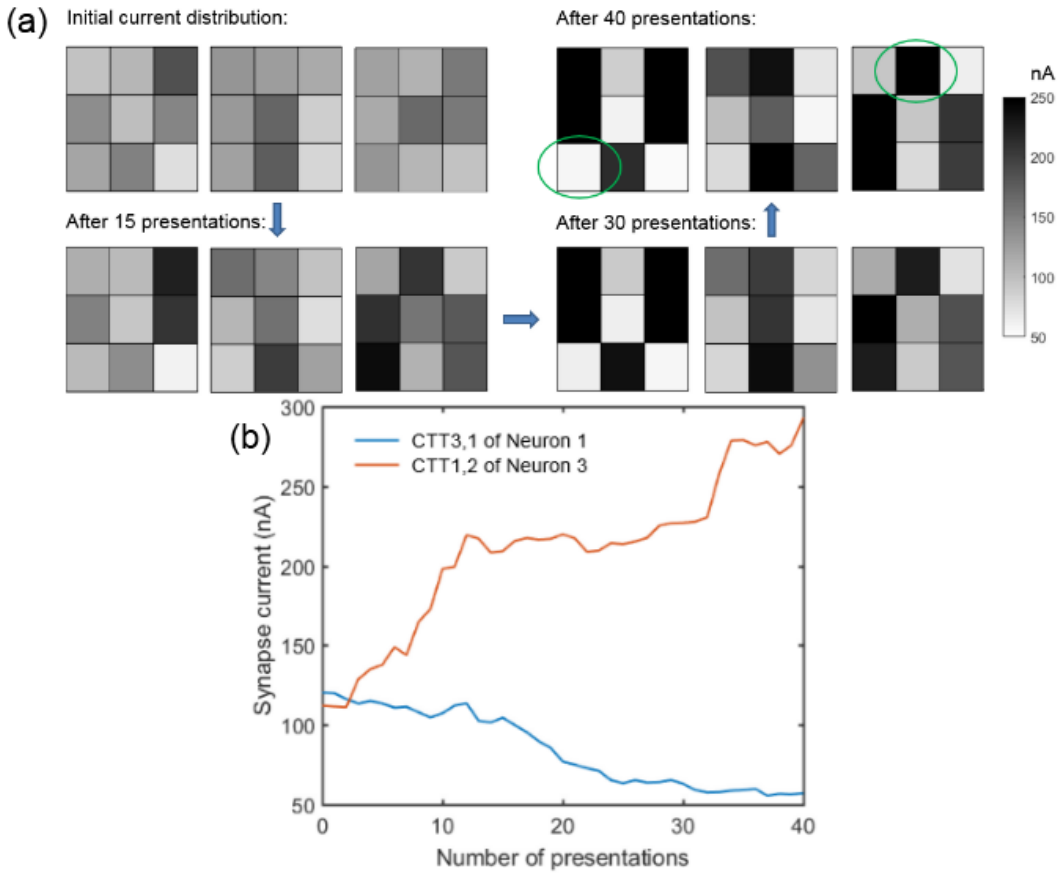

Figure 3.11 (a) The evolution of the synaptic weights. (b) Detailed look at the weight evolution of two CTTs circled by green in Fig. (a).

In this Section, we demonstrated the WTA network using the actual hardware in GlobalFoundries' 22FDX technology. Although perfect clustering was not obtained for all experiments due to inter-device and cycle-to-cycle variations, an average 96.7% clustering accuracy is reasonable. In addition, to the best of our knowledge, this is the first demonstration of a WTA network in an array.

3.2 Temporal correlation detection

3.2.1 Background

The main purpose of temporal correlation detection is to identify, among many stochastic processes, a group of processes that are correlated. In other words, suppose there are N binary processes X_1, X_2, … X_N, each taking the value of 1 at a probability of p and 0 at a probability of $1-p$ at any time instance, the problem is to determine a subset(s) of X_1 to X_N in which all processes are correlated (Fig. 3.12). Statistically, this problem is traditionally solved using k-means clustering or the covariance matrix. We shall first briefly review the two methods to reveal the complexity involved, and then formulate how CTT device dynamics can be exploited to solve the problem.

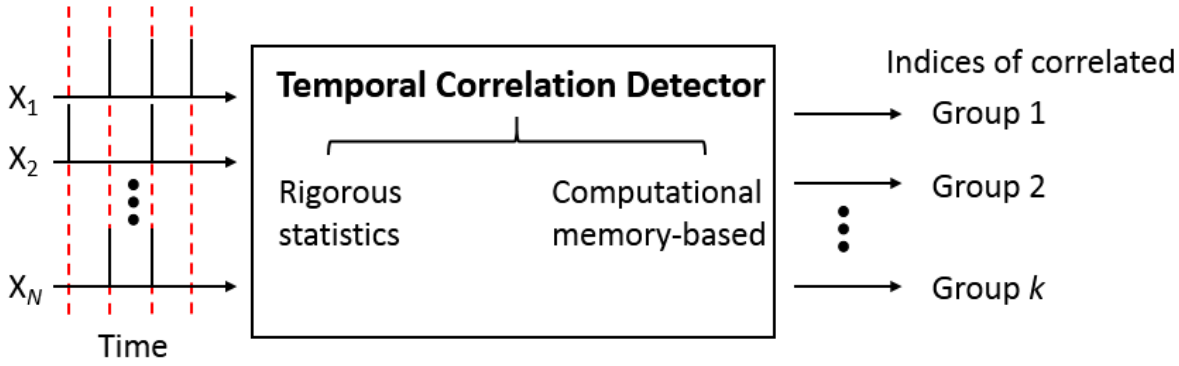

Figure 3.12 Schematic illustration of the temporal correlation detection problem.

K-means clustering focuses on finding k partitions $\{S_1, S_2, \ldots S_k\}$ of the processes and the corresponding centroids $C_1, C_2, \ldots C_k$, such that

$$\sum_{i=1}^{k} \sum_{X \in S_i} ||X - C_i||^2 \tag{3.1}$$

is the smallest among all possible partitions and centroids. Here $||\bullet||$ denotes the Euclidean distance. This problem is very difficult to solve, and simplified algorithms are often used instead as approximate solutions. The following is a step-by-step breakdown of one such example (Fig. 3.13).

(1) Start from a random set of centroids, $C_1^{(0)}$, $C_2^{(0)}$, … $C_k^{(0)}$.

(2) For each X_i (i = 1, 2, … N), calculate the Euclidean distance between X_i and C_j (j = 1, 2, … k), $D_{ij} = ||X_i - C_j||$. Assign X_i to Cluster j which yields the smallest D_{ij}.

(3) For each cluster S_i, update its centroid $C_i^{(M)}$ to the centroid of all processes assigned to S_i in Step (2).

(4) Repeat from Step (2).

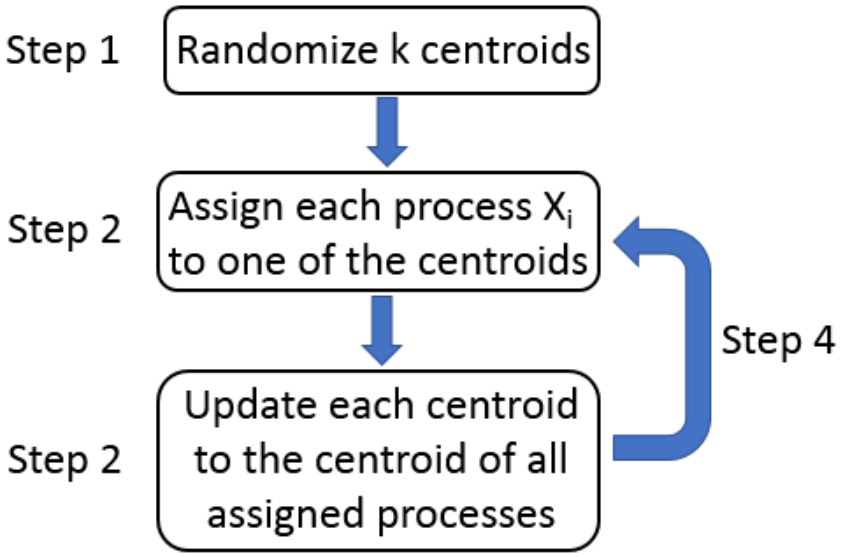

Figure 3.13 Flow chart of a simplified *k*-means clustering algorithm. The process is usually repeated multiple times to find an optimal solution.

Due to the random initialization of centroids, the same process is usually carried out multiple times to find the best solution. It is clear that *k*-means clustering requires data storage at all time instances and a large number of calculations of Euclidean distance between high-dimension vectors. This method is very time-consuming and requires a lot of memory.

Another method to detect temporal correlation is to use the covariance matrix [70]. It requires the calculation of

$$R_{ij} = \frac{1}{K}\sum_{k=1}^{K} X_i(k)X_j(k) \tag{3.2}$$

and the subsequent calculation of

$$W_i = \sum_{j=1}^{N} R_{ij} \tag{3.3}$$

In the limit of large K, the expectation of W_i can be estimated. It can be shown that, if $E(W_i) = (N - 1)p^2 + p$, then X_i belongs to the uncorrelated group; on the contrary, if $E(W_i) > (N - 1)p^2 + p$, then X_i belongs to a correlated group. Still, a significant amount of computation is necessary.

In [70], Sebastian *et al*. utilized the crystallization dynamics of PCM to significantly simplify this problem. At each time instance k, a current pulse is applied to the PCM whose corresponding process takes a value of 1. The amplitude or the duration of the current pulse is proportional to $M(k) = \sum_{i=1}^{N} X_i(k)$, which can be obtained with little computation effort. Using this approach, the authors demonstrated the satisfactory separation of PCM conductance for correlated and uncorrelated processes. The same method was also employed to detect the correlation of real-world rainfall data collected from 270 weather stations across the United States. Their experiments show that, out of the 270 stations, 245 are classified in the same way by the PCM-based approach and the k-clustering algorithm. In comparison, 251 are classified in the same way by the covariance approach and the k-clustering algorithm, a mere 2.4% improvement over 245. Considering that there is significant variation in PCM (Fig. 3.14), and that the procedure is very fast and requires little computation effort, their results are very promising.

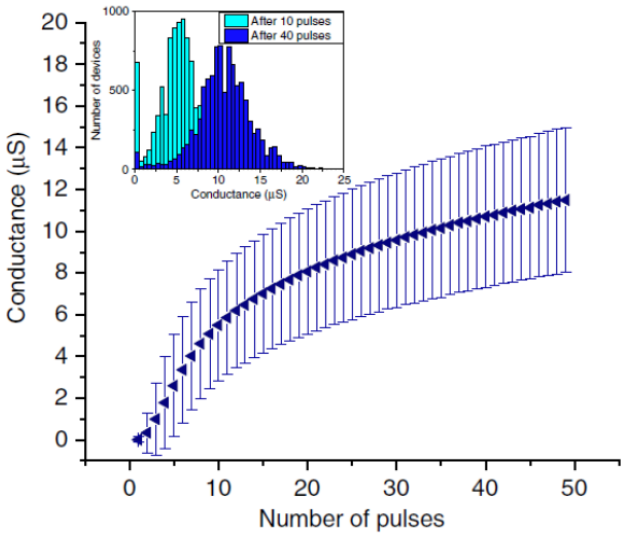

Figure 3.14 Mean and standard deviation of the PCM used in [70] when the SET current is 100 μA.

3.2.2 Simulation results with CTT characteristics

The key characteristic of PCM that enables its use as an in-memory computation unit is its amplitude- or pulse-number-dependent programming (Fig. 3.15): the higher the programming current, and/or the longer the programming pulses are applied, the larger the conductance is.

CTTs have similar programming dynamics: the conductance change is larger when more pulses are applied, or the voltage amplitude is larger. We demonstrate in this Section that this dynamics can indeed be exploited for temporal correlation detection. Shown in Fig. 3.16 is a typical relationship between the CTT conductance and the number of programming pulses when 20-μs, V_G = 2.6 V and V_D = 1.2 V pulses are applied. Also shown is the empirically fitted curve

$$I = C \times 10^{\alpha\left(\exp\left(-(N/t)^{\beta}\right)-1\right)} \tag{3.4}$$

where C is a constant, α reflects the largest V_T shift, t reflects the steepness of the curve, and β reflects the trap distribution. Here, C = 241.3 nA, α = 1.5, t = 170.9, and β = 0.5.

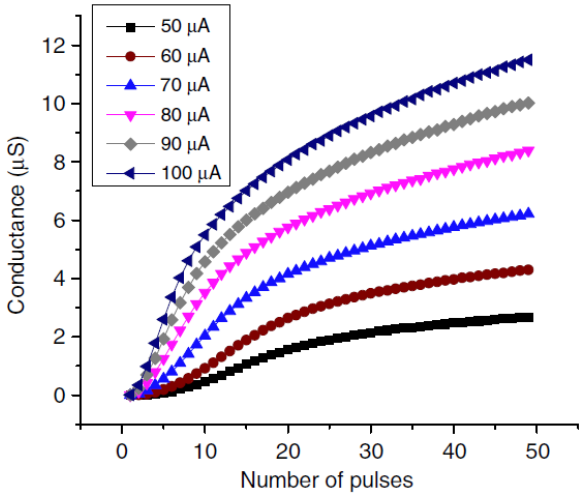

Figure 3.15 Amplitude- and pulse-number-dependent programming behavior exhibited by PCM used in [70]. Each point on the curves is an average from 10,000 devices.

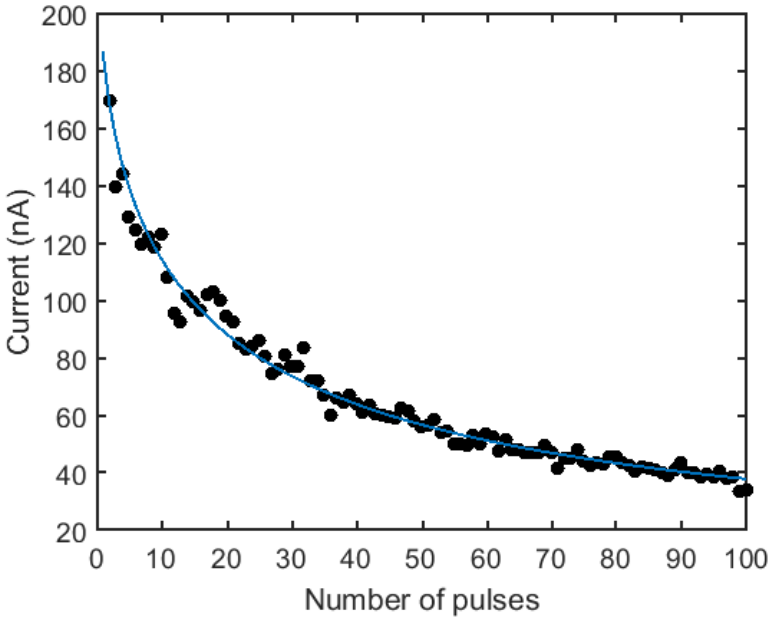

Figure 3.16 CTT conductance as a function of the number of programming pulses. Black dots: experiment; blue curve: empirically fitted curve. Here, the pulse is fixed at 20 µs, V_G = 2.6 V, and V_D = 1.2 V.

Taking advantage of this characteristic of CTT, we next investigate the feasibility of CTT-based temporal correlation detector. An approach similar to that in [70] is adopted with one

distinction: in their study, the amplitude or the duration of the current pulse is proportional to $M(k) = \sum_{i=1}^{N} X_i(k)$, while here, a fixed pulse (both in voltage amplitude and time duration) is applied to the CTTs if $M(k)$ exceeds a certain threshold, and no pulses are applied otherwise.

For the purpose of demonstration, a 600×800 black-and-white image shown in Fig. 3.17 is used. The 480,000 pixels correspond to 480,000 stochastic processes: the processes corresponding to the black pixels are correlated while others are uncorrelated. In this example, 143,402, or 29.9% processes are correlated. At each time instance, the uncorrelated processes have a probability of p to assume the value of 1 and a probability of $1 - p$ to assume the value of 0. For the correlated processes, a reference process is first selected. If the reference process assumes the value of 1 at any time instance, then other correlated processes have a probability of $p + \sqrt{c}(1 - p)$ to assume the value of 1; otherwise, other correlated processes have a probability of $p(1 - \sqrt{c})$ to assume the value of 1. Here, c is the correlation coefficient of the correlated processes. In the following results, p is equal to 0.1 unless otherwise stated.

Figure 3.17 A black-and-white image used for the demonstration of temporal correlation detection with CTT. Black pixels correspond to correlated processes while white pixels correspond to uncorrelated ones.

480,000 CTTs are randomly initialized with their inference currents uniformly distributed between 100 and 300 nA. At each time instance, if and only if $M(k) = \sum_{i=1}^{N} X_i(k) > \eta = 6\times10^4$ is a 20-μs pulse with V_G = 2.6 V and V_D = 1.2 V applied to the CTTs corresponding to a firing process and the inference current is updated according to Eq. 3.4. Fig. 3.18(a) shows an example of the inference current distribution for the correlated and uncorrelated processes after 1,000 time instances when c = 0.05. It is clear that the inference currents of CTTs for correlated and uncorrelated processes are separated, though not completely, even for a correlation coefficient as small as 0.05. To build a classifier, one can select a threshold inference current and label any processes with a current less than the threshold "correlated". For example, Fig. 3.18(b) shows the reconstructed image when a threshold of 80.5 nA is selected. The fidelity is high with some black dots in the white area and some white dots in the black area.

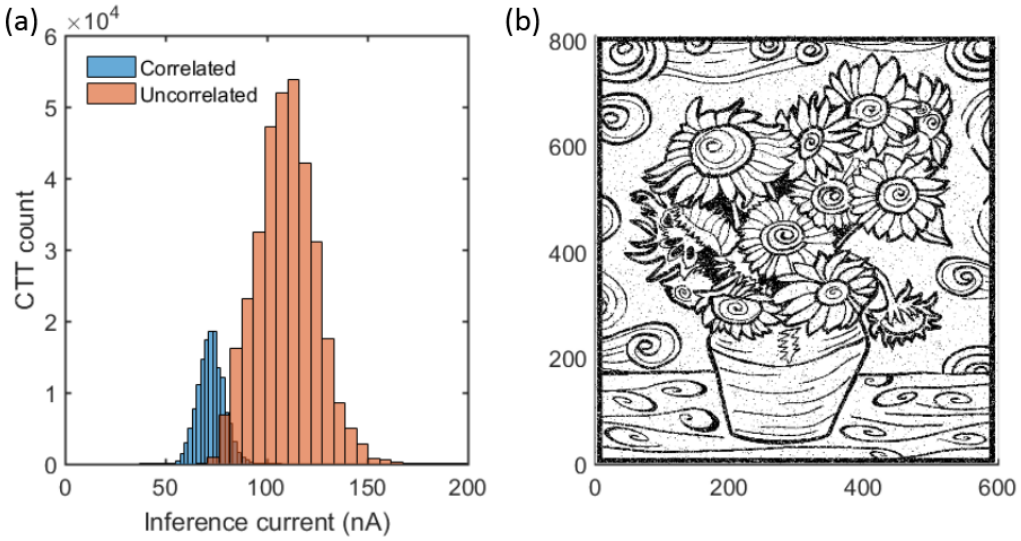

Figure 3.18 Temporal correlation detection achieved by CTT after 1,000 time instances when c = 0.05. (a) The separation of CTT inference currents between correlated and uncorrelated processes. (b) The reconstructed image when a threshold inference current of 80.5 nA is used by the classifier.

A quantitative way to evaluate the performance of a temporal correlation detector is the so-called precision-recall curve. For each threshold inference current selected, two quantities can be

computed: (1) recall, which is the ration between the number of correctly labeled processes to the total number of correlated processes, and (2) precision, which is the ratio between the number of correctly labeled processes to the number of all processes labeled as “correlated”. An example of the precision-recall curve for the detector in Fig. 3.18(a) is shown in Fig. 3.19. The reconstructed image in Fig. 3.18(b) uses a threshold inference current of 80.5 nA, which corresponds to a recall of 0.9 and a precision of 96.24 % in Fig. 3.19.

It is natural to anticipate the degradation of the detector performance as the correlation coefficient decreases. Fig. 3.20(a) clearly shows this trend. As shown in Fig. 3.20(b) for $c = 0.02$, it is worth noting the increased overlap between currents for correlated and uncorrelated processes compared to Fig. 3.18(a) where $c = 0.05$. Also note the much lower quality of the reconstructed image when the recall is 0.9. In this case, the precision is only 77.39 %. When $c = 0.01$, if a 100% recall is required, the precision drops to 29.9 %, identical to that of a random classifier.

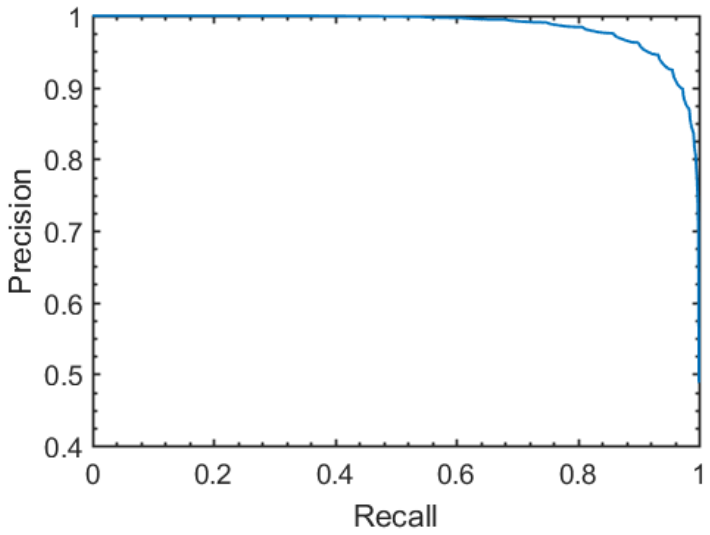

Figure 3.19 Precision-recall curve of the detector in Fig. 3.18(a).

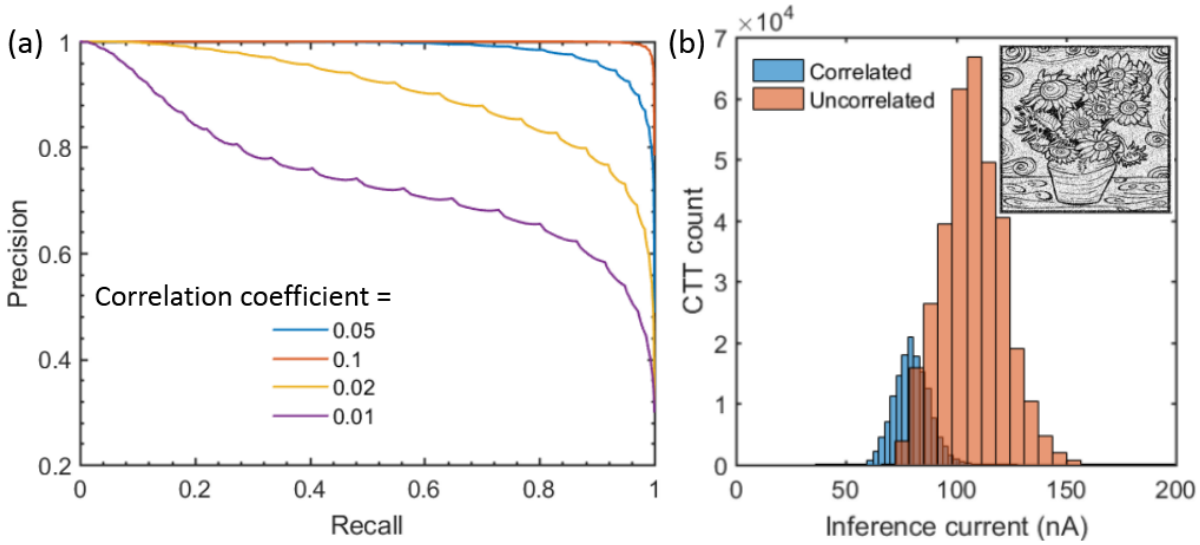

Figure 3.20 (a) Precision-recall curves for different correlation coefficients: 0.1 (red), 0.05 (blue), 0.02 (yellow), and 0.01 (purple). (b) The histogram of inference currents for correlated and uncorrelated processes when $c = 0.02$. The inset shows the reconstructed image at a recall of 0.9.

The area under the curve (AUC) is a good measure of the detector performance. Fig. 3.21 shows the general relationship between AUC and the correlation coefficient. It is seen that, even at a correlation coefficient of only 0.01, the correlation detector classifies about 2.4 times better than a random classifier. At a correlation coefficient of 0.1, the classifier is close to a perfect one with an AUC of 0.9994.

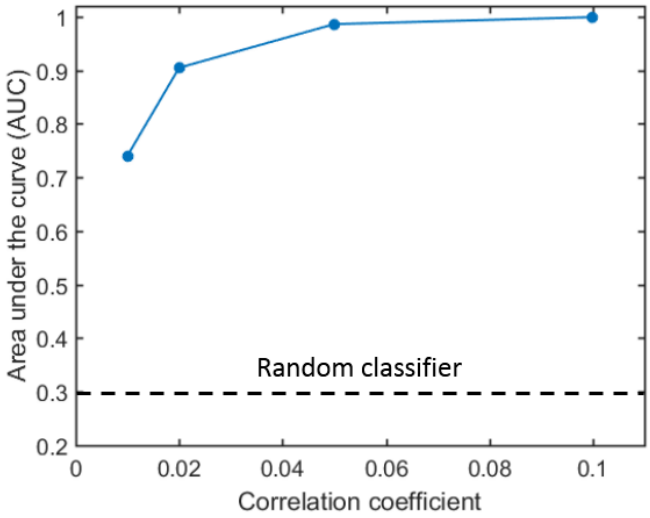

Figure 3.21 Area under the curve (AUC) of the precision-recall curves in Fig. 3.20(a) as a function of the correlation coefficient. The dashed line is the performance of a random classifier.

It is insightful to observe the evolution of the reconstructed image as time goes by. Fig. 3.22 shows the reconstructed images after different number of time instances. Continued improvement is clearly seen as more data becomes available in a longer period of time.

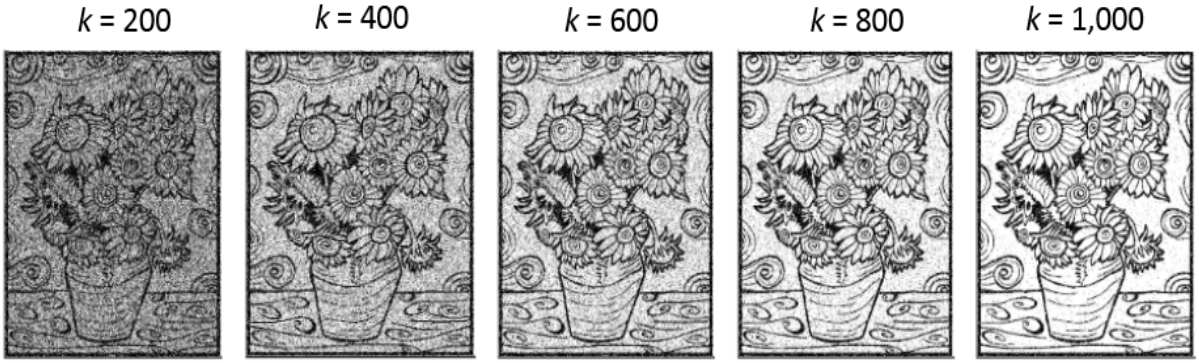

Figure 3.22 The evolution of the reconstructed image (at recall = 0.9) at different time instances k = 200, 400, 600, 800, and 1,000. Improving fidelity can be observed.

Fig. 3.23(a) illustrates the evolution of the precision-recall curve, where improving precision can be observed as more time instances are presented to the detector. Fig. 3.23(b) shows how the AUC changes as time goes by.

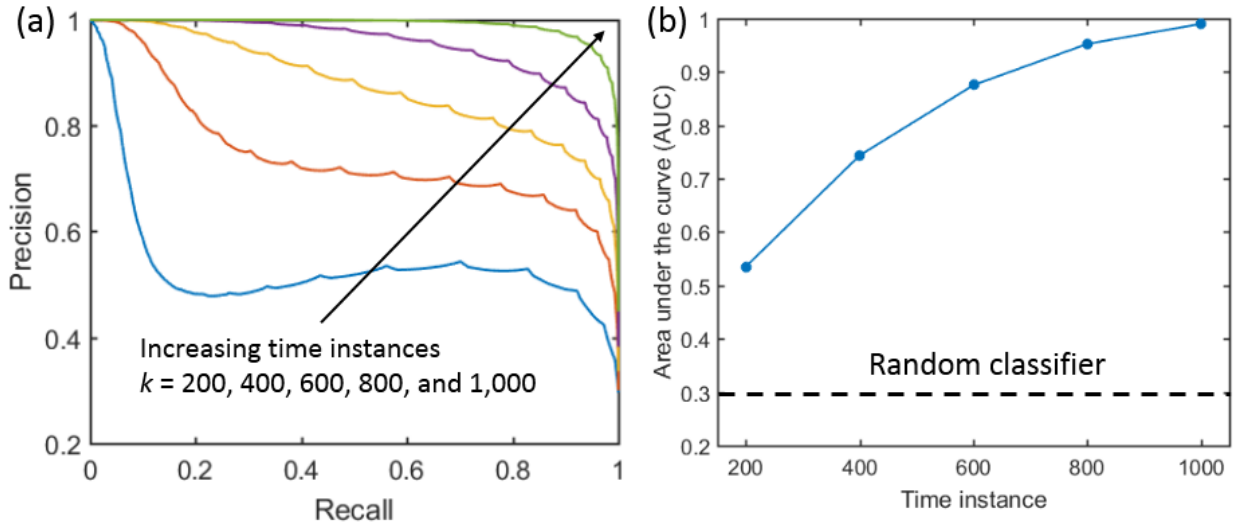

Figure 3.23 (a) The precision-recall curves at different time instances $k = 200$, 400, 600, 800, and 1,000. (b) AUC of the curves in Fig. (a). The dashed line is the performance of a random classifier. Here, $p = 0.1$ and $c = 0.05$.

We have shown that CTTs can be used to detect, among many stochastic processes, a group of correlated ones. We next show the possibility of using CTTs to identify more than one group of correlated processes. Here, 500,000 stochastic processes are used. Among them, there are 400,000 uncorrelated processes and two correlated groups each having 50,000 processes. The two correlated groups have a correlation coefficient of 0.8 and 0.4, respectively. Fig. 3.24 shows the histograms of CTT inference currents corresponding to the three groups. It is clear that the currents are well separated.

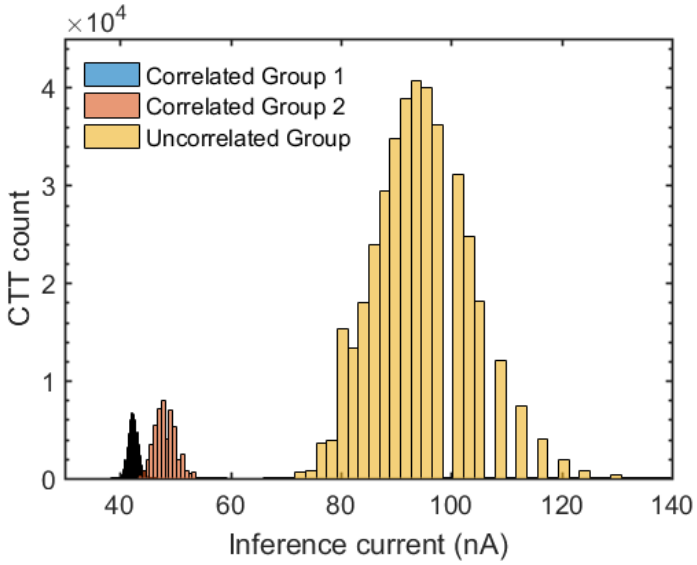

Figure 3.24 An example of the detection of two correlated groups and an uncorrelated one. The correlation coefficient for the two correlated groups are 0.8 and 0.4, respectively. It is clear that the inference currents corresponding to the CTTs belonging to the three groups are well separated.

Up to this point, it has been demonstrated that the CTT device dynamics can be utilized for temporal correlation detection. In practice, however, devices are not ideal, and variations can play a significant role in the detector performance. There is more than one source of variations. First, there is process variation resulting in different device initial states. For example, the standard deviation of V_T is approximately 25 mV, translating to about 2× difference in the inference current. Second, different devices may program differently. Third, there is intra-device cycle-to-cycle variation as can be observed in Fig. 3.16. This is due to the stochastic nature of the charge-trapping process. The last two effects are much more difficult to evaluate through simulation. Here, we focus on the effect of initial CTT states and will show experimental results in the next Section. Fig. 3.25 shows the statistical precision-recall curves for different correlation coefficients c = 0.1, 0.05, 0.02, 0.01. Each curve represents results from 1,000 runs, each starting from random CTT inference currents uniformly distributed between 100 nA and 300 nA. Each run also uses different stochastic processes. It is seen that the four curves are well separated, indicating that the effects of

initial weight distribution are not detrimental, and that one can infer the correlation coefficient from the shape of the precision-recall curve.

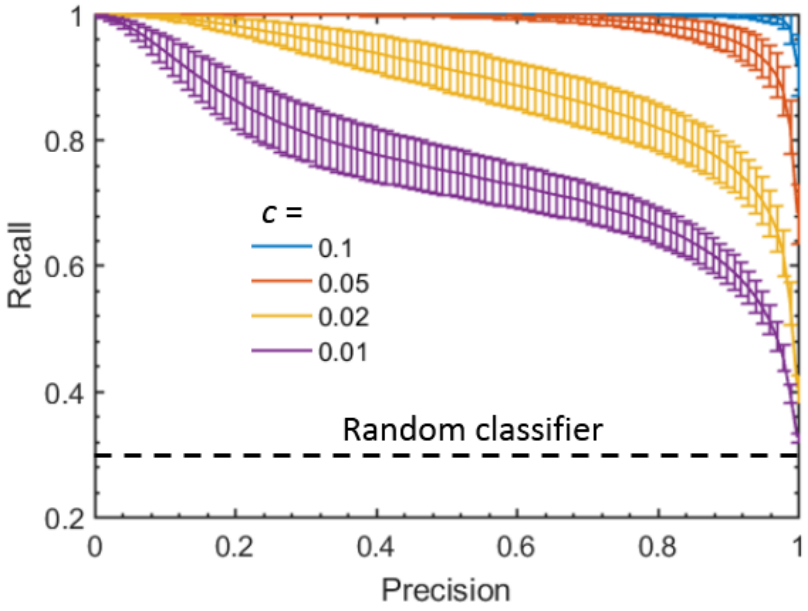

Figure 3.25 Statistics of the precision-recall curves for different correlation coefficients c = 0.1, 0.05, 0.02, and 0.01. Solid lines are average for 1,000 runs and the error bars indicate the standard deviation.

3.2.3 Experimental demonstration

Half of the standalone array is used here for the demonstration of temporal correlation detection. The 80 devices correspond to 80 random processes, 17 of which are correlated, as indicated by the black pixels in Fig. 3.26(b).

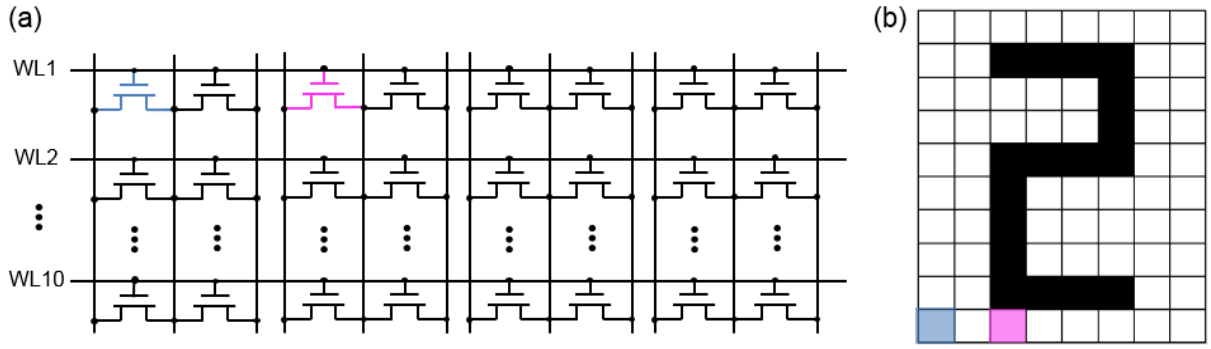

Figure 3.26 The configuration for the demonstration of temporal correlation detection using the standalone array. Each CTT corresponds to a binary random process. The correlated processes are indicated by black pixels in Fig. (b).

At any time instance, the probability of firing by any uncorrelated random process is p. For the correlated processes, the reference process also has a probability of firing equal to p. For the non-reference process, the probability of firing is $p + \sqrt{c}(1 - p)$ when the reference process fires, and $1 - p - \sqrt{c}(1 - p)$ when the reference process does not. Different correlation coefficients c are used and p = 0.1 unless otherwise stated. The firing threshold, η, is chosen to be 13.89 from simulations using measured CTT characteristic.

Because there are significantly fewer CTTs in this demonstration, the effect of inter-device and cycle-to-cycle variations are expected to be more severe since there are not enough devices or cycles for the variations to average out. Therefore, the ratio of the post-programming CTT current to the pre-programming one, $R = I_{post}/I_{pre}$, instead of the current alone, is used to distinguish CTTs corresponding to the correlated processes from those corresponding to uncorrelated processes. Shown in Fig. 3.27 (a) is the distribution of this ratio for the correlated and uncorrelated processes for c = 0.8, after 400 time instances. It is clear that the two groups are well separated. A threshold of 0.3 is chosen here for R: if $R < 0.3$, the corresponding process is considered one from the correlated group; otherwise, the corresponding process is considered one from the uncorrelated group. The perfectly reconstructed pattern is shown in Fig. 3.27(b).

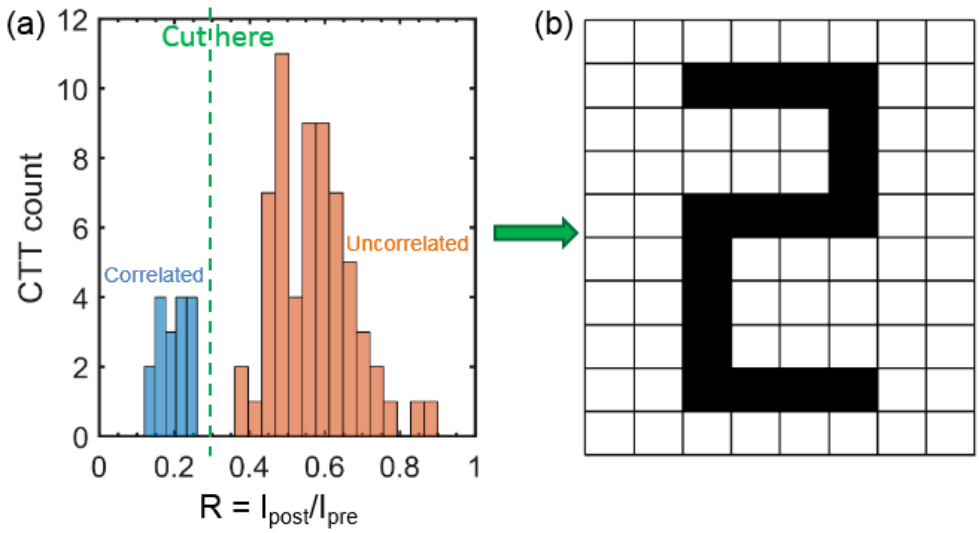

Figure 3.27 (a) The distribution of current ratio after 400 time instances when c = 0.8. (b) The reconstructed pattern using R = 0.3 as the threshold.

In this experiment, at 51 of the 400 time instances was the firing threshold exceeded. After each time it is exceeded and the devices are programmed, the currents of all CTTs were measured. Fig. 3.28 shows the evolution of R for the correlated and uncorrelated groups, where it can be seen that the CTTs corresponding to the correlated processes are almost always programmed when the firing threshold is reached.

Two more experiments were performed using the same array (after erasing it) but with smaller correlation coefficients: 0.5 and 0.2, respectively. The precision-recall curves after the three experiments are shown in Fig. 3.29, where a degradation can be observed as the correlation coefficient decreases.

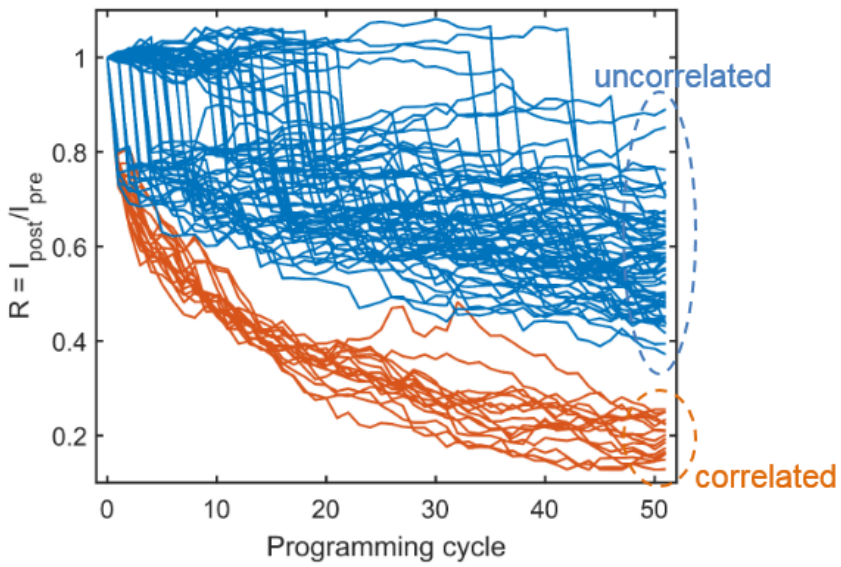

Figure 3.28 The evolution of the current ratio $R = I_{post}/I_{pre}$ for the correlated and uncorrelated groups.

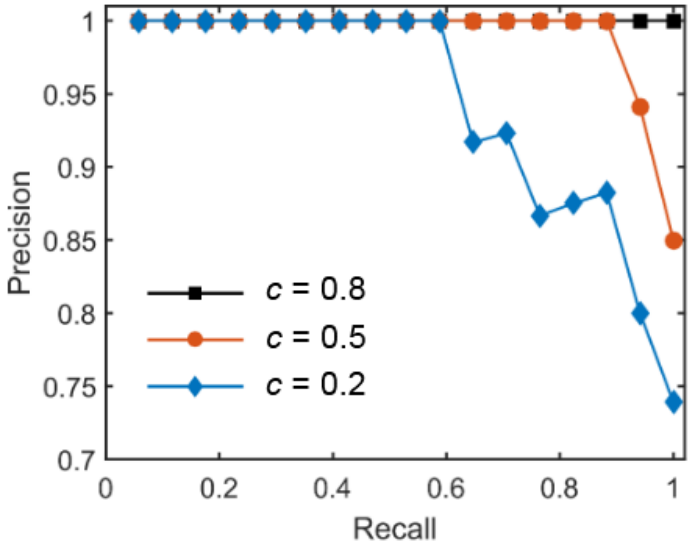

Figure 3.29 The precision-recall curves after 400 time instances, for c = 0.8, 0.5, and 0.2.

In this Section, the feasibility of temporal correlation detection is experimentally demonstrated using a CTT array. Although the scale of the experimental array is small, a much larger network was simulated. Moreover, as suggested in [70], more than one CTTs can be used to represent one random process, reducing the effects of variations.

4. CTTs in An Inference Engine

4.1 Implementation of fully connected neural networks using CTTs

Most inference engines, including fully connected perceptrons and much more complex convolutional neural networks, require two primitive operations: the calculation of the summation of weighted inputs, and the activation of the result. The first primitive operation computes the following:

$$y_j = \sum_{i=1}^{N} w_{ji} x_i,$$

where x_i is the input from ith input neuron, y_j is the internal state of the jth output neuron, and w_{ji} is the synaptic weight between the two (Fig. 4.1).

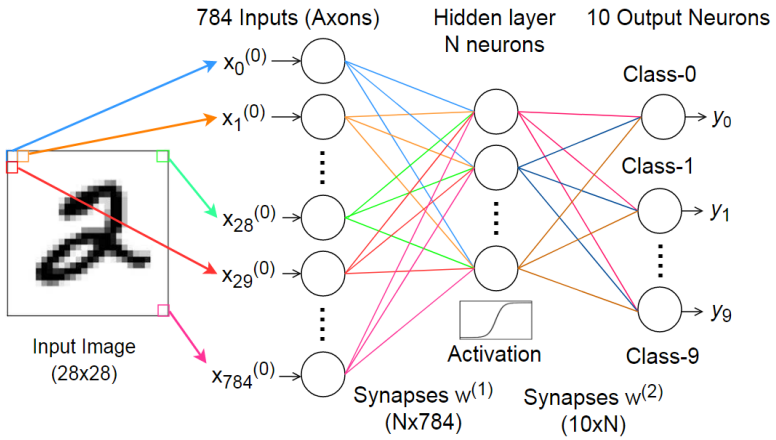

Figure 4.1 A fully connected neural network.

For gray-scale instead of binary inputs, there are two options to implement the neural network in hardware using CTTs as the synapses. As shown in Fig. 4.2, in the first approach, a constant gate voltage is applied a to all CTTs, which biases the devices in the linear region. A small drain voltage proportional to the intensity (x) of the input signal is applied to the drain of CTTs such that the drain current of the CTT is $x \bullet G_{ch}$, where G_{ch} is the linear channel conductance at the given gate

bias. If the CTTs are programmed in such a way that its G_{ch} is proportional to the target synaptic weight, then its drain current naturally represents the weighted input. Furthermore, all CTTs corresponding to the same output neuron are arranged in the same column, making the total current seen by the neuron the weighted summation of inputs. This current can then be sensed by an analog-to-digital converter (ADC) and further processed in the digital domain using any activation function. The activated values can then be converted to the analog domain again as the voltage amplitude using the digital-to-analog converter (DAC), to be applied to the next layer.

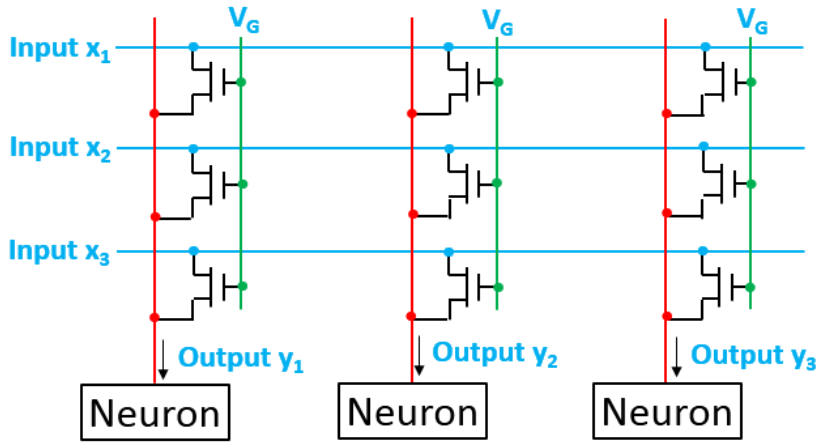

Figure 4.2 Illustration of the hardware implementation of a fully connected neural network using the amplitude of drain voltage as inputs.

However, this approach has a few drawbacks. First, the current is higher in the linear region than in the subthreshold region, resulting in higher energy consumption. Second, to ensure a linear operation of the CTT during inference, the highest drain voltage should be as small as possible, posing additional requirements on the ADC design. Third, because the architecture involves ADC and DAC, it is anticipated to consume a lot more power than an analog-only approach.

Another approach, as shown in Fig. 4.3, encodes the pixel intensities into the width of the pulse applied to the gate of the CTTs − the larger the input is, the longer the pulse is applied. Because the input information is carried by the pulse width, only a single voltage amplitude is necessary.

The current flowing through the CTTs in a column is integrated by the neuron, leading to a voltage on the capacitor of the integrator being proportional to the summation of weighted inputs:

$$V = \frac{1}{C}\sum I_i t_i$$

Here, C is the capacitance, I_i is the current of the CTT at the predetermined gate and drain bias, and t_i is the width of the gate pulse.

To implement the activation function in the analog domain, the rectifying linear unit (ReLU) is straightforward (Fig. 4.4(a)): a comparator can be used to discharge the capacitor until the voltage drops below a certain level V_{ref}, after which the discharging will stop (Fig. 4.4(b)).

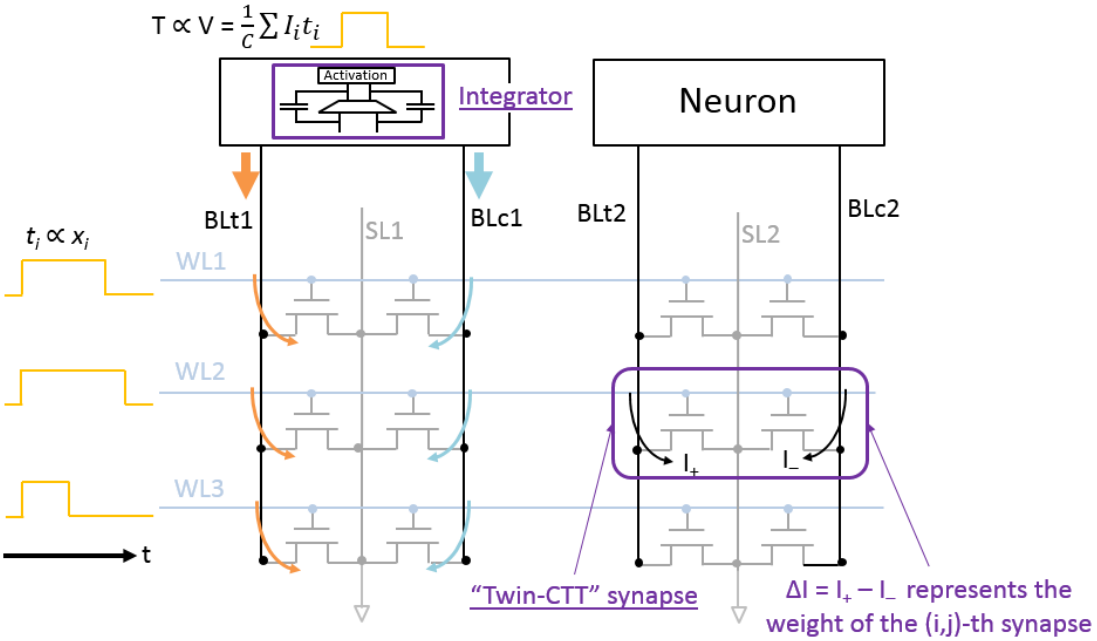

Figure 4.3 Illustration of the hardware implementation of a fully connected neural network using the pulse width as inputs.

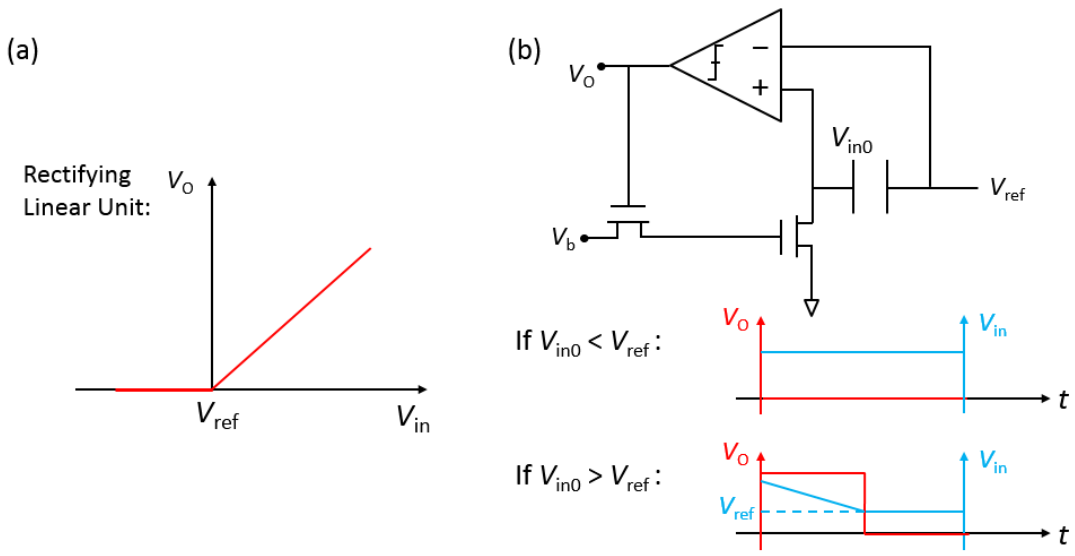

Figure 4.4 (a) The rectifying linear unit (ReLU) as the activation function. (b) Schematic illustration of the simple implementation of ReLU.

Another important point is that, a twin-CTT synapse cell is necessary – the current difference between two CTTs, $I = I^+ - I^-$, instead of the current of a single CTT, is used to represent the synaptic weight. This is primarily due to three reasons: 1) It allows the implementation of bipolar weights. If only unipolar weights are used, extra conversion is required in the digital domain, compromising the potential benefits of analog computation. 2) It reduces the effect of variation. 3) It reduces the effect of V_T/current recovery after programming since the behavior will be similar between the twin CTTs.

4.2 Fine-tuning of CTT weights

Before a CTT array can be deployed as a dot product engine to be used in an inference engine, the synaptic weights of the twin-CTT cells need to be programmed to satisfactory accuracy first. Fig. 4.5 shows the initial weights (at V_{GS} = 200 mV and V_{DS} = 50 mV bias) of a 10 WL × 4 BL twin-CTT array as fabricated by the foundry. The weights range from approximately −800 nA to 600 nA. Forty target weights ranging from −200 nA to 200 nA are randomly generated, and then the CTTs are programmed according to their corresponding weights. In each programming-

verification cycle of the entire array, all 40 twin cells are programmed. In each twin-cell, only one of the CTTs is programmed. If the measured current difference between the twin CTTs is smaller than the target weight, the "negative" CTT is programmed; if the measured current difference between the twin CTTs is larger than the target weight, the "positive" CTT is programmed.

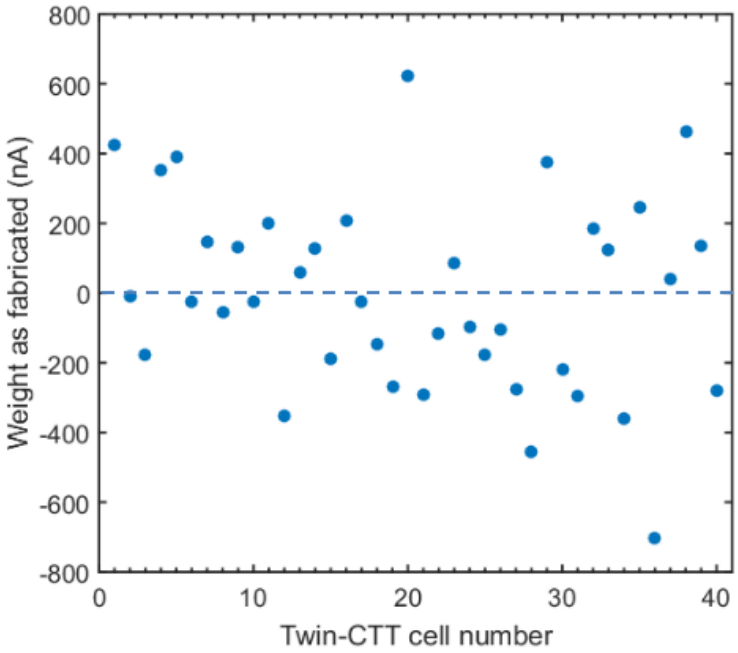

Figure 4.5 The initial weights (at V_{GS} = 200 mV and V_{DS} = 50 mV bias) of an as-fabricated twin-CTT array.

To alleviate the half selection problem, both the gate and the drain voltages need to be as small as possible, while still being able to program the CTTs. In the experiments, gate voltage and drain voltage start from 1.8 V and 1 V, respectively. When the programming is no longer efficient, the voltage is increased by 50 mV, but not to exceed 2.2 V for the gate voltage and 1.2 V for the drain voltage. Both voltages are increased by 50 mV when the programming becomes inefficient. The width of the programming pulses is at a constant 500 µs.

Fig. 4.6 shows the programmed twin-cell current as a function of the target current, right after the fine-tuning and 14 hours later. The goal is to have all the dots on the ideal $y = x$ line. However,

because of the half selection and thermal disturbance from adjacent cells, as well as the limit of programming resolution (for example, the pulses are 500-µs each, and voltage resolution is 50 mV), there is a difference between the programmed and the target current.

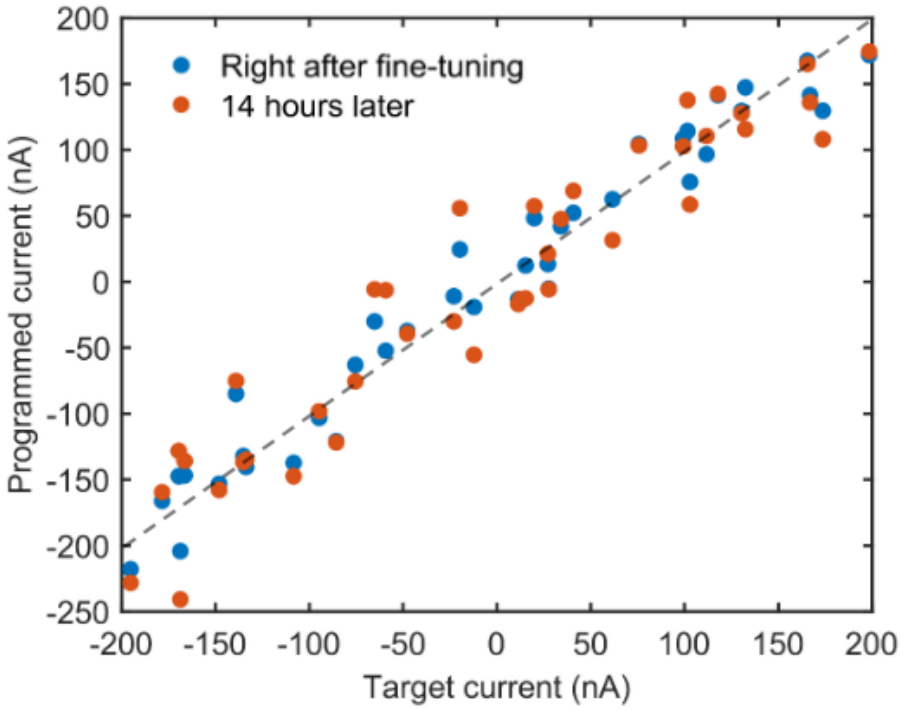

Figure 4.6 The programmed twin-CTT cell current vs. the target current, right after 22 programming-tuning cycles (blue) and 14 hours later.

It is observed that, right after the programming, the programmed currents were closer to the ideal line than 14 hours later. Fig. 4.7 shows the histogram of the difference between the programmed and the target currents. It is clear that the distribution of the difference becomes wider after 14 hours. Indeed, the standard deviation of the difference between the programmed and the target currents is 23.3 nA and 34.9 nA, respectively. This deviation from the target weights after the fine-tuning is attributed to the V_T recovery discussed in Section 2.2.3. It can be easily corrected by applying a few more pulses to the CTTs. Fig. 4.8 shows the programmed vs. target current after 8 more programming pulses, where the dots are pushed closer to the ideal line again. The standard deviation between the target and programmed current after the correction pulses is 22.1 nA,

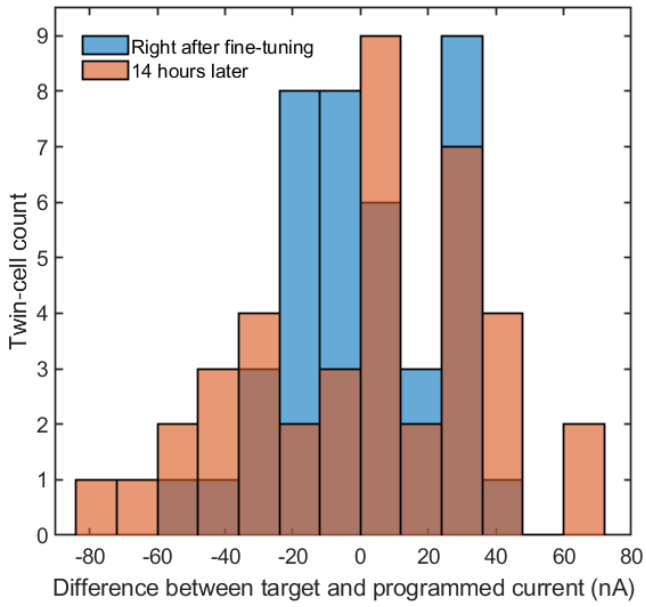

Figure 4.7 The histogram of the difference between the programmed current and target current, right after fine-tuning and 14 hours later.

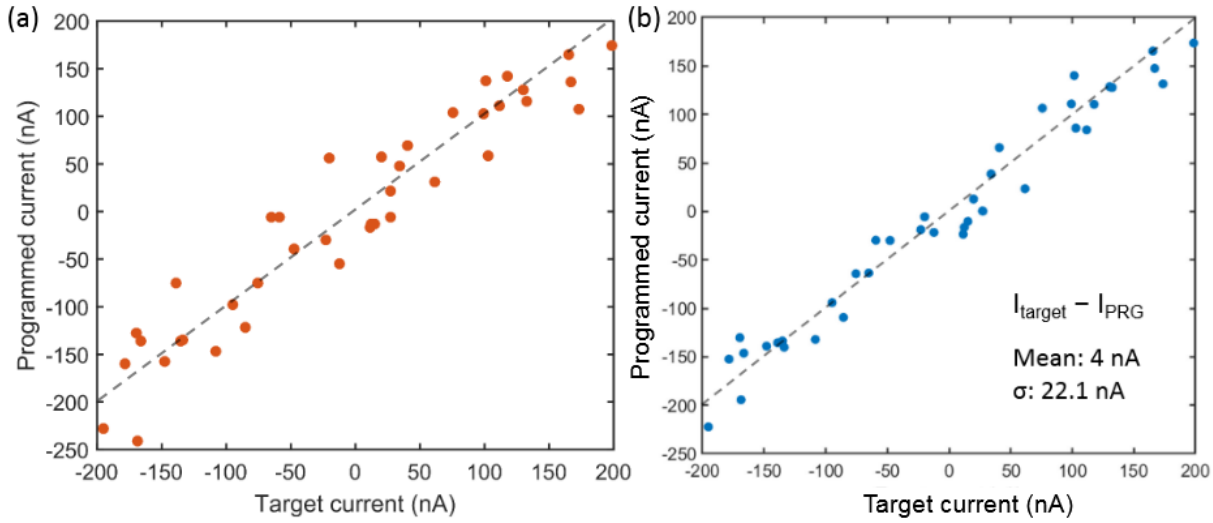

Figure 4.8 The programmed vs. target current, (a) 14 hours after fine-tuning, and (b) after applying 8 extra programming pulses after the 14 hours.

In another experiment, to alleviate the V_T recovery issue, the target CTTs are over-programmed such that their weights will be closer to the target after the recovery. The model in Section 2.2.3 is used to over-program the CTTs and the relationship between the programmed and target currents after different time periods (right after, 6 hours later, and 18 hours later) is shown in Fig. 4.9. The

standard deviations between the programmed and target currents for the three cases are summarized in Table 4.1. It can be seen that the deviation from the ideal behavior is not as severe as that observed when there is no over-programming involved.

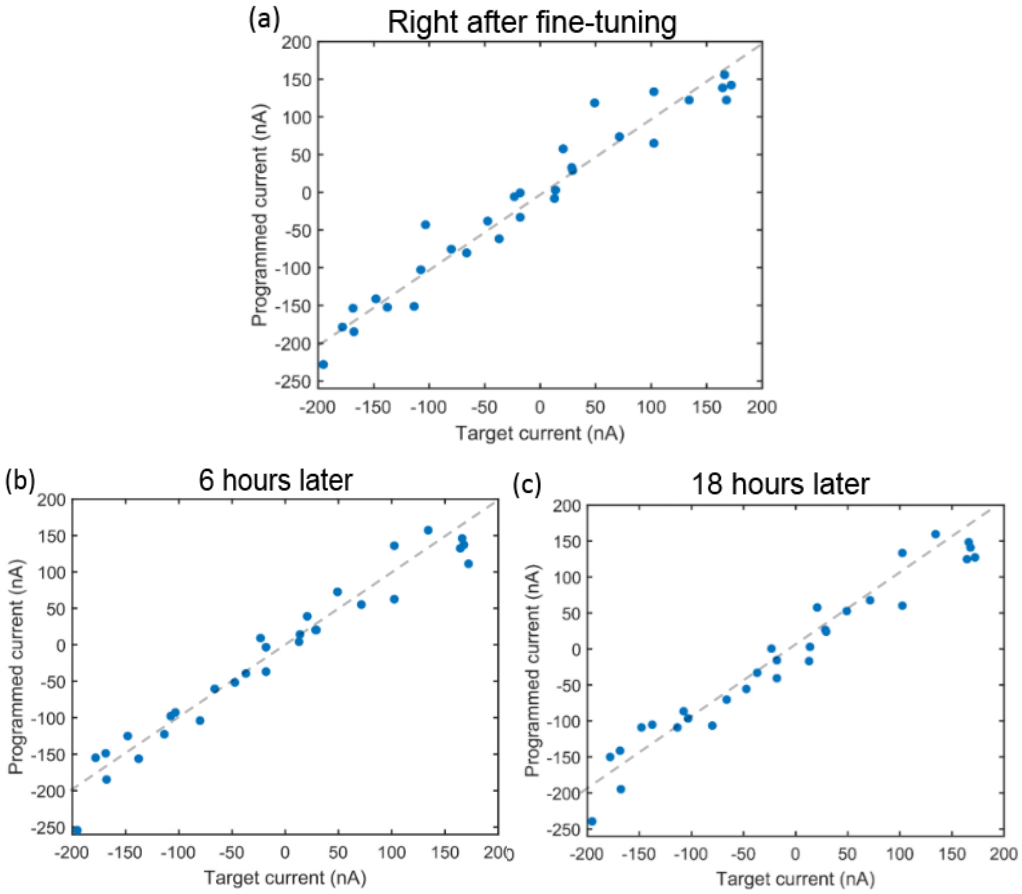

Figure 4.9 The programmed vs. target current with CTT over-programming, (a) right after fine-tuning, (b) 6 hours after fine-tuning, and (c) 18 hours after fine-tuning.

Table 4.1 Summary of the fine-tuning (with CTT over-programming) statistics of 30 twin-CTT cells.

$I_{PRG} - I_{target}$	Right after	6 hours later	18 hours later
Mean (nA)	2.5	4.86	2.4
Std deviation (nA)	27.2	25.1	25.8

To summarize, array-level twin-CTT cell fine-tuning is demonstrated in this Section using the custom-built CTT array. Low (1.8−2.2 V V_G and 1−1.2 V V_D) but variable gate and drain pulses

are needed to avoid the half selection problem. In order to have a good linear relationship between the programmed and target currents after a period of V_T recovery, over-programming of CTT is necessary. A 25.8 nA standard deviation of the difference between the programmed and target currents is demonstrated, which corresponds to 6.45% of the dynamic range of the synaptic weights. More accurate programming is anticipated but with more cycles and finer control of the gate/drain voltage and the programming pulse duration. In Section 4.4, the effect of the difference between the programmed and the ideal weights will be discussed.

4.3 Dot product engine using a CTT array

In this Section, the use of a programmed CTT array as a dot product engine is discussed. The 10×3 twin-CTT array programmed in the previous Section is used as the experimental platform.

Thirty binary test patterns with ten inputs each are applied to the CTT array as WL voltages: V_G = 200 mV is applied when input is 1 and V_G = −300 mV when the input is 0. The current of each column is measured using the configuration shown in Fig. 4.10. Fig. 4.11 shows the relationship between the measured weighted summation of inputs (current) and the ideal one, right after the fine-tuning and 18 hours later. It is worth noting that because the CTTs were intentionally over-programmed in the fine-tuning process, the dot product engine behaves better after 18 hours. The standard deviation of the difference between the measured and ideal currents decreases from 51.23 nA to 38.81 nA after 8 hours, or from approximately 5% to < 4% of the current range.

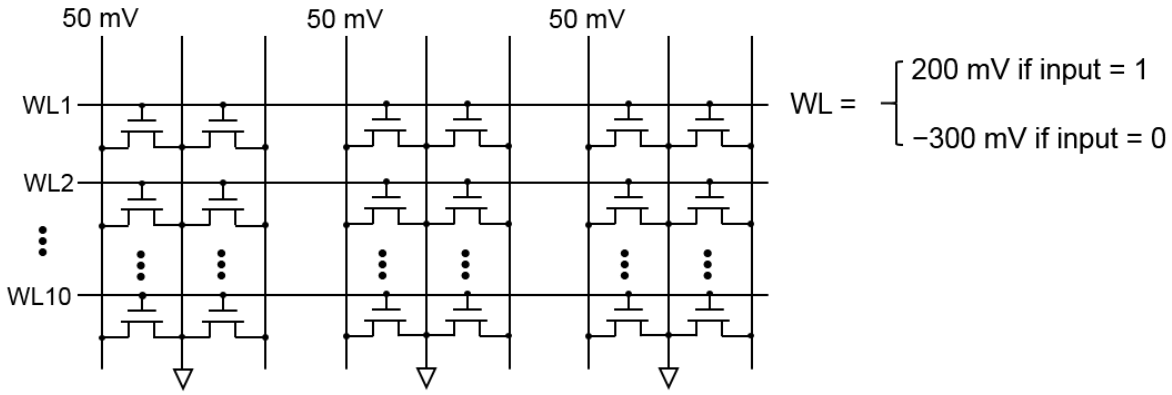

Figure 4.10 The configuration to measure column currents. V_G = 200 mV is applied when input is 1 and V_G = −300 mV when the input is 0.

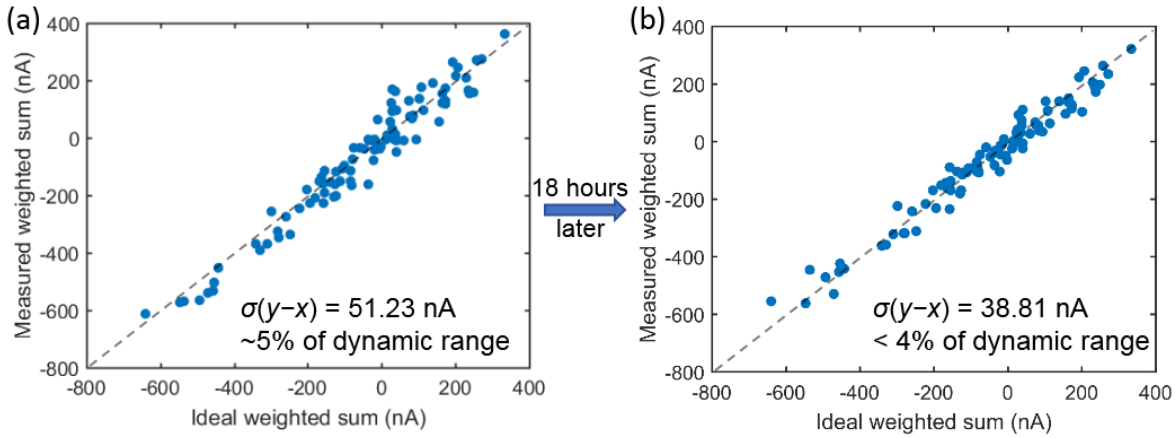

Figure 4.11 The relationship between the measured weighted summation of inputs (current) and the ideal one.

The stability of the dot product engine is also examined. Before measuring the column currents, individual CTT currents at the same bias are first measured in a block of time, t_1 to t_2. The column currents for all 30 test patterns are measured from t_3 to t_4. Here, the scan of individual CTT currents takes approximately 10 minutes and the measurement of all column currents takes around 20 minutes. Because of the V_T recovery within the 30-minute measurement window, a difference in the measured column current and the column current calculated from individual device current is anticipated. Fig. 4.12 shows this difference. It is important to note that, as time goes by, the standard deviation of the difference between the measured and calculated currents becomes

smaller, indicating that the change in device becomes smaller in a set period of time as the devices are rested for longer. After 18 hours for example, the standard deviation is only 11.12 nA, or approximately 1.1% of the current range.

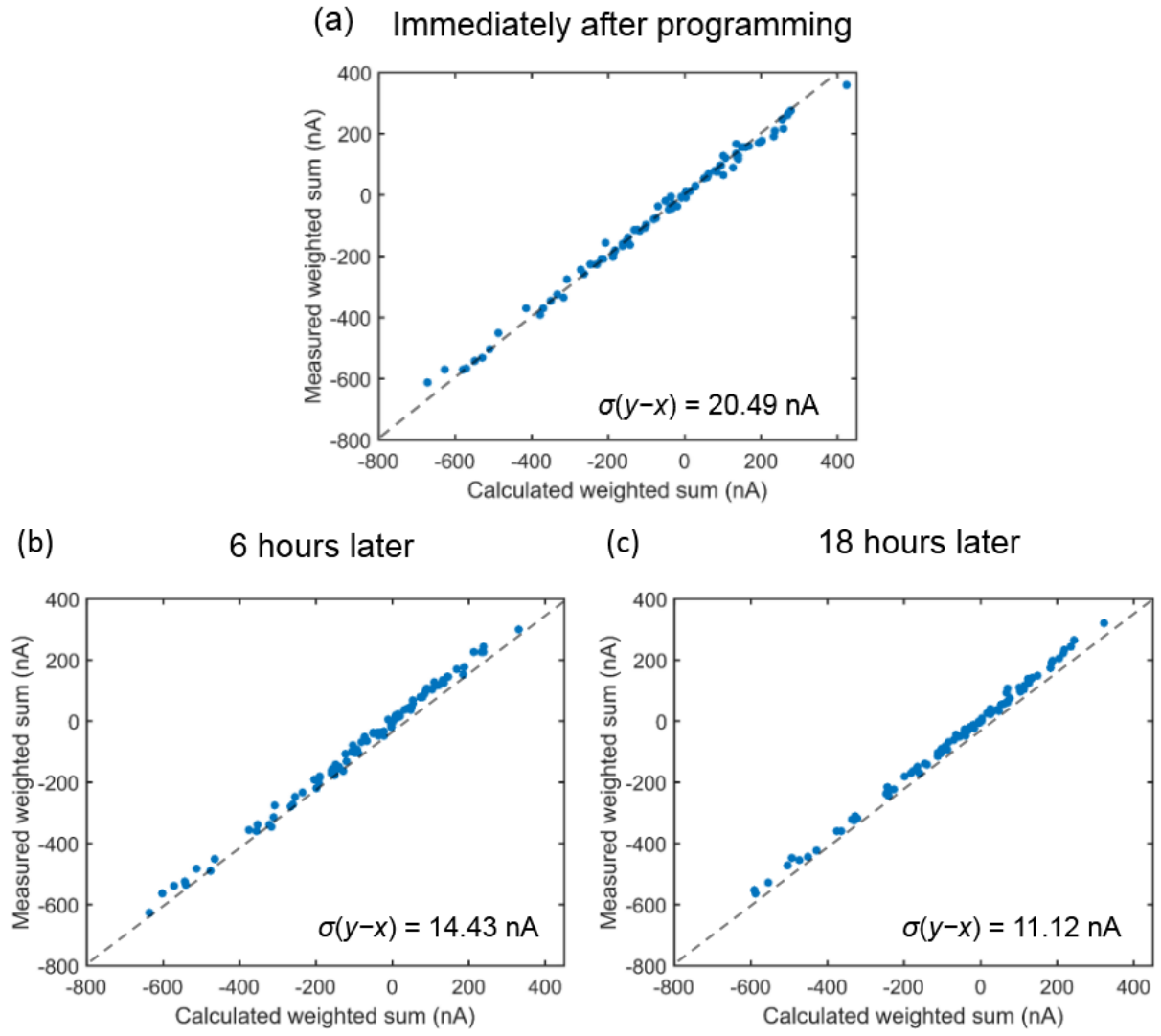

Figure 4.12 The relationship between the measured weighted summation of inputs (current) and the calculated one (using individual CTT currents). (a) Immediately after fine-tuning, (b) 6 hours later, and (c) 18 hours later.

Although the dot product engine has certain inaccuracy, it will be clear in next Section that, even with this kind of inaccuracy, the accuracy of inference is still within a reasonable acceptability, especially for applications where low-power is preferred over high accuracy.

4.4 Consideration of imperfections: effect of weight variation

There are two modes of a neural network: training and inference. Training is typically done in more than 32-bit precision floating point, and almost exclusively using GPU. Although there have been some reports on using emerging memory devices for this purpose [68, 71], the demonstrated network is very small with only tens of synapses, mainly due to substantial variation exhibited by those analog memory devices. For the inference mode, however, it is widely recognized that 8-bit weight precision is sufficient for most applications. For example, Google's tensor processing unit (TPU) uses 8-bit quantization of the trained weights and handles their datacenter's demand very well [72].

In order to have an inference engine utilizing CTTs as the analog synapses, the degradation in the inference accuracy as a result of the imperfect programming of the analog array needs to be evaluated. In this Section, two cases are investigated: a small two-layer neural network for the MNIST digit recognition, and the last fully connected layer in the GoogLeNet [73].

The two-layer network with 100 hidden neurons is trained using the standard back-propagation algorithm with 16-bit precision to achieve an accuracy of 97.3% on the 10,000 testing patterns, after which quantization of the weights in both layers is performed. Fig. 4.13 shows the relationship between the classification accuracy and the number of discrete weight levels. It is observed that when the weights are discretized to 10 levels, the degradation of the classification accuracy from the ideal case is only less than 0.3%. This indicates that the neural network is very robust to weight variations.

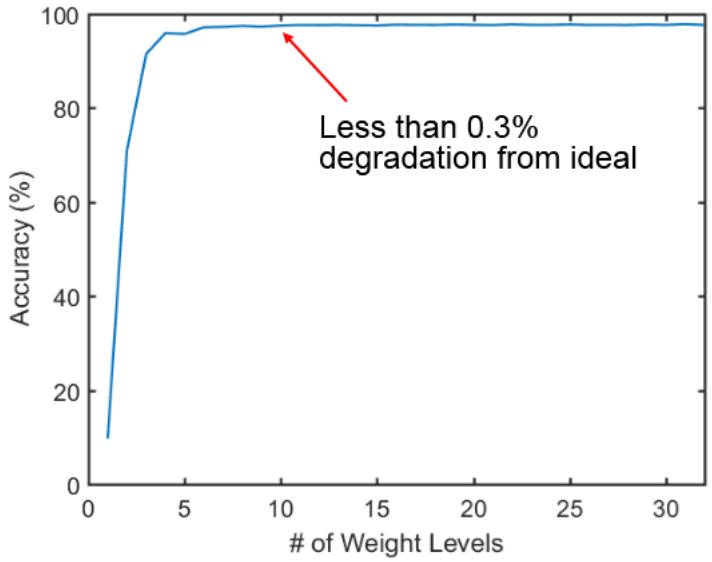

Figure 4.13 The relationship between the classification accuracy and the number of discrete weight levels.

In practice, CTTs are not programmed to discrete weight levels. Instead, they will be programmed to continuous, analog weights, albeit with some imperfections. Fig. 4.14 considers this case. For each percentage of randomness in the weights, 1,000 Monte Carlo simulation runs are performed. In each run, a normally distributed variation (whose standard deviation is equal to the percentage times the weight range) is added to each ideal weight in both layers and the classification accuracy on 10,000 patterns are obtained. In Fig. 4.14, it is observed that the average accuracy of 1,000 simulation runs degrades by less than 1% even when the standard deviation of the random variation is as high as 50% of the weight range. When the standard deviation is 10% of the weight range (considering that the value for CTT is about 6%, as shown in the previous Section), the classification accuracy is 97.28%, a mere 0.02% degradation from the ideal case.

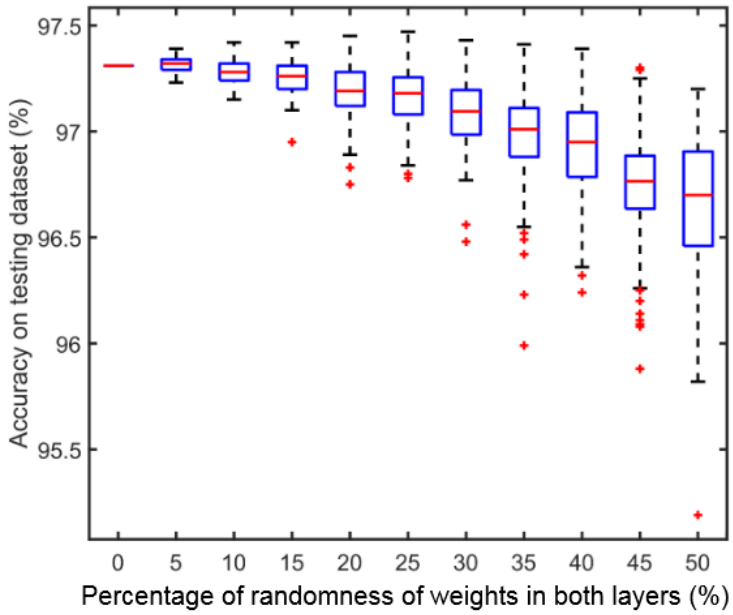

Figure 4.14 The degradation of MNIST classification accuracy as the variation in weights increases.

The impact of the imperfect programming of a CTT array on the classification accuracy of much more complicated neural networks also needs to be evaluated. Here, the last fully connected layer just before the softmax in the GoogLeNet is considered (Fig. 4.15) [73]. GoogLeNet is a deep convolutional neural network aiming at the ImageNet challenge [62].

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
⋮											
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 4.15 The structure of the GoogLeNet with the last fully connected layer highlighted [73].

The inputs to the fully connected layer are calculated from the GPU/CPU trained network and are discretized to 8 bits to emulate a realistic hardware. In hardware, the input information can be in the form of a voltage amplitude applied to the drain of the CTT or a pulse width applied to the gate. Fig. 4.16 shows the degradation of the top 5 and top 1 accuracy as the variation in the weights increases. At σ equal to 10% of the weight range, the top 5 and top 1 accuracy drops from 87.44% to 84.48%, and from 66.3% to 60.13, respectively. Recall that the σ for a CTT array is around 6%, which decreases the accuracy to 86.34%, an equivalent error rate increase of 8.76%.

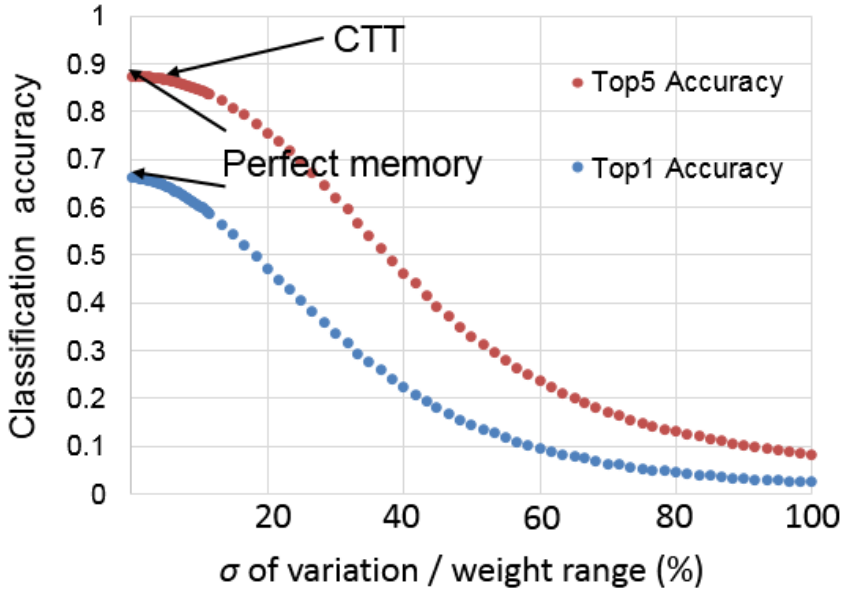

Figure 4.16 The degradation of the Top 5 and Top 1 accuracy as the variation in the weights increases.

To summarize, a fully connected neural network utilizing CTTs as the analog synapses is very robust to the imperfections in the programming of the array. CTT-based synapses provide virtually no accuracy degradation in the MNIST case, and an acceptable degradation even in a very large network.

4.5 Comparison with other analog memory devices

For an analog memory to be successfully used in an inference engine, two properties are of paramount importance: the analog programmability and the power consumption per multiple-and-accumulate (MAC) operation. The former determines the inference accuracy and the latter determines the power efficiency.

The programming accuracy, defined as $A(t) = \sigma/R$, where t is the time, σ is the standard deviation of the difference between the programmed current and the target current, and R is the range of the target current, is a good parameter to quantify the analog programmability. The time dependence of the programming accuracy is a result of the limited analog retention – after programming, the current changes within a certain period of time, as has been discussed in Sections 2.2.3, 4.2 and 4.3. For individual CTTs, with over-programming to account for the current/V_T recovery after programming, 1.6% programming accuracy can be retained one hour after programming. For twin-CTT cells in an array, ~ 6% programming accuracy has been demonstrated 18 hours after programming.

There is very limited data reported in the literature about the programming accuracy of other analog memory devices. Among the existing reports, the best is arguably the embedded flash (eFlash) [45], where the average programming accuracy of three test chips is 5.5%. However, the timeframe for this programming accuracy was not discussed, although the current does drift over long term. The comparison of the programming accuracy between CTTs, eFlash, and two memristive devices is summarized in Table 4.2.

Table 4.2 Summary of the programming accuracy of the CTT, eFlash, and memristor.

Device	Programming accuracy − individual device (%)	Programming accuracy – array (%)	Timeframe	Reference
CTT	1.6	~ 6	1/18 hours	This work
eFlash	0.3 − 30	~ 5.5	N/A	[24, 45]
Memristor	~ 1	~ 5.5	1 s	[23]
Memristor	N/A but current can vary by > 2 times in one hour	N/A	2 hours	[36]

So far, in terms of power efficiency, there is no fair comparison between an analog memory-based inference engine and CPU/GPU-based systems. Most analog memory-based engines to date take digital signals as inputs and are limited to the fully connected network. One reason is that virtually all datasets are created for the digital AI community. A lot of further research−especially on the architecture, is necessary to improve the performance and functionality of an analog inference engine. Therefore, here we only compare the power efficiency between CTT-based and other analog memory-based inference engines.

The power efficiency of an analog memory-based inference engine depends on many factors. The core is the memory array, whose power can be estimated using the largest voltage applied to the device and the resultant current. The peripheral circuitry−including I/O, control, the ADC (if voltage amplitude carries the information) or integrator and comparator (if pulse-width modulation scheme is employed) − is also an important power contributor. Since there is no apparent reason for a CTT-based inference engine to be inferior to others in the peripheral circuitry, especially considering the fact that CTTs are built-in for any advanced-node technologies, here, we only compare the power consumption of the array, without considering the periphery.

A figure of merit (FoM) for energy efficiency is the energy consumption per MAC operation. (J/MAC). Table 4.3 summarizes the comparison between the energy efficiency of CTTs and other

analog memory devices reported in the literature, assuming pulse-width modulation scheme with 8-bit inputs and the shortest pulse of 1 ns.

Table 4.3 Summary of the energy efficiency of the CTTs and other analog memory devices.

	CTT	Memristor [74]	PCM [70]	eFlash [45]	SONOS [75]
Operating Voltage (V)	≤ 0.05	1.2	0.2	2.7	0.6−0.8
Current (A)	10^{-8}–10^{-7}	10^{-6}–10^{-5}	10^{-7}–10^{-6}	10^{-8}–10^{-6}	10^{-7}–10^{-6}
J/MAC	10^{-16}–10^{-15}	10^{-12}–10^{-11}	10^{-14}–10^{-13}	10^{-14}–10^{-13}	10^{-14}–10^{-13}

We have seen that, besides being 100% CMOS-only, logic voltage-compatible, and three-terminal, CTTs also have desirable properties compared to other analog memory candidates. It has as good analog programmability as the eFlash, and the lowest power consumption per MAC owing to its three-terminal nature (such that a very low gate voltage can be applied to reduce the drain current).

5. Conclusions and Future Prospects

5.1 Conclusions

This dissertation provides a comprehensive study on various aspects of CTT-based neuromorphic computing. The main motivation is the CMOS-only and manufacturing-ready nature of the device, which requires no material/process modifications at all in any advanced-node technologies with HfO_2 in the gate dielectric. The three-terminal nature of CTTs also provides an extra knob for tunability and eliminates the need for a selection device.

First, the CTT is characterized for its characteristics that are critical to neuromorphic applications. One important property is the analog retention: after programming, the recovery (up-drifting) of the CTT's current is larger for lower current levels (higher V_T). Also, considering the substantial variation in the devices, verification mechanism after programming is required to fine-tune the CTT to a desired current level. Using the approach developed in Chapter 2, very fine tuning of individual CTTs is demonstrated, with the standard deviation of the difference between programmed and target currents being only 1.6% of the current range. In addition, two desired properties, the spike-timing dependent plasticity and the weight-dependent plasticity, are experimentally demonstrated.

In Chapter 3, two unsupervised learning algorithms − winner-takes-all (WTA) clustering and temporal correlation detection, are studied using CTTs as the analog synapses. The feasibility of implementing the two algorithms using CTTs as the analog synapses is evaluated with experimental data. Both algorithms are experimentally demonstrated with a custom-built CTT array in GlobalFoundries 22nm fully depleted SOI technology. To the best of our knowledge, the

winner-takes-all clustering network, having 27 hardware synapses, is the largest experimental demonstration of the algorithm with emerging memory devices.

Finally, the use of CTTs for a dot product engine, a critical component in any neural networks, is discussed. Array-level fine-tuning of twin-CTT weights is demonstrated, with the standard deviation of the difference between the programmed and the target currents being ~ 6% of the weight range, regardless of the half selection issue and thermal disturbance from adjacent cells. The 6% standard deviation is equivalent to 4-bit precision in a digital system. The operation of the programmed array as a dot product engine is measured and it is shown that the array becomes more stable over time. The implications of the imperfect programming in the accuracy of an inference engine are studied in a two-layer network for MNIST digit recognition and the last fully connected layer of GoogLeNet for ImageNet challenge. There is virtually no accuracy degradation in the MNIST case and an acceptable 8.76% error rate increase in the ImageNet case.

In conclusion, it has been shown in this dissertation that, the charge-trap transistor possesses many desired characteristics to be used as an analog synapse device. Experimental studies demonstrate the feasibility of using the device for both unsupervised and supervised learning applications.

5.2 Future prospects

Three major directions of future research are suggested below:

- One important question this dissertation has not discussed is that, how to build an energy-efficient convolutional neural network using analog memory. The answer to this question is not necessarily dependent on what analog memory is involved. The analog computing community has been actively pursuing the answer for a decade, and when there is one, all analog devices can

be put into perspective and evaluated for their pros and cons for a particular application. Clearly this is an important problem but beyond the scope of this dissertation.

- Although it has been shown that the accuracy degradation of an inference engine due to the imperfect CTT programming is not devastating in a small-scale fully connected neural network (up to 4 million synapses), studies need to be done at scale. Specifically, the analog resiliency of larger and deeper neural networks needs to be investigated, considering the effect of not only the imperfection of programming, but also the imperfection of peripheral circuits (e.g. current integrator). This topic has been largely overlooked by the AI community until very recently [76–78], mainly because most of the networks are run on GPUs and CPUs where the weight inaccuracy is minimal. Most of the existing approaches to addressing the weight uncertainty and quantized weights aim at model compression or run-time acceleration [79, 80]. Moving forward, when CTTs are used as analog synapses in a neural network, network-device co-optimization is anticipated to address this issue [81].
- Towards a very large scale integrated neuromorphic system, the silicon interconnect fabric (Si-IF) might be necessary to reduce the communication cost between different components of the system to maintain the energy-efficiency for the scaled-out system. The system architecture and communication protocols are areas for future investigation.

References

[1] Kelin Kuhn, “High Mobility Materials for CMOS Applications” in “*High Mobility Materials for CMOS Applications*,” Elsevier, 2018.

[2] S. S. Iyer, “Heterogeneous Integration for Performance and Scaling,” in *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, no. 7, pp. 973-982, July 2016. doi: 10.1109/TCPMT.2015.2511626

[3] Z. Wan and S. S. Iyer, “Three-dimensional wafer scale integration for ultra-large-scale cognitive systems,” *2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, Burlingame, CA, 2017, pp. 1-2. doi: 10.1109/S3S.2017.8309199

[4] A. A. Bajwa *et al*., “Heterogeneous Integration at Fine Pitch (≤ 10 μm) Using Thermal Compression Bonding,” *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*, Orlando, FL, 2017, pp. 1276-1284. doi: 10.1109/ECTC.2017.240

[5] D. S. Modha, R. Ananthanarayanan, S. K. Esser, A. Ndirango, A. J. Sherbondy, and R. Singh, “Cognitive computing,” *Commun. ACM*, vol. 54, pp. 62–71, Aug. 2011, doi: 10.1145/1978542.1978559

[6] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, “Convolutional networks for fast, energy-efficient neuromorphic computing,” *Proc. Nat. Acad. Sci. USA*, vol. 113, pp. 11441–11446, Aug. 2016, doi: 10.1073/pnas.1604850113

[7] S. B. Furber, “Brain-inspired computing,” *IET Comput. Digit. Techn*., vol. 10, no. 6, pp. 299–305, Nov. 2016, doi: 10.1049/iet-cdt.2015.0171

[8] A. Calimera, E. Macii, and M. Poncino, "The human brain project and neuromorphic computing," *Funct. Neurol.*, vol. 28, pp. 191–196, Jul./Sep. 2013, doi: 10.11138/FNeur/2013.28.3.191

[9] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, pp. 668–673, Aug. 2014, doi: 10.1126/science.1254642

[10] P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. P. Wong, and H. Qian, "Face classification using electronic synapses," *Nature Commun.*, vol. 8, May 2017, Art. no. 15199, doi: 10.1038/ncomms15199.

[11] R. F. Lyon and C. Mead, "An analog electronic cochlea," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1119-1134, July 1988. doi: 10.1109/29.1639

[12] C. Mead, "Neuromorphic electronic systems," in *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629-1636, Oct. 1990. doi: 10.1109/5.58356

[13] M. A. Mahowald and C. Mead, "The Silicon Retina," *Scientific American*. 264 (5): 76–82, May 1991.

[14] B B. V. Benjamin *et al*., "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," in *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699-716, May 2014. doi: 10.1109/JPROC.2014.2313565

[15] N. Srinivasa and J. M. Cruz-Albrecht, "Neuromorphic Adaptive Plastic Scalable Electronics: Analog Learning Systems," in *IEEE Pulse*, vol. 3, no. 1, pp. 51-56, Jan. 2012. doi: 10.1109/MPUL.2011.2175639

[16] E. Painkras *et al*., "SpiNNaker: A multi-core System-on-Chip for massively-parallel neural net simulation," *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, San Jose, CA, 2012, pp. 1-4. doi: 10.1109/CICC.2012.6330636

[17] T. Sharp, F. Galluppi, A. Rast, S. Furber, "Power-efficient simulation of detailed cortical microcircuits on SpiNNaker," *Journal of Neuroscience Methods*, vol. 210, pp. 110–118, 2010.

[18] F. Walter, F. Röhrbein, A. Knoll, "Neuromorphic implementations of neurobiological learning algorithms for spiking neural networks," *Neural Networks*, vol. 72, pp. 152−167, 2015.

[19] D. B. Strukov, G. S. Snider, D. R. Stewart and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, May 2008.

[20] S. Mandal, A. El-Amin, K. Alexander, B. Rajendran, and R. Jha, "Novel synaptic memory device for neuromorphic computing," *Sci. Rep.*, vol. 4, p. 5333, Jun. 2014. doi: 10.1038/srep05333

[21] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. W. Burr, N. Sosa, A. Ray, J.-P. Han, C. Miller, K. Hosokawa, and C. Lam, "NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning," in *IEDM Tech. Dig.*, Washington, DC, USA, Dec. 2015, pp. 17.1.1–17.1.4, doi: 10.1109/IEDM.2015.7409716

[22] S. B. Eryilmaz, E. Neftci, S. Joshi, S. Kim, M. BrightSky, H.-L. Lung, C. Lam, G. Cauwenberghs, and H.-S. P. Wong, "Training a probabilistic graphical model with resistive switching electronic synapses," *IEEE Trans. Electron Devices*, vol. 63, no. 12, pp. 5004–5011, Dec. 2016, doi: 10.1109/TED.2016.2616483

[23] G. C. Adam, B. D. Hoskins, M. Prezioso, F. Merrikh-Bayat, B. Chakrabarti and D. B. Strukov, "3-D Memristor Crossbars for Analog and Neuromorphic Computing Applications," in *IEEE Transactions on Electron Devices*, vol. 64, no. 1, pp. 312-318, Jan. 2017. doi: 10.1109/TED.2016.2630925

[24] X. Guo, “Mixed Signal Neurocomputing Based on Floating-gate Memories,” UCSB Dissertation, 2017.

[25] L. Chua, “Memristor-The missing circuit element,” in *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507-519, September 1971. doi: 10.1109/TCT.1971.1083337

[26] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder and W. Lu, “Nanoscale Memristor Device as Synapse in Neuromorphic Systems,” *Nano Lett.* 2010, 10, 1297–1301.

[27] H. -. P. Wong *et al*., “Metal–Oxide RRAM,” in *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951-1970, June 2012. doi: 10.1109/JPROC.2012.2190369

[28] A. Chanthbouala, *et al.*, “A ferroelectric memristor,” *Nature Materials*, vol. 11, pp. 860−864, October 2012.

[29] D. Walczyk, Ch. Walczyk, T. Schroeder, T. Bertaud, M. Sowinska, M. Lukosius, M. Fraschke, B. Tillack, Ch. Wenger, “Resistive switching characteristics of CMOS embedded HfO2-based 1T1R cells,” *Microelectronic Engineering* vol. 88, pp. 1133–1135, 2011.

[30] Y. Wu *et al*., “On the Bipolar Resistive Switching Memory Using TiN/Hf/HfO_2/Si MIS Structure,” in *IEEE Electron Device Letters*, vol. 34, no. 3, pp. 414-416, March 2013. doi: 10.1109/LED.2013.2241726

[31] M.Y. Chan, T. Zhang, V. Ho, P.S. Lee, “Resistive switching effects of HfO2 high-k dielectric,” *Microelectronic Engineering* vol. 85, pp. 2420–2424, 2008.

[32] H. Y. Lee *et al*., “Evidence and solution of over-RESET problem for HfO_x based resistive memory with sub-ns switching speed and high endurance,” *2010 International Electron Devices Meeting*, San Francisco, CA, 2010, pp. 19.7.1-19.7.4. doi: 10.1109/IEDM.2010.5703395

[33] J. J. Yang, D. B. Strukov and D. R. Stewart, “Memristive devices for computing,” *Nature Nanotechnology*, vol. 8, pp. 13−24, January 2013.

[34] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang and H.-S.Philip Wong, "Stochastic learning in oxide binary synaptic device for neuromorphic computing," *Frontiers in Neuroscience*, vol. 7, October 2013.

[35] S.G. Hu, Y. Liu, Z. Liu, T.P. Chen, J.J. Wang, Q. Yu, L.J. Deng, Y. Yin and Sumio Hosaka, "Associative memory realized by a reconfigurable memristive Hopfield neural network," *Nature Communications*, 6:7522, 2015.

[36] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum and H. -. P. Wong, "An Electronic Synapse Device Based on Metal Oxide Resistive Switching Memory for Neuromorphic Computation," in *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2729-2737, Aug. 2011. doi: 10.1109/TED.2011.2147791

[37] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla, B. Rajendran, S. Raoux, R. S. Shenoy, "Phase change memory technology," *J. Vac. Sci. Technol. B*, 28(2) 2010.

[38] H. -. P. Wong *et al*., "Phase Change Memory," in *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201-2227, Dec. 2010. doi: 10.1109/JPROC.2010.2070050

[39] S. Park *et al*., "RRAM-based synapse for neuromorphic system with pattern recognition function," *2012 International Electron Devices Meeting*, San Francisco, CA, 2012, pp. 10.2.1-10.2.4. doi: 10.1109/IEDM.2012.6479016

[40] K. Seo, I. Kim, S. Jung, M. Jo, S. Park, J. Park, J. Shin, K. P. Biju, J. Kong and K. Lee, "Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device," *Nanotechnology* 22, 254023, 2011.

[41] R. Yang, K. Terabe, G. Liu, T. Tsuruoka, T. Hasegawa, J. K. Gimzewski and M. Aono, "On-demand nanodevice with electrical and neuromorphic multifunction realized by local ion migration," *ACS Nano* 6, 9515–21, 2012.

[42] D. Kuzum, R. G. D. Jeyasingh, B. Lee and H.-S. P. Wong, "Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing," *Nano Lett.* 2012, 12, 2179–2186.

[43] Y. Zhang, J. Feng, Y. Zhang, Z. Zhang, Y. Lin, T. Tang, B. Cai, and B. Chen, "Multi-bit storage in reset process of Phase-change Random Access Memory (PRAM)," *Phys. Stat. Sol., Rapid Res. Lett.*, vol. 1, no. 1, pp. R28–R30, 2007.

[44] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: Materials, devices, and applications," *Nanotechnology*, vol. 24, pp. 382001-1–382001-22, Sep. 2013, doi: 10.1088/0957-4484/24/38/ 382001.

[45] X. Guo *et al*., "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology," *2017 IEEE International Electron Devices Meeting (IEDM)*, San Francisco, CA, 2017, pp. 6.5.1-6.5.4. doi: 10.1109/IEDM.2017.8268341

[46] https://www.mythic-ai.com

[47] https://www.syntiant.com

[48] Byoung Hun Lee *et al*., "Ultrathin hafnium oxide with low leakage and excellent reliability for alternative gate dielectric application," *International Electron Devices Meeting 1999. Technical Digest (Cat. No.99CH36318)*, Washington, DC, USA, 1999, pp. 133-136. doi: 10.1109/IEDM.1999.823863

[49] K. Mistry *et al*., "A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging," *2007 IEEE International Electron Devices Meeting*, Washington, DC, 2007, pp. 247-250. doi: 10.1109/IEDM.2007.4418914

[50] E. P. Gusev *et al*., "Ultrathin high-K gate stacks for advanced CMOS devices," *International Electron Devices Meeting. Technical Digest (Cat. No.01CH37224)*, Washington, DC, USA, 2001, pp. 20.1.1-20.1.4. doi: 10.1109/IEDM.2001.979537

[51] W. J. Zhu, T. P. Ma, S. Zafar and T. Tamagawa, "Charge trapping in ultrathin hafnium oxide," in *IEEE Electron Device Letters*, vol. 23, no. 10, pp. 597-599, Oct. 2002. doi: 10.1109/LED.2002.804029

[52] E.P. Gusev, C. D'Emic, S. Zafar, A. Kumar, "Charge trapping and detrapping in HfO2 high-j gate stacks," *Microelectronic Engineering* 72 (2004) 273–277.

[53] E. Cartier, B. P. Linder, V. Narayanan and V. K. Paruchuri, "Fundamental understanding and optimization of PBTI in nFETs with SiO2/HfO2 gate stack," *2006 International Electron Devices Meeting*, San Francisco, CA, 2006, pp. 1-4. doi: 10.1109/IEDM.2006.346773

[54] Kenji Shiraishi, Keisaku Yamada, Kazuyoshi Torii, Yasushi Akasaka, Kiyomi Nakajima, Mitsuru Konno, Toyohiro Chikyow, Hiroshi Kitajima, Tsunetoshi Arikado, Yasuo Nara, "Oxygen-vacancy-induced threshold voltage shifts in Hf-related high-k gate stacks," *Thin Solid Films* 508 (2006) 305 – 310.

[55] Y. Liu, A. Shanware, L. Colombo and R. Dutton, "Modeling of charge trapping induced threshold-voltage instability in high-*k* gate dielectric FETs," in *IEEE Electron Device Letters*, vol. 27, no. 6, pp. 489-491, June 2006. doi: 10.1109/LED.2006.874760

[56] D. Heh, C. D. Young and G. Bersuker, "Experimental Evidence of the Fast and Slow Charge Trapping/Detrapping Processes in High-k Dielectrics Subjected to PBTI Stress," in *IEEE Electron Device Letters*, vol. 29, no. 2, pp. 180-182, Feb. 2008. doi: 10.1109/LED.2007.914088

[57] F. Khan, E. Cartier, C. Kothandaraman, J. C. Scott, J. C. S. Woo, and S. S. Iyer, "The impact of self-heating on charge trapping in high-κ-metal-gate nFETs," *IEEE Electron Device Lett.*, vol. 37, no. 1, pp. 88–91, Jan. 2016, doi: 10.1109/LED.2015.2504952.

[58] F. Khan, E. Cartier, J. C. S. Woo, and S. S. Iyer, "Charge trap transistor (CTT): An embedded fully logic-compatible multiple-time programmable non-volatile memory element for high-κ-metal-gate CMOS technologies," *IEEE Electron Device Lett.*, vol. 38, no. 1, pp. 44–47, Jan. 2017, doi: 10.1109/LED.2016.2633490.

[59] J. Viraraghavan *et al*., "80Kb 10ns read cycle logic Embedded High-K charge trap Multi-Time-Programmable Memory scalable to 14nm FIN with no added process complexity," *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, Honolulu, HI, 2016, pp. 1-2. doi: 10.1109/VLSIC.2016.7573462

[60] B. Jayaraman *et al*., "80-kb Logic Embedded High-K Charge Trap Transistor-Based Multi-Time-Programmable Memory with No Added Process Complexity," in *IEEE Journal of Solid-State Circuits*, vol. 53, no. 3, pp. 949-960, March 2018. doi: 10.1109/JSSC.2017.2784760

[61] D. A. Dallmann and K. Shenai, "Scaling constraints imposed by self-heating in submicron SOI MOSFET's," in *IEEE Transactions on Electron Devices*, vol. 42, no. 3, pp. 489-496, March 1995. doi: 10.1109/16.368045

[62] O. Russakovsky, *et al*., "ImageNet Large Scale Visual Recognition Challenge," *IJCV,* 2015.

[63] T. Chang, S.-H. Jo and W. Lu, "Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor," *ACS Nano*, 5 (9), pp. 7669−7676, 2011.

[64] G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10 464–10 472, Dec. 1998.

[65] X. Gu and S. S. Iyer, "Unsupervised Learning Using Charge-Trap Transistors," in *IEEE Electron Device Letters*, vol. 38, no. 9, pp. 1204-1207, Sept. 2017. doi: 10.1109/LED.2017.2723319

[66] A. Serb, J. Bill, A. Khiat, R. Berdan, R. Legenstein, and T. Prodromakis, "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," *Nature Commun.*, vol. 7, Sep. 2016, Art. no. 12611, doi: 10.1038/ncomms12611.

[67] C. Kim *et al*., "Demonstration of Unsupervised Learning With Spike-Timing-Dependent Plasticity Using a TFT-Type NOR Flash Memory Array," in *IEEE Transactions on Electron Devices*, vol. 65, no. 5, pp. 1774-1780, May 2018. doi: 10.1109/TED.2018.2817266

[68] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, pp. 61–64, May 2015, doi: 10.1038/nature14441.

[69] J. Bill and R. Legenstein, "A compound memristive synapse model for statistical learning through STDP in spiking neural networks," *Front. Neurosci.*, Dec. 2014, 8:412.

[70] Sebastian *et al*., "Temporal correlation detection using computational phase-change memory," *Nat. Comm.*, 1115 (2017).

[71] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi and G. W. Burr, "Equivalent-accuracy accelerated neural network training using analogue memory," *Nature* 558, 60 (2018).

[72] Jouppi *et al*., "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of ISCA '17*, Toronto, ON, Canada, June 24-28, 2017, 12 pages.

[73] Szegedy *et al*., "Going deeper with convolutions," arXiv:1409.4842, 2014.

[74] Y. Kim, Y. Zhang and P. Li. "A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing," *ACM J. Emerg. Technol. Comput. Syst.* 11, 4, Article 38 (April 2015), 25 pages. doi: http://dx.doi.org/10.1145/2700234

[75] L. Fick, D. Blaauw, D. Sylvester, S. Skrzyniarz, M. Parikh and D. Fick, “Analog in-memory subthreshold deep neural network accelerator,” *2017 IEEE Custom Integrated Circuits Conference (CICC)*, Austin, TX, 2017, pp. 1-4. doi: 10.1109/CICC.2017.7993629

[76] Reagen, Brandon, et al., "Ares: A framework for quantifying the resilience of deep neural networks." 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). IEEE, 2018.

[77] Marco Donato et al., “On-Chip Deep Neural Network Storage with Multi-Level eNVM,” In DAC ’18: DAC ’18:The 55th Annual Design Automation Conference 2018, June 24–29, 2018, San Francisco, CA, USA.

[78] I. V. Isaev and S. A. Dolenko, “Training with noise as a method to increase noise resilience of neural network solution of inverse problems,” Opt. Mem. Neural Networks (2016) 25: 142.

[79] C. Blundell, J. Cornebise, K. Kavukcuoglu and D. Wierstra, “Weight Uncertainty in Neural Networks,” Proceedings of the 32nd International Conference on Machine Learning. Lille, France, 2015.

[80] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv and Y. Bengio, “Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations,” Journal of Machine Learning Research 18 (2018) 1–30.

[81] Private discussions with Prof. Iyer, Prof. Roychowdhury, and Zhe Wan.